UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

MODELAGEM CONCEITUAL DE PROCESSOS INDUSTRIAIS COM APLICAÇÕES

Raphaela Galhardo Fernandes Lima

Orientador: Prof. Dr. Luiz Affonso H. Guedes de Oliveira

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutora em Engenharia Elétrica e de Computação.

Número de ordem PPgEE: M110
Natal, RN, Fevereiro de 2014

UFRN / Biblioteca Central Zila Mamede.
Catalogação da Publicação na Fonte.

Lima, Raphaela Galhardo Fernandes.
Modelagem conceitual de processos industriais com aplicações.
/Raphaela Galhardo Fernandes Lima. – Natal, RN, 2014.
116 f.: il.

Orientador: Prof. Dr. Luiz Affonso H. Guedes de Oliveira.

Tese (Doutorado) – Universidade Federal do Rio Grande do Norte.
Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.

1. Processos industriais – Automação industrial - Tese. 2. Ontologia – Tese. 3. JENA – Tese. 4. Protégé SPARQL – Tese. 5. Automação avançada – Tese. I. Oliveira, Luiz Affonso H. Guedes de. II. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM CDU 681.5

MODELAGEM CONCEITUAL DE PROCESSOS INDUSTRIAIS COM APLICAÇÕES

Raphaela Galhardo Fernandes Lima

Tese de Doutorado aprovada em 21 de Fevereiro de 2014 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Affonso H. Guedes de Oliveira(orientador) ..... DCA/UFRN

Prof. Dr. Adrião Duarte Dória Neto ..... DCA/UFRN

Prof[a] Dr[a] Vanja Maria de França Bezerra ..... DEQ/UFRN

Prof[a] Dr[a] Claudia Maria Fernandes Araújo Ribeiro ..... IFRN, UERN

Prof. Dr. Raimundo Santos Moura ..... UFPI

*Dedico esse trabalho a meu avô José (*in memoriam*), que infelizmente não teve a oportunidade de presenciar minha formação em engenharia, mestre e doutora, mas que com certeza está muito orgulhoso da formação que venho conquistando.*

Agradecimentos

Agradeço a minha família que torceu muito para esse dia acontecer, em especial a meu marido, Gleydson Lima.

Ao meu orientador, prof. Affonso, que é, sem dúvida nenhuma, um dos melhores professores que qualquer aluno pode ter. Ensina, orienta, é compreensível e torce com sinceridade para o sucesso dos seus alunos.

Aos meus colegas Guga, Allan e Ivanovitch por toda contribuição que deram a esse trabalho.

À UFRN, por fornecer toda estrutura necessária à minha formação acadêmica.

Resumo

A hipótese principal desta tese é que para o desenvolvimento de aplicações de automação industrial de forma eficiente, é necessário ter uma boa estruturação dos dados que serão manipulados. Então, com o objetivo de estruturar conhecimento envolvido no contexto de processos industriais, esta tese propõe uma ontologia, denominada OntoAuto, que modela conceitualmente os elementos envolvidos na descrição de processos industriais. Para validar a ontologia proposta, são apresentadas diversas aplicações. Na primeira, são modelados conceitualmente dois processos industriais típicos: unidade de tratamento DEA (*Dietanolamina*) e forno industrial. Na segunda aplicação, a ontologia foi utilizada para realizar uma filtragem semântica de alarmes, que aliada as análises de correlações, determina relações temporais entre alarmes de um processo industrial. Na terceira aplicação, a ontologia foi usada para modelagem e análise de custo de construção e operação de processos. Na quarta aplicação, a ontologia é adotada para analisar a confiabilidade e disponibilidade de uma planta industrial. Tanto para a aplicação que envolve custos quanto para a da área de confiabilidade, foi necessário criar novas ontologias, OntoEcon e OntoConf, respectivamentem, que importam o conhecimento representado na OntoAuto, porém acrescentando informações específicas.

Como principais conclusões da tese, tem-se que ontologias são abordagens bastante adequadas para a estruturação do conhecimento sobre processos industriais e baseado nelas, é possível desenvolver diversas aplicações avançadas na área de automação industrial.

Palavras-chave: Processos Industriais, Ontologia, JENA, Protégé, SPARQL, Automação Avançada.

Abstract

The main hypothesis of this thesis is that the development of industrial automation applications efficiently, you need a good structuring of data to be handled. Then, with the aim of structuring knowledge involved in the context of industrial processes, this thesis proposes an ontology called OntoAuto that conceptually models the elements involved in the description of industrial processes. To validate the proposed ontology, several applications are presented. In the first, two typical industrial processes are modeled conceptually: treatment unit DEA (Diethanolamine) and kiln. In the second application, the ontology is used to perform a semantic filtering alarms, which together with the analysis of correlations, provides temporal relationships between alarms from an industrial process. In the third application, the ontology was used for modeling and analysis of construction cost and operation processes. In the fourth application, the ontology is adopted to analyze the reliability and availability of an industrial plant. Both for the application as it involves costs for the area of reliability, it was necessary to create new ontologies, and OntoEcon OntoConf, respectivamentem, importing the knowledge represented in OntoAuto but adding specific information.

The main conclusions of the thesis has been that ontology approaches are well suited for structuring the knowledge of industrial processes and based on them, you can develop various advanced applications in industrial automation.

Keywords: Industrial Process, Ontology, JENA, Protégé, SPARQL.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos e Abreviaturas

CLP : Controlador Lógico Programável;
DAML : DARPA Agente Markup Language;
DEA : Dietanolamina;
EPS : Enterprise Production System;
ERP : Enterprise Resource Planning;
ESS : Emergency Shutdown System;
FMEA : Failure Mode en Effects Analysus;
FTA : Fault Tree Analysis;
GLP : Gás Liquefeito de Petróleo;
GC : Gás Combustível;
KIF : Knowledge Interchange Format;
MTTF : Tempo médio de funcionamento até a ocorrência de um defeito;
MTTR : Tempo médio até o sistema reparar um defeito;
OIL : Ontology Inference Layer;
OWL : Web Ontology Language;
RDF : Resource Description Framework;
RDQL : RDF Data Query Language;
SCADA : Supervisory Control and Data Acquisition;
SPARQL : SPARQL Protocol and RDF Query Language;
SOC : Sistemas de Organização do Conhecimento;
XOL : Ontology Exchange Language.

Capítulo 1

Introdução

O enfoque da automação industrial é a otimização dos processos industriais, aumentando a produtividade e qualidade das tarefas executadas pelo homem, sendo um método de produção onde os operadores humanos são providos de maquinaria para auxiliá-los em seus trabalhos.

É crescente o número de indústrias que utilizam tecnologia da automação, objetivando melhorar sua competitividade, gerar produtos e informações de alta qualidade, reduzir custos, satisfazer as necessidades sociais e econômicas e estar sempre cumprindo as regulamentações ambientais e de segurança. Dois outros objetivos também relevantes são a redução da margem de falhas no processo e a segurança do trabalho. Como exemplos de aplicações industriais, podem ser citadas as aplicações de controle de tráfego aéreo, de gerenciamento de processos químicos, as refinarias de petróleos, as manufaturas de explosivos ou farmacêuticos, entre outras.

A Figura 1.1 apresenta a estrutura básica de um processo industrial [Dunn 2002], formada por:

- Aplicação: entidade física com funções e operações que estão sendo monitoradas e controladas. Plantas ou processos industriais;
- Computador: monitoramento e controle por hardware e software da aplicação em tempo real;
- Sensores: dispositivos que convertem variáveis do processo (quantitativos físicos, por exemplo: temperatura, nível, vazão, etc.) da aplicação em sinais elétricos que servem de entrada para o computador;
- Atuadores: dispositivos que convertem os sinais elétricos da saída do computador para o quantitativo físico que controla a função da aplicação;
- Operador: uma ou mais pessoas responsáveis por monitorar a operação do sistema em tempo real;
- Link de Transferência de Dados: caminho para transferência de dados entre um ou mais computadores.

O processo de automação industrial envolve diversas atividades que devem ocorrer de maneira satisfatória, permitindo que as indústrias sempre atinjam os objetivos citados anteriormente. Um sistema de automação atual pode ser melhor compreendido se o dividirmos em níveis, conforme Figura 1.2 [Filho 1993]. Na base da pirâmide, encontra-se o nível de sensores e atuadores. Estes são responsáveis pela interação direta com o

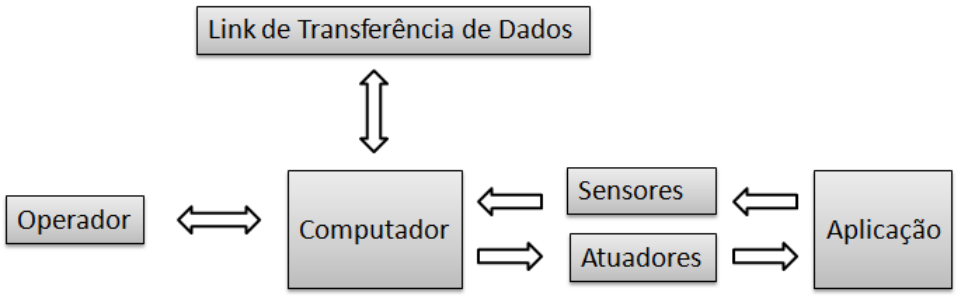


Figura 1.1: Estrutura Básica de um Processo Industrial.

processo, fazendo a leitura das variáveis relevantes através dos sensores e interferindo no processo por intermédio dos atuadores. No nível imediatamente acima, encontram-se os controladores lógico programáveis (CLP) e os *Supervisory Control and Data Acquisition* (SCADA), que realizam o controle regulatório e supervisão, respectivamente. O terceiro nível, *Enterprise Production Systems* (EPS), é o responsável pela gerência de informação. Neste nível, são armazenados dados referentes aos processos que podem ser utilizados como informação útil. No topo da pirâmide estão os sistemas responsáveis pela transformação desses dados em informação de negócio, *Enterprise Resource Planning* (ERP) [Filho 1993].

Figura 1.2: Modelo em camadas de um Sistema de Automação.

Uma atividade bastante importante é a supervisão. Os sistemas supervisórios permitem que as informações de um processo sejam monitoradas e rastreadas. Estas informações são coletadas através de equipamentos de aquisição de dados e, em seguida, manipuladas, analisadas, armazenadas e, posteriormente, exibidas ao operador do sistema. Estes sistemas também são chamados de SCADA [Boyer 2010].

No contexto da automação de processos, o monitoramento e controle de processos industriais são feitos utilizando tecnologias de computação e comunicação, permitindo que os dados sejam coletados em ambientes complexos e sejam representados amigavelmente para o operador.

Com o surgimento de equipamentos de campo cada vez mais inteligentes e baratos, uma grande variedade de dados provenientes desses equipamentos passou a ser disponibilizada em tempo real para os sistemas supervisórios ou mesmo para outras aplicações.

Isso fez com que a área de automação industrial tivesse um novo desafio, o de transformar esse grande volume de dados em informação útil à tomada de decisão, de modo que venha a contribuir efetivamente na melhoria da operação e planejamento do processo como um todo.

Em resumo, a quantidade extraída de informação não é um obstáculo, porém, muitas vezes, a qualidade desse dado pode ser um problema. Em algumas situações, a grande quantidade de dados disponível não é traduzida necessariamente em informação de boa qualidade. Uma maneira de conseguir trabalhar com esses dados com maior eficiência, seria estruturando o conhecimento das plantas industriais, permitindo que informações semânticas sejam extraídas, facilitando o processamento dos dados.

Os sistemas de organização de conhecimento (SOC) abrangem aquilo que pretende organizar e representar o conhecimento, como por exemplo, as taxonomias, tesauros e ontologias. São sistemas conceituais semanticamente estruturados que englobam termos, definições, relacionamentos e propriedades dos conceitos. A partir deles, podemos estruturar o conhecimento envolvido nos processos industriais e, com essa representação do conhecimento, desenvolver aplicações que possam extrair informação de qualidade da grande quantidade de dados que são disponibilizados.

1.1 Motivação e Objetivos

A principal motivação desta tese é representar o conhecimento existente no contexto dos processos industriais. Com isso, será possível desenvolver aplicações nas mais diversas áreas, auxiliando a tomada de decisão dos operadores das plantas industriais. Essas aplicações podem ser aproveitadas em processos diferentes, por serem desenvolvidas em cima de um conhecimento representado de forma genérica.

Assim, o principal objetivo dessa tese é estruturar o conhecimento de processos industriais, permitindo que diversas aplicações possam ser desenvolvidas em cima de uma única arquitetura, sendo reaproveitadas em processos diferentes. Conforme apresentado na Figura 1.3, é proposta nesta tese uma ontologia central (OntoAuto) utilizada para modelar os conceitos e propriedades mais gerais relacionados aos processos industriais, dando especial atenção à caracterização dos componentes que compõem as plantas industriais e dos fluxos de energias que atravessam esses processos. A OntoAuto pode ser importada e utilizada em ontologias para outros contexto de aplicações (OntoEcon, OntoConf, etc.), reutilizando tudo o que tem na ontologia central e complementando com as informações necessárias.

Dois processos industriais bem típicos na literatura, foram modelados com base na OntoAuto e serão detalhados mais a seguir:

- Unidade de Tratamento DEA.
- Fornos Industriais.

Para validar a ontologia com foco nos processos industriais, foram desenvolvidas algumas aplicações nas seguinte áreas:

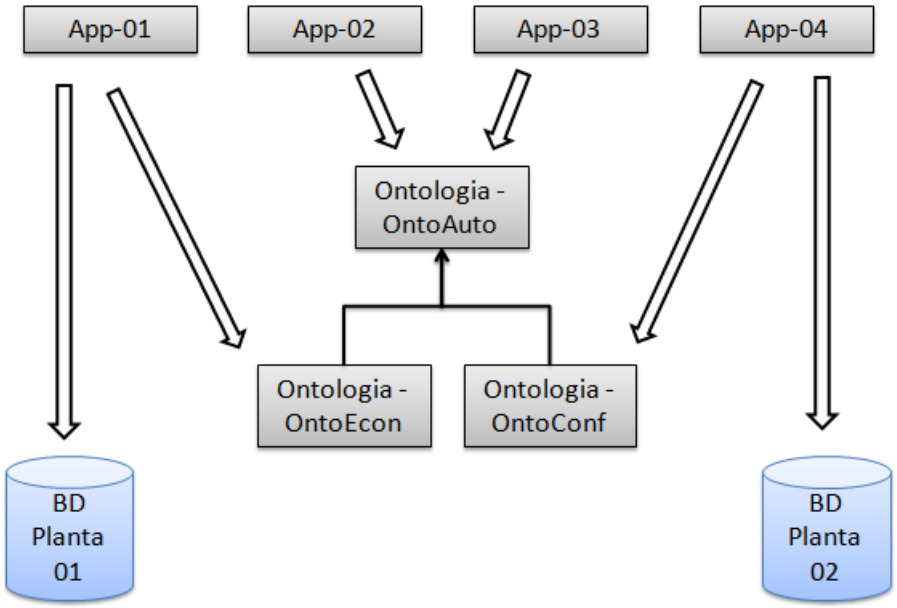


Figura 1.3: Arquitetura da Representação do Conhecimento.

- Correlação semântica de alarmes: a estruturação do conhecimento com a OntoAuto permitiu aprimorar a técnica de correlação de alarmes, incorporando informações semântica e melhorando a qualidade dos resultados. Para essa aplicação, utilizou-se apenas a OntoAuto, onde o conhecimento modelado por ela foi suficiente.
- Avaliação econômica: desenvolvimento de uma aplicação relacionada à etapa de avaliação econômica de projetos de processos industriais. Para este caso, a OntoAuto não possuía todos os conceitos necessários e precisou ser especializada. Então, foi criada uma ontologia, OntoEcon, que importa todos os conceitos e propriedades da OntoAuto e acrescenta novo conceitos e propriedades específicos da área de avaliação econômica.
- Confiabilidade: desenvolvimento de uma aplicação relacionada à área de confiabilidade de processos industriais, permitindo a geração automática de árvores de falhas. Para este caso, a OntoAuto também não possuía todos os conceitos necessários e precisou ser especializada. Então, foi criada uma ontologia, OntoConf, que importa todos os conceitos e propriedades da OntoAuto e acrescenta novo conceitos e propriedades específicos para geração de árvores de falhas.

Com isso, valida-se a hipótese de que é possível desenvolver aplicações utilizando o conhecimento estruturado e de forma genérica. Isso significa que a mesma aplicação poderá ser reaproveitada em processos industriais diferentes. As aplicações também poderão acessar informações presentes nas bases de dados dos processos industriais e interpretar esses dados através do conhecimento que foi estruturado.

A estruturação do conhecimento foi feita utilizando ontologia, já que ela permite representar domínios de conhecimento através de uma linguagem formal, possibilitando que inferências sejam realizadas de forma mais simplificada sobre o conteúdo que ela

representa.

1.2 Contribuições

A principal contribuição desse trabalho é a modelagem conceitual de processos indutriais que possa ser reaproveitada em aplicações de diversos processos, evitando retrabalho no desenvolvimento. Além disso, com propósito de validação da proposta foram desenvolvidas e já estão disponíveis aplicações nas áreas de gerência de alarmes, avaliação econômica e confiabilidade.

1.3 Organização da Tese

Essa tese de doutorado está organizada da seguinte maneira:

- Capítulo 2: descreve a fundamentação teórica envolvida neste trabalho relacionada à representação do conhecimento, com foco em ontologias.
- Capítulo 3: descreve a ontologia OntoAuto desenvolvida com foco na representação dos conhecimentos gerais de processos industriais.
- Capítulo 4: apresenta dois estudos de caso instanciando a OntoAuto para processos indutriais.
- Capítulo 5: apresenta uma aplicação desenvolvida no contexto de gerenciamento de alarmes, utilizando a OntoAuto criada, para realizar correlação semântica de alarmes.
- Capítulo 6: apresenta uma aplicação desenvolvida no contexto de avaliação econômica de processos industriais, utilizando a OntoAuto criada e complementando com alguns conceitos específicos. Com isso, auxiliando projetistas quanto a viabilidade econômica de projetos de processos industriais.
- Capítulo 7: apresenta uma aplicação desenvolvida no contexto de confiabilidade de processos industriais, utilizando a OntoAuto criada e complementando com alguns conceitos específicos. Com isso, permitindo a geração de árvores de falhas a partir da ontologia.
- Capítulo 8: apresenta as conclusões e perspectivas futuras relacionadas ao trabalho desenvolvido.

Capítulo 2

Fundamentação Teórica

2.1 Representação do Conhecimento

Os Sistemas de Organização do Conhecimento (SOC) são sistemas conceituais semanticamente estruturados que englobam termos, definições, relacionamentos e propriedades dos conceitos. Na organização e recuperação da informação, os SOC cumprem o objetivo de padronização terminológica para facilitar e orientar a indexação e os usuários. Quanto à estrutura, variam de um esquema simples até o multidimensional, enquanto que suas funções incluem a eliminação da ambiguidade, controle de sinônimos ou equivalentes e estabelecimento de relacionamentos semânticos entre conceitos [Carlan 2010].

Como exemplos de modelos de representação do conhecimento temos os tesauros, a taxonomia e as ontologias. O tesauro é uma linguagem documentária caracterizada pela especificidade e pela complexidade existente no relacionamento entre os termos que comunicam o conhecimento especializado. [Cavalcanti 1978] define tesauro como "uma lista estruturada de termos associados, empregada por analistas de informação e indexadores, para descrever um documento com a desejada especificidade, no nível de entrada, e para permitir aos pesquisadores a recuperação da informação que procuram".

A palavra taxonomia vem do grego taxis=(ordem) e onoma=(nombre). Para [Martinez 2004], a taxonomia, em um sentido amplo, é "a criação da estrutura (ordem) e dos rótulos (nomes) que ajudam a localizar a informação relevante". Em um sentido mais específico, é "o ordenamento e rotulação de metadados, que permite organizar sistematicamente a informação primária". A estrutura mais citada na literatura para a taxonomia é a hierárquica, sendo uma forma de caracterizá-la na divisão em classes e subclasses, utilizada na construção de ontologias.

Uma ontologia define os termos usados para descrever e representar uma área do conhecimento. Segundo [Gruber 1993], "ontologia é uma especificação formal e explícita de uma conceitualização, o que existe é aquilo que pode ser representado".

Os tesauros propõem um conjunto estruturado de termos sob a base de um sistema de conceitos aptos a organizar conteúdos, auxiliando a representação desse conteúdo e evitando as ambiguidades linguísticas. Já as ontologias possibilitam por meio de aplicações lógicas a construção de modelos computacionais para um determinado domínio de aplicação, fazendo com que os objetivos das ontologias vão além daqueles almejados pelos tesauros. Em relação às taxonomias, elas trabalham no sentido de organizar a informa-

ção, diferente das ontologias que buscam estabelecer relações semânticas entre conceitos, em forma de redes conceituais. Os dois processos de representação do conhecimento são complementares e aperfeiçoam o processo de representação e recuperação da informação.

A utilização de ontologias na representação do conhecimento tem sido bastante difundida. Elas permitem o gerenciamento do conhecimento e a recuperação de conteúdos e informações. Além disso, possibilitam o compartilhamento e reuso de conhecimento comum de informações estruturados. Por tudo isso e também por permitir a interoperabilidade entre sistemas, nesse trabalho, as ontologias foram escolhidas como forma de representar o conhecimento de processos industriais.

2.2 Ontologias

O termo ontologia tem origem no grego ontos=(ser) e logos=(palavra). Este termo foi introduzido por Aristóteles (384-322 a.c). Em seu sentido filosófico, foi introduzido com o objetivo de distinguir o estudo do ser como tal. O Dicionário Oxford de Filosofia define ontologia como "...o termo derivado da palavra grega que significa 'ser', mas usado desde o século XVII para denominar o ramo da metafísica que diz respeito àquilo que existe" [Blackburn & Marcondes 1997].

Existem muitas definições sobre o termo Ontologia na literatura [Uschold & Gruninger 1996], [Gruber 1993], [Bill Swartout & Russ 1996]. Em algumas situações, essas definições podem apresentar pontos de vista diferentes ou até mesmo complementares para uma mesma realidade.

Uma ontologia pode ser definida como uma especificação de uma conceitualização, isto é, uma descrição de conceitos e relações que existem em um domínio de interesse [Sowa 1999], [Guarino 1997]. Em outras palavras, uma ontologia consiste desses conceitos e relações, e suas definições, propriedades e restrições, descritas na forma de axiomas. Ou seja, ela estabelece um vocabulário comum sobre um dado domínio de conhecimento para uma comunidade que se interesse pelo mesmo domínio. Esta definição foi a adotada para o desenvolvimento deste trabalho.

Ontologias podem ser escritas em uma linguagem formal, fornecendo uma descrição exata do conhecimento. O seu vocabulário é definido com base em uma conceitualização, evitanto interpretações ambíguas desse vocabulário.

Além disso, as ontologias criadas podem ser compartilhadas. Dessa forma, é possível definir uma ontologia para um determinado domínio e disponibilizá-la para que outros interessados reaproveitem a ontologia existente, ao invés de ter todo o trabalho de construir uma outra correspondente ao domínio da que já existe.

Existem muitas ontologias disponíveis em formato eletrônico que podem ser importadas para o ambiente de desenvolvimento em uso, como por exemplo as bibliotecas: Ontolígua e DAML.

Também é possível criar uma ontologia que herde uma outra mais genérica, incluindo apenas informações específicas ao seu domínio. Ontologias podem ser compartilhadas para uso, em conjunto, com outras ontologias ou ferramentas, possibilitando também a interoperabilidade.

Para criar, atualizar, reutilizar ontologias de forma prática, é importante usar um editor de ontologias. Existem vários na literatura e um bastante eficiente e escolhido para ser utilizado nesta tese foi o Protégé [Grosso 1999].

Para manipular programaticamente ontologias, ou seja, para manipular o conteúdo semântico proporcionado por uma ontologia de um domínio de interesse, deve-se utilizar um *framework* de programação com este propósito. Esses *frameworks* possuem bibliotecas que permitem programas de computadores interagirem com o conteúdo semântico e instâncias da ontologia, permitindo utilizar os benefícios de motores de inferências e linguagens de consultas. O *framework* utilizado neste trabalho foi o JENA [Reynolds 2009], que utiliza a liguagem Java e permite carregar a ontologia definida para a memória de uma aplicação.

2.2.1 Classificação das Ontologias

As ontologias podem ser classificadas de várias maneiras. Alguns autores as classificam por níveis de generalização, enquanto outros as classificam por categorias ou por tipo de uso.

Abaixo algumas das classificações propostas por esses autores:

- Abordagem quanto à função [R. Mizoguchi & Ikeda 1994]:
  - Ontologias de domínio: vocabulário sobre conceitos, seus relacionamentos, sobre atividades e regras que os governam em cima de uma área de conhecimento.
  - Ontologias de tarefa: vocabulário sistematizado de termos, especificando tarefas que podem ou não estar no mesmo domínio.
  - Ontologias gerais: vocabulário relacionado a coisas, eventos, tempo, espaço, casualidade, comportamentos, etc.
- Abordagem quanto ao grau de formalismo [Uschold & Gruninger 1996]:
  - Ontologias altamente informais: expressa em linguagem natural.
  - Ontologias semi-informais: expressa em linguagem natural de forma restrita e estruturada.
  - Ontologias semiformais: expressa em linguagem artificial definida formalmente.
  - Ontologias rigorosamente formal: terminologias da linguagem definidas com semântica formal, teoremas e provas.
- Abordagem quanto à estrutura [Haav & Lubi 2001]:
  - Ontologias de alto nível: conceitos muitos gerais como espaço, tempo, evento, etc. Em geral, os conceitos são independentes de um problema particular ou domínio, permitindo o seu compartilhamento em uma maior proporção.
  - Ontologias de domínio: vocabulário relacionado a um domínio genérico (Ex.: automóveis, medicina, documentos).
  - Ontologias de tarefa: descrição de uma tarefa ou atividade (Ex.: diagnósticos de doenças).

- Abordagem quanto ao conteúdo [Guarino 1997]
  - Ontologias genéricas: utilizada para a descrição de conceitos bem gerais, por exemplo, espaço, tempo, matéria, objeto, evento, casualidade, comportamento, ação, etc. Estes conceitos são independentes de um problema ou domínio particular;
  - Ontologias de domínio: definem conceituações de domínios particulares, descrevendo o vocabulário relacionado a um domínio genérico, como por exemplo, Direito e Automóveis.
  - Ontologias de tarefas: definem conceituações sobre termos relacionados à execução de uma tarefa específica, independente do domínio em questão.
  - Ontologias de aplicação: definem conceitos necessários à aplicação de uma tarefa em um determinado domínio. Estes conceitos geralmente correspondem a papéis desempenhados por entidades do domínio quando da realização de uma certa atividade;
  - Ontologias de representação: explicam as conceituações que fundamentam os formalismos de representação de conhecimento.

2.2.2 Elementos da Ontologia

Para desenvolver uma ontologia é necessário definir um conjunto de elementos que podem ser divididos em [Noy & McGuinness 2001]:

- Conceito (ou classe): relacionado a determinado domínio. Utilizado para a descrição dos conceitos envolvidos no domínio do problema. Podem expressar qualquer coisa sobre a qual alguma coisa é dita, como uma tarefa, função, ação, estratégia, processo de raciocínio, entre outros. Uma classe também pode ter subclasses que representam conceitos mais específicos que a sua superclasse.
- Propriedade: atributo de um conceito. Um conceito possui um conjunto de propriedades associadas, descrevendo as várias características e atributos vinculados a ele.
- Restrição: representada através de axiomas, define determinados limites para os conceitos de um domínio. As restrições também podem ser utilizadas para representar o "ou" exclusivo (xor), quando instâncias de duas ou mais classes podem se relacionar com instâncias de uma outra classe específica. As restrições podem ainda ser utilizadas para definir melhor a semântica de classes especializadas derivadas de classes gerais.

Uma ontologia juntamente com o conjunto de instâncias individuais de classes constituem uma base de conhecimento.

Em geral, para desenvolver uma ontologia deve-se: definir as classes envolvidas, organizar estas classes em uma hierarquia (definição de subclasses, caso necessário), definir as propriedades das classes e os seus possíveis valores, e por fim, crias instâncias das classes, especificando os valores de suas propriedades.

2.2.3 Construção de Ontologias

O processo de construção de uma ontologia não tem uma forma pré-definida. É um processo iterativo e a ontologia vai sendo complementada à medida que se vai tendo conhecimento do domínio que está sendo modelado. Há vários caminhos que podem ser seguidos e pode ser encontrado na literatura um conjunto diverso de métodos e metodologias existentes. Antes de começar a falar sobre isto, é importante diferenciar os conceitos de "método" e "metodologia".

Um método pode ser definido como um conjunto de processos ou procedimentos ordenados usados na engenharia de um produto ou na realização de um serviço [Oscar Corcho & Gómez-Pérez 2003]. Já uma metodologia é uma série integrada de técnicas ou métodos criando uma teoria geral de sistemas de como uma classe de pensamento pode ser executada [Oscar Corcho & Gómez-Pérez 2003]. Assim, pode-se concluir que metodologia e método são conceitos distintos, já que uma metodologia refere-se ao conhecimento sobre métodos, isto é, determina como e quando uma dada atividade pode ser realizada [Oscar Corcho & Gómez-Pérez 2003].

Há na literatura vários métodos e metodologias para o desenvolvimento de ontologias. Existe também a definição de outros métodos e metodologias voltados à remodelagem, à aprendizagem, à avaliação, à expansão de ontologias [Gómez-Pérez & Rojas 1999], [JU. Kietz & Volz 2000], [Klein & Fensel 2001], [Gómez-Pérez 2000], mas que não serão o foco desta seção.

Entre os métodos existentes, pode ser citado um que foi utilizado para o desenvolvimento do projecto Cyc, baseado em conhecimento e composto pelas fases de extração do conhecimento de senso comum, extração do conhecimento auxiliada ou gerenciada por uma ferramenta computacional [Lenat & Guha 1989]. Outro também que pode ser citado é o que foi proposto por *Uschold e King's*, formado por quatro etapas: identificação da finalidade da ontologia, construção da ontologia, avaliação da ontologia e sua documentação [Uschold & King 1995].

Em relação às metodologias, podem se citar a *Methontology* formada por um grupo de etapas de desenvolvimento (especificação, conceitualização, formalização, integração, implementação e manutenção), um ciclo de vida baseado em evolução de protótipos e técnicas para realizar as atividades de planejamento, desenvolvimento e suporte [Mariano Fernández López & Pazos-Sierra 1999]. Uma outra metodologia é a *On-to-Knowledge* baseada na identificação de metas que devem ser concluídas pelas ferramentas de gerenciamento de conhecimento e na análise de cenários existentes [S. Staab & Sure 2001].

A seguir será apresentada a metodologia *Simple Knowledge-Engineering* [Noy & McGuinness 2001], dividida em sete etapas.

Metodologia *Simple Knowledge-Engineering*

A metodologia *Simple Knowledge-Engineering* (SKEM) [Noy & McGuinness 2001] define que uma ontologia deve ser construída em sete etapas, iniciando com uma espécie de rascunho que é revisado e refinado ao longo das etapas de seu desenvolvimento. O refinamento é feito, pois detalhes novos vão aparecendo e gerando mudanças na modelagem da ontologia. Dessa forma, definindo-se como um processo iterativo.

Para iniciar o desenvolvimento da ontologia, é preciso saber que os conceitos de uma ontologia deve ser semelhantes a objetos (físicos ou lógicos) e a relacionamentos dentro do domínio de interesse. É como definir sujeitos (objetos) e verbos (relacionamentos) em uma sentença que descreve o domínio [Noy & McGuinness 2001].

A seguir serão apresentadas as etapas para a construção de uma ontologia segundo esta metodologia [Noy & McGuinness 2001]:

1. Determinação do domínio e escopo da ontologia: definição do domínio de cobertura da ontologia, de quem utilizará a ontologia, dos tipos de questões que a informação da ontologia deve responder e quem usará e manterá a ontologia. Estas definições podem mudar ao longo do processo de desenvolvimento da ontologia, mas irão ajudar a limitar o escopo do modelo.
2. Possibilidade de reutilização de ontologias existentes: verificar se não há na literatura uma ontologia existente que precisaria ser apenas refinada e extendida para o domínio e escopo particular definidos na etapa 1. Muitas ontologias já estão disponíveis em formato eletrônico podendo ser importadas para o ambiente de desenvolvimento em uso.
3. Enumeração dos termos importantes na ontologia: identificação de termos para os quais será necessário fazer questões ou gerar explicações. Ou seja, quais termos serão interessantes de ser explicados, quais as propriedades que eles devem ter, quais são aqueles que se deseja extrair informações. É importante definir os termos sem se preocupar se há relacionamento entres eles, se eles representam conceitos, etc.
4. Definição das classes e de suas hierarquias: esta definição pode ser feita através de várias abordagens [Uschold & Gruninger 1996]. Uma primeira abordagem é a *top-down*, iniciando com a definição de conceitos mais gerais no domínio e em seguida, partindo para a especialização destes conceitos. Uma segunda, *bottom-up*, iniciando pela definição de conceitos específicos de classes (nas folhas da árvore da hierarquia) e depois, agrupando estas classes em conceitos mais genéricos. Uma outra, que seria uma combinação das duas abordagens citadas anteriormente, definindo-se os conceitos mais importantes primeiro e depois é feita uma generalização e especialização apropriada destes. Definir que abordagem aplicar, em geral, depende da maneira pela qual a pessoa que irá desenvolver a ontologia está acostumada a trabalhar. Independente da abordagem escolhida, deve-se iniciar a definição das classes, com base nos termos enumerados na etapa 3, identificando aqueles que descrevem objetos concretos e não aqueles utilizados para a descrição destes objetos. Estes termos selecionados serão as classes e serão organizados na hierarquia considerando se haverá ou não níveis de especializações.
5. Definição das propriedades das classes: apenas a definição das classes não é suficiente para abranger os itens abordados na etapa 1. É necessária a definição, para cada classe, da estrutura de interna dos conceitos envolvidos. Da lista de termos definida na etapa 3, seleciona-se quais termos representam propriedades das classes, identificando que classe eles descrevem.
6. Definição dos valores das propriedades: as propriedades de uma classe podem assumir diversos valores, desde que estejam em conformidade com os tipos de valores

permitidos (Ex.: String, Number, Boolean, instância de outra classe, etc), com o número de valores (cardinalidade) que eles podem ter, ou dentro de um conjunto possível de valores, entre outras.

7. Criação de instâncias: criação de instâncias individuais das classes presentes na hierarquia definida.

A seção seguinte apresentará alguns linguagens existentes para o desenvolvimento de uma ontologia com base em um método ou metodologia pré-definida.

2.2.4 Linguagens

A criação de uma ontologia é feita com base na identificação e especificação de um conjunto de elementos, como visto na seção Elementos da Ontologia. Dessa forma, é importante que toda esta especificação esteja organizada, possibilitando o seu entendimento e compartilhamento.

Uma ontologia pode ser formalizada de diversas maneiras, como por exemplo, utilizando textos, tabelas e gráficos. Porém, quando a ontologia é construída com o propósito de ser utilizada em sistemas de software, é importante que seja codificada em uma linguagem compreensível a estes sistemas.

Há diferentes linguagens de ontologias, cada uma com suas facilidades e particularidades. As subseções abaixo apresentam uma descrição resumida de algumas linguagens.

RDF (*Resource Description Framework*)/*RDF-Schema*

RDF é um modelo descrito com a utilização do XML [Hjelm 2001]. Sua característica está na forma com que o modelo de dados utiliza metadados para descrever recursos, a fim de apresentar um significado ao recurso. Para visualização da representação do RDF, podem-se utilizar grafos rotulados que são construídos com os objetos: recursos, propriedades, literais e declarações. Este modelo foi caracterizado como um modelo padrão para descrição de recurso com propriedades. O RDF-Schema é uma extenção do RDF e fornece uma descrição de grupos de recursos e os relacionamentos existentes entre eles [Kim et al. 2006]. Por isso, o usuário tem a flexibilidade de criar vocabulários representados por classes e propriedades com características restritas, a fim de serem reaproveitadas em outros modelos.

XOL (*Ontology Exchange Language*)

Esta linguagem pode especificar conceitos, taxonomia e relações binárias [P. Karp & Thomere 1999]. Não possui mecanismos de inferência e foi projetada para facilitar o compartilhamento de ontologias.

OIL (*Ontology Inference Layer*)

É uma linguagem criada para representar semântica de uma maneira acessível por máquinas, modelando domínios de conhecimento em forma de ontologias. Desenvolvida

para ser compatível com os padrões do W3C, incluindo XML e RDF, a linguagem OIL explora as primitivas de modelagem de RDFSchema [M. Klein & Horrocks 2000].

DAML (*DARPA Agente Markup Language*) + OIL

A DAML [Oscar Corcho & Gómez-Pérez 2003] foi criada com o objetivo de desenvolver uma linguagem e ferramentas de modo a facilitar o conceito da Web Semântica.

A versão DAML+OIL [Oscar Corcho & Gómez-Pérez 2003] provê meios para modelar domínios de conhecimento através de ontologias.

KIF (*Knowledge Interchange Format*)

A linguagem KIF tem o objetivo de garantir independência de semântica em uma linguagem gráfica, facilitando a comunicação no domínio de interesse. Suas notações capturam certos axiomas de forma implícita, pois utiliza diferentes tipos de notação para diferentes tipos de associação [Genesereth & Fikes 1992].

Lingo (Linguagem Gráfica para descrever Ontologias)

É uma linguagem de notação gráfica e não formal [Ricardo Falbo & Rocha 1998]. Possui uma meta-ontologia, e a semântica de suas notações pode ser diretamente mapeada em um conjunto equivalente de axiomas. Ao descrever graficamente os elementos da linguagem, está sendo descrito o conjunto de axiomas que eles representam.

OWL (*Web Ontology Language*)

A OWL foi criada para descrever classes e as relações existentes entre elas, além de possibilitar que essas classes sejam reutilizadas, ou herdadas, em documentos Web e aplicações. Uma ontologia OWL pode incluir descrição de classes, propriedades e suas instâncias [Matthew Horridge & Wroe 2004].

A linguagem OWL pode ser dividida em três sub-linguagens:

- OWL *Lite*: utilizada quando é necessário criar uma classificação hierárquica simples e possui menor complexidade formal. Suporta restrições de cardinalidade, porém elas se restringem a valores de 0 ou 1.
- OWL DL: utilizada quando é necessário modelar sistemas com um maior nível de detalhamento e restrições. Ela exige a separação de tipos (classe, propriedade e indivíduo) e permite cardinalidades mínima, máxima e intervalos.
- OWL *Full*: permite o máximo de expressividade e de liberdade sintática do RDF. Também permite aumentar o significado de um vocabulário (RDF ou OWL), porém sem nenhuma garantia computacional (completude, decidibilidade, raciocínio).

2.2.5 Ferramentas

Existem vários editores de ontologias disponíveis na literatura, que além da sua edição, permitem a sua visualização. Entre eles, Protégé [Grosso 1999], OntoEditor [de Oliveira et al. 2007], OilEd [Sean Bechhofer & Stevens 2001].

A seguir será apresentada uma descrição resumida da ferramenta Protégé, já que é bastante utilizada na literatura, uma ferramenta livre em java e de código livre.

Protégé

Protégé é uma ferramenta livre em java, de código aberto, utilizada para a criação, visualização e manipulação de ontologias [Grosso 1999]. Está disponível para *download* no site *http://protege.stanford.edu/*.

O modelo de aquisição de conhecimento da Protégé é abstrato e as ontologias são definidas através de uma interface gráfica, criando-se conceitos do domínio que se deseja representar sob uma estrutura de árvore, organizados em uma hierarquia de subclasses.

A plataforma Protégé suporta a modelagem de ontologias de duas formas, via os editores *Protégé-Frames* e *Protégé-OWL*, descritos resumidamente abaixo:

- Editor *Protégé-Frames* [Sachs 2006]: permite ao usuário construir e popular ontologias baseadas em *frames*, de acordo com o protocolo *Open Knowledge Base Connectivity* (OKBC). Nesta modelagem, a ontologia é formada por um conjunto de classes organizadas hierarquicamente para representar o domínio de conhecimento. Cada classe é formada ainda por um conjunto de propriedades e definições de relacionamentos que podem ter com outras classes. Além disso, é possível definir instâncias destas classes, que representam exemplares individuais dos conceitos definidos pelos valores assumidos por suas propriedades.
- Editor *Protégé-OWL* [Knublauch et al. 2004]: permite ao usuário construir uma ontologia para a Web Semântica, particularmente utilizando a linguagem escrita em OWL. Uma ontologia OWL pode incluir descrição de classes, propriedades e suas intâncias.

As ontologias criadas por esta ferramenta podem ser exportadas para vários formatos, entre eles: RDF, OWL e XML Schema.

2.2.6 Base de Conhecimento e Regras de Inferência

Uma base de conhecimento consiste em um agrupamento de conhecimento representado mediante uma técnica adequada ao sistema em questão. Essas informações podem ser utilizadas na solução dos problemas apresentados pelos clientes, por meio de ferramentas de Inteligência Artificial (IA) ou sistemas especialistas. Neste contexto, uma ontologia para sistemas baseados em conhecimento é uma especificação para os objetos, conceitos, outras entidades e o relacionamento entre eles que possam existir em alguma área de interesse.

Os sistemas especialistas empregam informações nem sempre completas manipulando-as através de métodos de raciocínio simbólico sem seguir modelos numéricos, objetivando

produzir aproximações satisfatórias. Sendo assim, quanto mais completo e corretamente estiver representado o conhecimento, melhor será o resultado do sistema. Para tanto se faz necessário a aquisição de conhecimento, uso de heurísticas, de métodos de representação de conhecimento e de mecanismos de inferência.

Mecanismo de inferência ou motor de inferência é um elemento permanente, que pode ser inclusive reutilizado por vários sistemas especialistas. É a parte responsável pela busca das regras da base de conhecimento para serem avaliadas, direcionando o processo de inferência. O conhecimento deve estar preparado para uma boa interpretação e os objetos devem estar em uma determinada ordem representados por uma árvore de contexto.

Quando se trabalha com ontologias informatizadas, é interessante ter mecanismos automatizados para manipulá-las. Para isso, existe o *framework* Jena [McBride 2009] usado para realizar manipulação, consulta e criação de ontologias. Para o Jena, uma ontologia é representada por um grafo chamado de modelo, formado por um conjunto de triplas (*statements*) que expressam um fato sobre um recurso. Cada nó do grafo representa um recurso (conceitos) e os arcos as propriedades (relacionamentos) da ontologia.

O Jena é uma ferramenta bastante utilizada já que possui um fórum com uma comunidade ativa, além de uma API bem feita. Ela oferece suporte à manipulação de ontologias em RDF e OWL, oferece suporte à inferência de ontologias e permite o armazenamento de ontologias inferidas em um banco de dados. Também possui um mecanismo próprio para a execução de regras, as chamadas *jena rules*, que permite a inferência de novo conhecimento através da execução das mesmas [Reynolds 2009].

O Jena possui classes para executar consultas sobre modelos ontológicos. Essas consultas podem ser feitas em RDQL (*RDF Data Query Language*), que é uma linguagem de consulta sob modelos RDF .

Existe também uma outra linguagem mais avançada para executar queries chamadas SPARQL, que é uma linguagem orientada a dados e uma recomendação do W3C a partir de Janeiro de 2008 e foi utilizada neste trabalho.

O propósito da linguagem SPARQL [John Hebeler & Perez-Lopes 2009] é permitir que arquivos RDF sejam consultados através de uma linguagem SQL *Like*. Permite também ao usuário combinar dados de arquivos RDF, provenientes de diferentes fontes.

2.3 Trabalhos Relacionados

Na literatura há alguns trabalhos que empregam ontologias no contexto de automação de processo industriais. Em [Laallam & Sellami 2007] é apresentado um sistema baseado em ontologia desenvolvido para auxiliar a tomada de decisão em processos que compõem uma estação de compressão de gás. A ideia básica é que os especialistas nos processos que realizam a atividade de supervisão e controle tenham uma forma mais eficiente de compartilhar conhecimento, para isso uma ontologia foi desenvolvida para representar tal conhecimento. Apesar de uma estação de compressão ser formada por vários elementos (turbinas, compressores, duto, chaminé, etc.), a ontologia desenvolvida naquele trabalho restringiu-se a representar o conhecimento da turbina a gás, não apresentando uma generalização para os demais componentes.

Em [Muñoz & Espuña 2010], a ontologia foi aplicada na área de gestão em lote de plantas químicas que envolve a coleta e o processamento de grande quantidades de dados, que são posteriormente analisados. Estes dados podem ser visto como uma fonte valiosa de informação para tomada de decisão, independentemente da utilização da sua análise. O problema é que organizar a representação do conhecimento desses dados não é uma tarefa trivial e objetivando resolver esse problema, utilizou-se ontologia para a descrição semântica dos dados e informações, agilizando a coleta de dados. Na área de gerenciamento de alarmes, o trabalho descrito em [Quintão 2008] desenvolveu um modelo de um Sistema Informatizado de Gerenciamento de Alarmes baseado em Recomendações de Ações (SIGARA), que realiza a filtragem de anomalias críticas, e a apresenta aos operadores do processo apenas as que são relevantes e necessitam tomar uma ação, e também a recomendação personalizada dessas ações em caso de falhas do sistema. Para isso foi utilizada uma ontologia denominada ONTORMA (*Ontology for Reusing Multi-agent Software*) [Lindoso 2006], conforme a metodologia MAAEM (*Multi-Agent Application Engineering Methodology*). Nesse trabalho, a ontologia não foi utilizada para a representação conceitual das plantas industriais, ao invés disto utilizou-se um repositório para o armazenamento dos produtos da engenharia de domínio e de aplicações multiagente.

Ainda na área de gerenciamento de alarmes, em [O. Aizpurúa & Jiménez 2008] desenvolveu-se um trabalho com o objetivo de melhorar a análise dos alarmes por parte do operador, onde a ontologia foi utilizada para criar uma representação padronizada do domínio envolvido. Nesse trabalho, a ontologia foi baseada no fato de que nos sistemas de energia submetidos a situações reais, o conhecimento envolvido na ontologia é muito complexo, sendo necessário ser dividido entre várias ontologias, no caso: ontologia de fluxo, ontologia de lógica de controle, ontologia de eventos, ontologia de alarmes, ontologia de sequência de trip (parada parcial ou completa do sistema) [A. Bernaras & Bartolomé 1996]. Outros trabalhos também vêm sendo desenvolvidos no intuito dessa representação do conhecimento de forma padronizada, como um trabalho que apresenta uma ontologia onde o domínio são os processos envolvidos na exploração de petróleo [R. Du & Wu 2010].

[S. Natarajan & Srinivasan 2012] desenvolveu uma ontologia chamada de OntoSafe aplicada a processos de supervisão, fornecendo um gerenciamento semântico de situações anormais, permitindo a integração de todas as informações necessárias para formar um julgamento coeso sobre a condição de estado do processo.

[Küçük Dilek & Cadirci 2008] propôs uma ontologia aplicada à análise da qualidade da energia elétrica, que é medida com base em um conjunto de parâmetros, como frequência e harmônicos obtidos a partir de transmissão de energia elétrica e sistemas de distribuição. Para o processo de construção da ontologia normas e regulamentos pertinentes foram utilizados. A ontologia proposta também suporta aplicações linguísticas, fornecendo um número de propriedades lingüísticas aplicável a todos os seus conceitos

[Anthony A.R. Diniz & de Melo 2012] propôs o uso do conceito de ontologias para representação de uma unidade de tratamento DEA, combinando informações de seus documentos descritivos, complementadas por um especialista em seu modo de operação. A partir da ontologia foi desenvolvida uma ferramenta para análise de alarmes, que combina os dados de um sistema de gerenciamento de alarmes com as informações sintetizadas no

modelo gerado. Nesse trabalho, a ontologia foi desenvolvida apenas voltada para a unidade de tratamento DEA, não sendo abrangente para outros processos industriais.

As ontologias também vêm sendo aplicadas à engenharia de processos químicos, como por exemplo, o projeto OntoCAPE [Jan Morbach & Marquardt 2007], [Jan Morbach & Marquardt 2009], que desenvolveu uma ontologia de domínio para essa área. Nesse domínio, o projeto, a construção, a operação de plantas químicas são consideradas. A OntoCAPE consiste de um conjunto de modelos parcial interconectados hierarquicamente organizados através de vários níveis de abstração. Algumas das áreas incluídas na modelagem são: substâncias, propriedades termofísicas, unidades de operações, equipamentos, bem como modelos matemáticos. Essa ontologia é bem complexa, porém com relação às aplicações desenvolvidas nessa tese, ela não se mostrou totalmente adequada, pois a forma como modela a associação entre entradas e saídas dos componentes nos processos industriais não era suficiente.

Também existem trabalhos voltados para área de controle de processos. [Novak & Sindelar 2013] desenvolveu uma ferramenta aplicada a controle avançado de processos utilizando ontologias. A premissa fundamental para algoritmo de controle de projeto é implementar e aperfeiçoar um modelo de simulação. Os métodos existentes para descrição da planta utilizados hoje em dia são insuficientes para descrever plantas flexíveis e modernas. Com isso, nesse trabalho o processo industrial foi modelado com ontologias. A solução suporta a integração eficiente de diversas ferramentas de engenharia, tais como simuladores, sistemas SCADA, ou até mesmo software proprietário utilizado em projetos particulares.

Voltado para a melhoria da eficiência na de construção de processos industriais, o trabalho proposto em [Abele et al. 2013] apresenta uma abordagem para a validação automática de modelos de plantas CAEX (C*omputer Aided câmbio Engineering*) por sua transformação em ontologias e posterior aplicação de raciocínio para fins de validação, através da web semântica.

Na área de diagnóstico, [Kupcik et al. 2012] apresenta um trabalho bastante focado na interoperabilidade de aplicações. Para isso, desenvolveu um mecanismo de interoperabilidade baseado em ontologias. Ele ilustra vários cenários envolvendo ontologias para os sistemas que compartilham conjuntos de dados de diagnóstico ou termos significados de sistemas dinâmicos, fazendo com que haja uma melhor compreensão e manipulação entre eles.

Alguns trabalhos também estão sendo desenvolvidos na área de confiabilidade de processos indutriais. Como por exemplo, [Olawande Daramola & Biffl 2011] apresenta o uso de ontologias na modelagem dos conceitos envolvidos em HAZOP (*Hazard and Operability*) e FMEA (Failure Mode and Effect Analysis), objetivando diminuir o tempo e esforço associados a essas análises. [Li 2012] apresentou um trabalho resumido também aplicado à análise de modos de falhas com FMEA.

Os trabalhos encontrados na literatura, em geral, buscam resolver problemas específicos na área de processos industriais e não desenvolver uma abordagem mais genérica. Nesta tese, desenvolveu-se uma ontologia de domínio para a área de processos industriais, também utilizada por outras ontologias de aplicações. Com as ontologias desenvolvidas, permite-se desenvolver aplicações nas mais diversas áreas, como gerência de alarmes,

projetos, confiabilidade.

Capítulo 3

Ontologia Aplicada ao Domínio de Processos Industriais

Nessa seção será apresentado o núcleo da ontologia desenvolvida, que objetiva modelar estruturas de plantas de processos industriais, a OntoAuto. O foco dessa ontologia é permitir o desenvolvimento de diversas aplicações na área da automação industrial, como detecção de falhas, diagnóstico de anomalias, monitoramento inteligente de processos, avaliação econômica, gerência de alarmes, confiabilidade, etc, utilizando conhecimento estruturado. Inicialmente, foi dada especial atenção à caracterização dos componentes que compõem as plantas industriais e dos fluxos de energias que atravessam esses processos.

3.1 OntoAuto - Ontologia Aplicada a Processos Industriais

Os principais conceitos (classes) definidos na ontologia (OntoAuto) criada estão ilustrados na Figura 3.1 e descritos abaixo:

- *IndustrialProcess*: representa o conceito de um processo industrial;
- *Component*: consiste nos elementos presentes na representação das plantas industriais, como equipamentos, tubulações, instrumentos, alarmes, operadores humanos, entre outros;
- *RawMaterial*: entidade utilizada para representar as possíveis matérias primas utilizadas nos processos industriais;
- *Utility*: entidade utilizada para representar as possíveis utilidades usadas nos processos industriais, como por exemplo, vapor e água.

Como pode ser visto na Figura 3.1, para a classe “Component” foram criadas algumas subclasses:

- *Equipment*: consiste nos elementos que podem fazer alguma transformação física ou química nos fluidos que circulam pela planta, como por exemplo: torres, vasos, tanques, forno, bombas, etc;

- *Pipe*: são condutos fechados destinados ao transporte de fluidos. As tubulações são constituídas de tubos de tamanhos padronizados, colocados em série;
- *Instrument*: consiste em elementos de automação e instrumentação que podem estar conectados a equipamentos da planta, como por exemplo: atuadores, controladores, sensores e transmissores;
- *Alarm*: sinalização visual e/ou sonora que identificam a ocorrência de um problema que deve ser percebido com rapidez;
- *Acessory*: entidade que representa qualquer outro componente que não seja um equipamento, uma tubulação, um instrumento ou um alarme;
- *ExternalEvent*: representam fontes importantes de problemas no sistema de automação. Não são de fato componentes físicos das plantas industriais, mas foram modelados como componentes. Por exemplo: fogo, chuva;
- *OperatorHuman*: representa a entidade de um operador humano do processo.

A ontologia foi desenvolvida utilizando uma abordagem top-down. Na Figura 3.1 a classe "Thing" se caracteriza como a classe raiz da hierarquia. As classes "Instrument", "Alarm", "Equipment", "Pipe", "Acessory", "ExternalEvent", "OperatorHuman" foram agrupadas em subclasses da classe "Component", pois ambas tem algumas características em comum e serão descritas mais adiante.

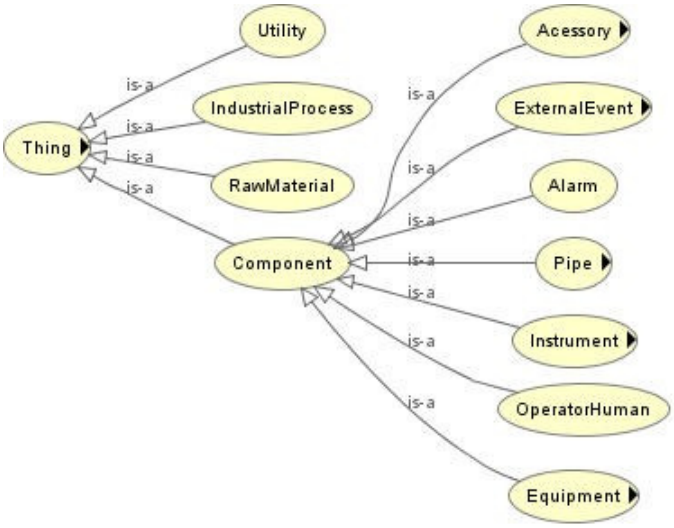


Figura 3.1: Principais classes da ontologia.

Para as classes "ExternalEvent" e "Acessory" foram criadas as subclasses como mostrado nas Figuras 3.2 e 3.3, respectivamente.

Em relação às tubulações, basicamente são constituídas de tubos que se destinam ao transporte de fluidos e troca de calor. Enquadram-se nessa categoria, por exemplo, os dutos que apenas transportam materiais, as serpentinas que permitem a troca de calor, as chaminés de caldeiras e fornos industriais. Com isso, para a classe "Pipe" foram criadas as subclasses presentes na Figura 3.4.

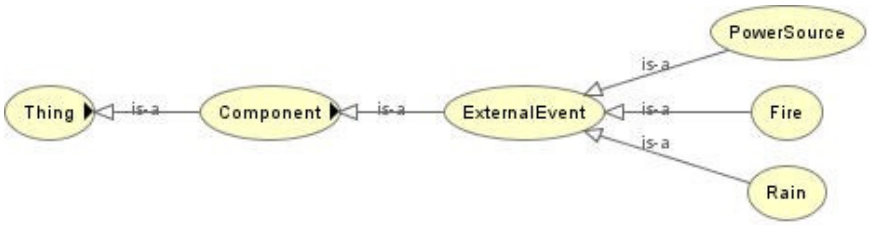


Figura 3.2: Subclasses da classe “ExternalEvent”.

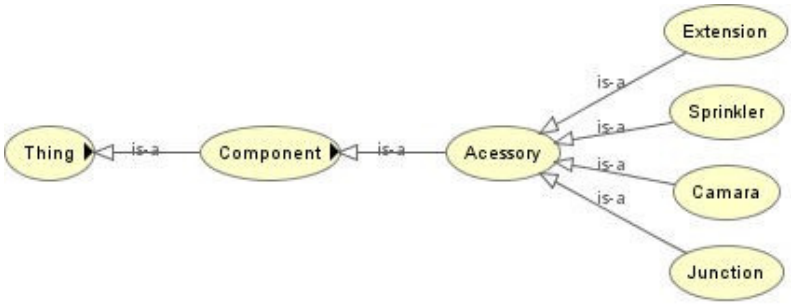


Figura 3.3: Subclasses da classe “Acessory”.

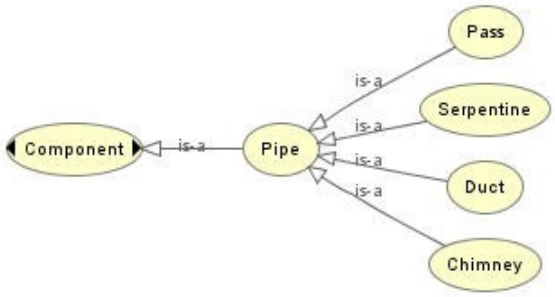


Figura 3.4: Subclasses da classe “Pipe”.

Em relação aos equipamentos e instrumentos, existe uma grande variedade deles que podem estar presentes em uma planta industrial. Dessa forma, também houve a necessidade de criar subclasses para as classes “Equipment” e “Instrument”, de modo a refinar a abstração, como pode ser visto nas Figuras 3.6 e 3.5, respectivamente. Nessas figuras estão apenas algumas das subclasses usadas para exemplificar. Vale salientar, que se, por exemplo, novos equipamentos ainda não modelados forem surgindo, basta acrescentá-los à ontologia.

Da forma que se propõe esta modelagem, permite-se que a ontologia possa ser reaproveitada e refinada caso haja a necessidade de novas especializações dessas classes.

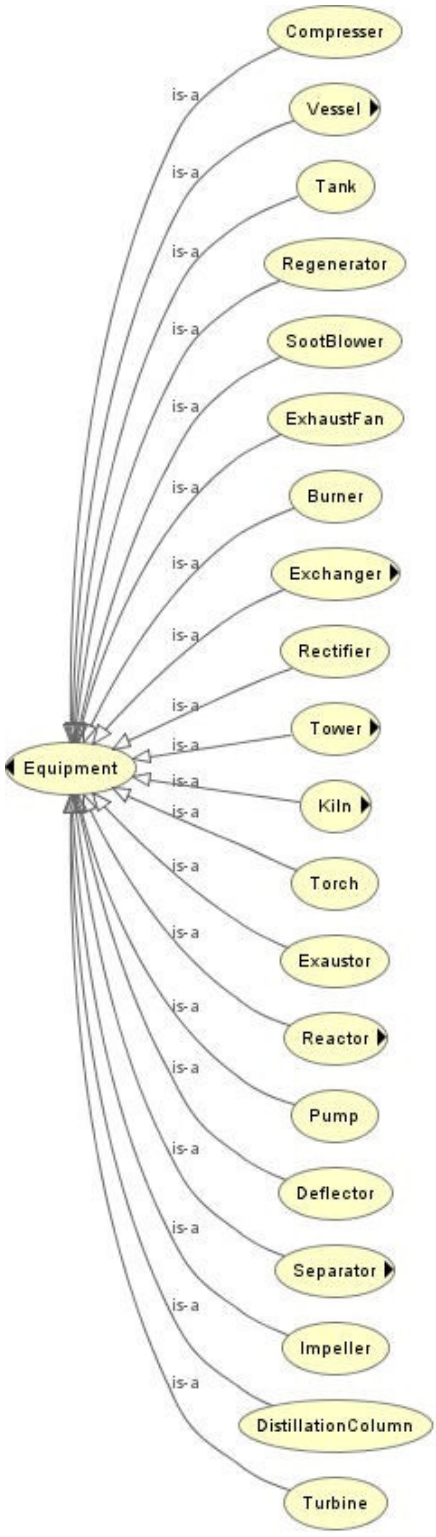


Figura 3.5: Subclasses da classe “Equipment”.

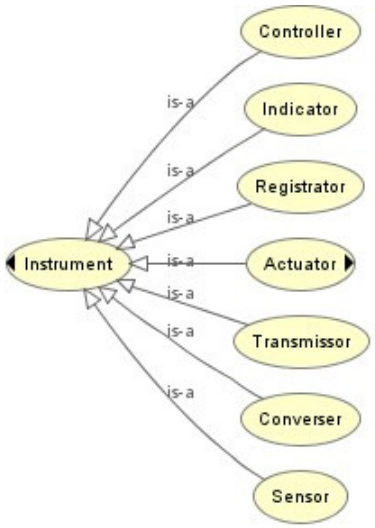


Figura 3.6: Subclasses da classe “Instrument”.

3.1.1 Principais Propriedades da OntoAuto

O próximo passo foi definir quais propriedades (características e relações entre indivíduos) indivíduos destas classes necessitariam ter. Assim, constatou-se que uma característica importante seria um identificador para cada componente da planta. Por exemplo, os equipamentos e instrumentos possuem tagnames (tag), tubulações possuem diâmetros nominais. Assim, a classe “Component” foi associada à uma propriedade do tipo “String” e nome “hasIdentifier”. Como em instrumentação industrial os instrumentos e equipamentos possuem tags, então foi necessário criar para estas categorias de componentes uma propriedade denominada de “hasTag”, que é subpropriedade da propriedade “hasIdentifier”. Além disso, incluiu-se uma propriedade a mais denominada de “hasDescription” para permitir descrição textual de vários dos elementos da planta industrial. Grande parte das propriedades definidas para a OntoAuto estão descritas nas Tabelas 3.1 e 3.2.

Tabela 3.1: Propriedades da OntoAuto - *Data Type Properties*.

Propriedade	Domínio	Tipo de Dado	Descrição
hasIdentifier	Component	string	Armazena o valor do identificador do componente.
hasTag	Instrument, Equipment, Alarm	string	Armazena o valor da TAG do instrumento, equipamento ou alarme. Herda da propriedade “hasIdentifier”

hasDescription	Component, RawMaterial, Utility, IndustrialProcess	string	Representa um descrição textual do elemento de domínio a que está associado. Por exemplo, uma descrição do processo industrial, da matéria prima, etc.
hasOrder	Instrument, Serpentine	int	Usada quando for necessário identificar a ordem em que determinado instrumento ou serpentina aparece no componente em que se encontra. Por exemplo, uma câmara de um forno pode ter várias serpentinas arranjadas sequencialmente e pode ser interessante identificar a ordenação delas. O mesmo pode acontecer em sensores ordenados dentro de uma tubulação.

Tabela 3.2: Propriedades da OntoAuto - *Object Properties*.

Propriedade	Domínio	Tipo de Dado	Descrição
hasIndustrialProcess	Component	IndustrialProcess	Associa um determinado indivídulo de Component ao indíviduo do processo industrial correspondente.
hasAlarm	Sensor, Controller	Alarme	Representa o conjunto de alarmes associados ao instrumento (Ex.: sensor, controlador). Propriedade inversa: isAlarme.
isAlarme	Alarme	Sensor, Controller	Indica a que instrumento (Ex.: controlador, sensor) o alarme está associado. Propriedade inversa: hasAlarme.
hasController	Actuator	Controller	Representa o controlador associado ao atuador (Ex.: válvula atuadora). Propriedade inversa: isController.

isController	Controller	Actuator	Indica a qual atuador está associado a determinado controlador. Propriedade inversa: hasController.
hasInstruments	Pipe, Equipment	Instrument	Representa o conjunto de instrumentos associados a uma tubulação ou equipamento. Propriedade inversa: isInstrument.
isInstrument	Instrument	Pipe, Equipment	Indica a que tubulação ou equipamento o instrumento está associado. Propriedade inversa: hasInstruments.
sendSignal	Sensor	Controller	Representa o controlador para o qual o sensor envia informação. Propriedade inversa: receiveSignal
receiveSignal	Controller	Sensor	Representa de que sensor o controlador recebe informação. Propriedade inversa: sendSignal.
hasComponents	Equipment	Component	Utilizado para representar quando determinado equipamento é formado por vários componentes. Por exemplo, um forno é formado por vários outros equipamentos, como: passos, serpentinas, câmaras, chaminé, coifa, etc.
isComponent	Component	Equipmment	Identifica de que Equipamento o componente faz parte.

3.1.2 Modelando o Fluxo de Direção da Planta Industrial

Um dos aspectos mais importantes da ontologia proposta é modelar o fluxo que a informação, material, energia seguem ao longo da planta industrial. Além de modelar também a conexão entre os componentes da própria planta. Observe na Figura 3.7 como os componentes da planta estão conectados entre si. O “Heater” tem duas entradas, uma rotulada com o valor 1 com origem desconhecida e outra com o valor 2 com origem sendo a saída do “Controller”. Já o “Sensor” possui uma única entrada, rotulada com o valor 1

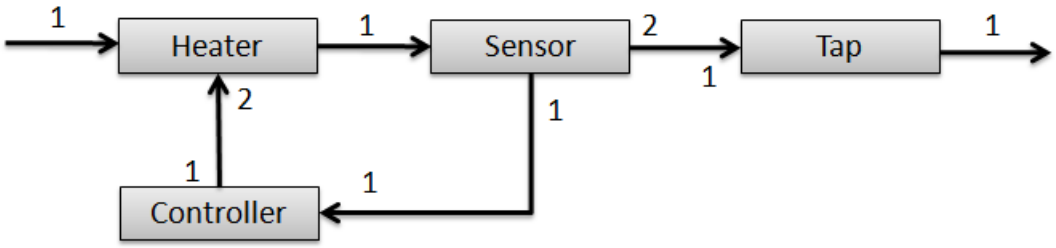


Figura 3.7: Modelando o fluxo de direção da planta industrial.

e duas saídas, a rotulada com valor 1 corresponde à entrada do “Controller” e a rotulada com valor 2 é a entrada do componente “Tap”.

Na OntoAuto, foi necessário modelar essa relação entre entradas e saídas e também o tipo do fluxo que passa entre eles. Para isso, foram criadas duas novas classes para a ontologia “Input” e “Output”, conforme mostrado na Figura 3.8. Também foi criada a classe “FlowType” para representar o tipo do fluxo que pode se propagar ao longo do sistema através de seus componentes, por exemplo, material, informação, energia ou comando.

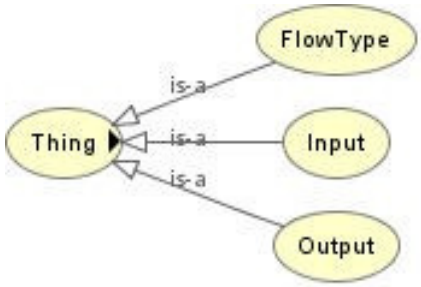


Figura 3.8: Classes para modelagem de entradas, saídas e tipos de fluxos entre os componentes da planta.

No exemplo da Figura 3.7, ao modelá-lo com a OntoAuto, pode-se dizer que existe para o equipamento “Heater”, dois indivíduos da classe “Input”, representando pelas entradas 1 e 2 e um indivíduo da classe “Output”, representado pela saída 1. O instrumento “Sensor” possui associação com um indivíduo da clase “Input”, representado pela entrada 1 e dois indivíduos da classe “Output”, representaods pelas saídas 1 e 2. Além disso, observa-se que a entrada 1 do “Sensor” corresponde à saída 1 do “Heater”, assim como a saída 1 do “Sensor” corresponde à entrada 1 do “Controller” e a saída 2 do “Sensor” corresponde à entrada 1 do “Tap”.

A modelagem do fluxo de direção da planta é realizada efetivamente através de algumas outras propriedades presentes na OntoAuto. Essas propriedades permitirão rotular as entradas e saídas e fazer a associação entre elas e com os seus respectivos componentes. Essas propriedades estão descritas na Tabela 3.3.

Tabela 3.3: Propriedades da OntoAuto - Modelagem do Fluxo de Direção.

Propriedade	Domínio	Tipo de Dado	Descrição
hasValue	Input, Output	string	Valor textual para rotular as entradas e saídas dos componentes.
hasComponentInputOutput	Input, Output	Component	Associa a entrada e/ou saída ao componente relacionado.
hasAssociateInput	Output	Input	Indica qual entrada é associada a determinada saída.
hasAssociateOutput	Input	Output	Indica qual saída é associada a determinada entrada.
isFlowType	Input	FlowType	Indica o tipo de fluxo que entra no componente: material, informação, energia ou comando.

3.2 Considerações Finais

Neste capítulo foi apresentada a OntoAuto, ontologia desenvolvida para representar o domínio associado a processos industriais. A ontologia foi desenvolvida de forma incremental, ou seja, à medida que o domínio envolvido foi sendo explorado, novos conceitos e novas propriedades foram sendo inseridas na ontologia. Um dos objetivos é utilizar essa ontologia como base para o desenvolvimento de aplicações de automação avançada, como por exemplo, na área de gerência de alarmes, avaliação econômica e confiabilidade.

À medida que as aplicações que utilizam essa ontologia foram sendo desenvolvidas, percebeu-se a necessidade de complementá-la com novos conceitos e novas propriedades. Diante desse contexto, na OntoAuto só foram acrescidas novas informações que fossem referente ao núcleo da representação do conhecimento de processos industriais. Para novos conceitos e novas propriedades específicas de uma determinada área em que uma aplicação precisava ser desenvolvida, novas ontologias foram criadas importando o conhecimento já representado na OntoAuto.

Capítulo 4

Aplicação 01: Modelando Processos com a OntoAuto

Com o objetivo de validar a capacidade da OntoAuto em modelar semanticamente processos industriais, neste capítulo são apresentadas instanciações da modelagem de dois processos industriais típicos: Unidade de Tratamento DEA e Forno Industrial.

4.1 Unidade de Tratamento DEA

De uma refinaria de petróleo é possível produzir diversos produtos, como por exemplo: óleo diesel, gasolina, nafta, querosene, coque, asfalto, enxofre, etc. Uma refinaria é formada por diversos arranjos de unidades de processamento em que são compatibilizadas as características dos vários tipos de petróleo que nela são processados, com o objetivo de suprir derivados em quantidade e qualidade especificadas [Arnold & Stewart 1999], [Meisen & Kennard 1982].

Os processos de refino são dinâmicos e sua sequência é estabelecida de forma que um ou mais fluídos de entrada do processo são transformados em outros fluídos que constituem a saída do processo. Essa transformação é realizada por algum processamento nas unidades de refino.

Os processos de refino podem ser classificados em quatro grupos: separação, conversão, tratamento e auxiliares. Um processo de tratamento, um dos focos deste trabalho, tem como objetivo melhorar a qualidade dos produtos através da redução de impurezas, como composto de enxofre e nitrogênio, sem causar profundas modificações das frações. Uma unidade de tratamento DEA é um exemplo desse tipo de processo.

Uma unidade produtiva pode possuir um sistema de tratamento de dietilamina (DEA) que é uma planta industrial que permite o tratamento dos gases gerados nas unidades de processo de gás liquefeito de petróleo (GLP) e gás combustível (GC), removendo principalmente H2S e CO2.

O tratamento DEA é um processo específico para remoção de H2S de frações gasosas do petróleo, especialmente aquelas provenientes de unidades de craqueamento. Ele também remove CO2 eventualmente encontrado na corrente gasosa.

O esquema da unidade de tratamento DEA, em geral, pode ser descrito conforme a Figura 4.1. O GLP a ser tratado entra pela torre extratora, onde entra em contato com a

DEA pobre (solução aquosa de DEA regenerada, ou seja, DEA que não contém H2S). A DEA pobre é admitida na parte superior da torre e a DEA rica (DEA com H2S) coletada na parte inferior é encaminhada para o sistema de regeneração. Na parte de cima da torre extratora, o GLP é separado da fase aquosa e segue para um tambor de decantação, onde os traços de DEA arrastados são removidos.

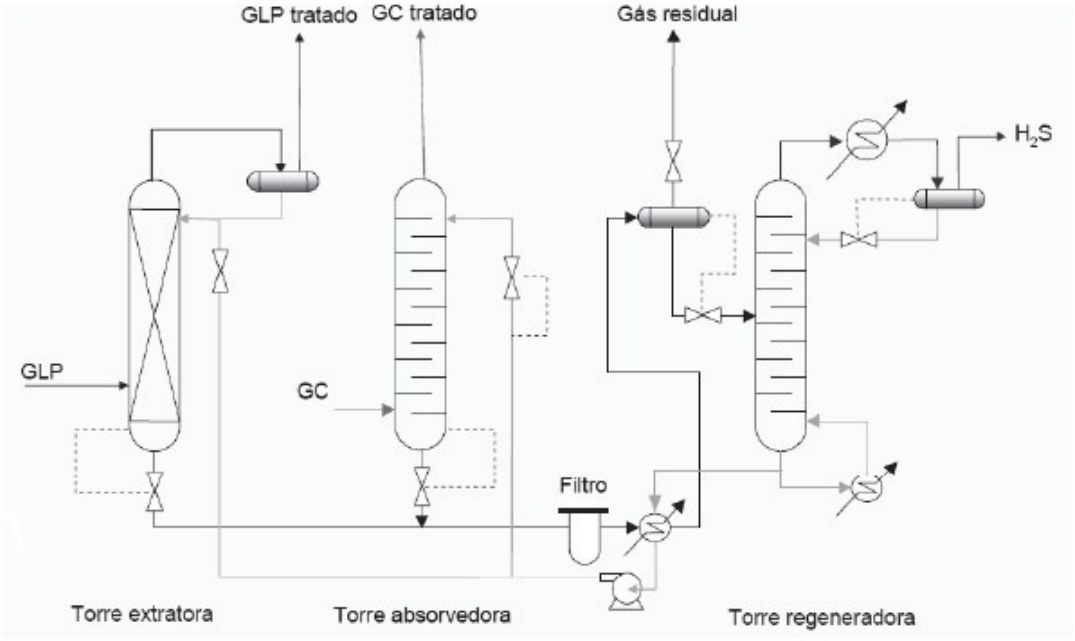


Figura 4.1: Esquema da Unidade de Tratamento DEA.

O gás combustível a ser tratado é encaminhado para a torre de absorção onde entra em contato com a solução DEA. Semelhante ao que acontece na torre extratora, a DEA pobre entra pela parte superior e a DEA rica sai pela parte inferior, sendo encaminhada para o sistema de regeneração após ter sido unida com a DEA rica vinda do sistema de extração. O GC tratado, retirado do topo da torre, é enviado para o sistema de gás combustível.

A torre regeneradora de DEA promove regeneração da corrente de DEA rica, ou seja, o H2S absorvido pela DEA é liberado, através de calor.

No caso da unidade de tratamento DEA utilizada para análise, não há a seção de extração, ou seja, apenas existem as torres de absorção e regeneração. Além disso, não há remoção de CO2. Outras mudanças podem ser verificadas nos fluxogramas de tubulação e instrumentação da área de tratamento DEA (absorção e regeneração) utilizados como referência.

Iniciando a análise com base na estrutura de classes da ontologia, em termos de equipamento, basicamente a planta real da unidade de tratamento DEA utilizada é formada por:

- Duas torres: uma para absorção de H2S e outra para regeneração da DEA rica;
- Duas bombas: uma para circulação da solução DEA e outra para o refluxo da torre regeneradora;

- Quatro vasos: um coletor de hidrocarboneto líquido, um coletor de DEA arrastada, um separador de hidrocarboneto e um ao topo da torre regeneradora;
- Quatro sistemas de troca de calor: dois permutadores, um condensador ao topo da torre regeneradora e um resfriador de DEA pobre.
- Um sistema para filtragem para fluido de processo.

Com isso, o primeiro passo foi criar uma instância da ontologia para a DEA. Em seguida, foram criados indivíduos para cada equipamento presente na planta real. Além dos indivíduos que representam os equipamentos, outros também foram criados, como por exemplo, alarmes, dutos e instrumentos. No caso, foram instanciados:

- 41 alarmes.
- 51 dutos.
- 38 sensores.
- 10 sensores chave.
- 7 controladores.
- 8 válvulas de controle.

Para a definição das características dos indivíduos e de como esses equipamentos, instrumentos, dutos e alarmes se relacionam entre si, para cada indivíduo foram definidos os valores para as propriedades da OntoAuto descritas anteriormente. A Figura 4.2 apresenta graficamente, no contexto da instância DEA modelada na ontologia, todos os indivíduos que representam equipamentos, sendo que os componentes sinalizados por um círculo representam classe e os sinalizados por losangos representam indivíduos.

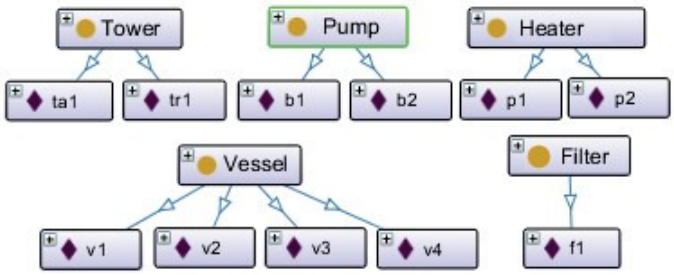


Figura 4.2: Indivíduos das classe “Equipment” para a DEA.

Na Figura 4.3, é possível visualizar como os indivíduos se relacionam entre si através das propriedades definidas pela ontologia. No caso, ct1, cpe3, cpe1, cpe2, cn1, cn2 e cn3 são indivíduos da classe “Controller”; a33, a34, a35 e a36 são indivíduos da classe “Alarm”; e spe2 é indivíduo da classe “Sensor”. As colorações das setas pontilhadas indicam como os indivíduos se relacionam. A seta azul contínua indica quais indivíduos estão associados à classe “Controller”. A Figura 4.4 descreve as propriedades representadas por cada tipo de seta pontilhada.

Se o fluxograma de tubulações e instrumentação da unidade de tratamento DEA utilizada como estudo de caso for comparado à Figura 4.5, verifica-se que o indivíduo d01, pertencente à classe “Pipe”, representa o duto de entrada da unidade de tratamento DEA

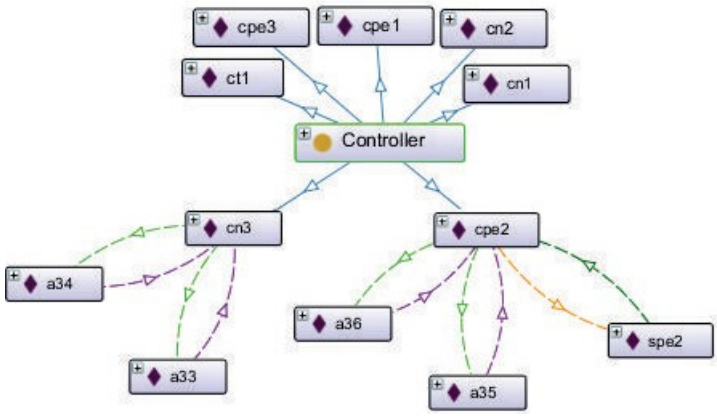


Figura 4.3: Visualização de relacionamento entre indivíduos da DEA.

Seta	Propriedade	Significado
	isAlarm	Indivíduo a esquerda da seta é um alarme associado ao indivíduo a direita.
	hasAlarm	Indivíduo a esquerda da seta possui os alarmes representados pelos indivíduos da direita.
	receiveSignal	Indivíduo a esquerda da seta recebe sinal do indivíduo a direita.
	sendSignal	Indivíduo a esquerda da seta envia sinal para o indivíduo a direita.

Figura 4.4: Propriedades exibidas na Figura 4.3.

de identificador 8”-HC-98DA-316-Bo. A montante desse duto existe o vaso V-95517, representado pelo indivíduo v1; a montante de v1, há um outro duto de identificador 8”-HC-980A-714-Bo, representado por d02, e assim por diante. Subsequente a d02, existe a torre de absorção T-98507 (ta1). Observa-se que ela possui dois dutos de saída, d03 (6”-DA-980A-651-Bo) e d06 (8”-HC-980A-718-Bc), e dois dutos de chegada, d02 e d09 (6”-DA-980A-684-Cc), ou seja, a ontologia proposta também modela a situação em que um mesmo componente da planta industrial pode ter mais de uma entrada ou mais de uma saída.

A Figura 4.5 apresenta alguns indivíduos das subclasses da classe “Component” da instância da DEA. Para cada um desses componentes, foi necessário criar X instâncias da classe “Input”, onde X é a quantidade de entradas do componente, e Y instâncias da classe “Output”, onde Y é a quantidade de saídas do componente. Para analisar melhor, considere a Figura 4.6, o indivíduo d02 tem uma saída associada, out_d02 (indivíduo da classe “Output”). Por sua vez, out_d02 está associado ao indivíduo in_ta1 (indivíduo da classe “Input”) pela propriedade “hasAssociateInput”. O indivíduo in_ta1 está associado ao componente ta1 pela propriedade “componentInputOutput” e à out_d02 pela propriedade “hasAssociateOutput”. Além disso, o componente ta1 possui três saídas associadas (out1_ta1, out2_ta1, out3_ta1), ambas indivíduos da classe “Output”. As saídas out1_ta1,

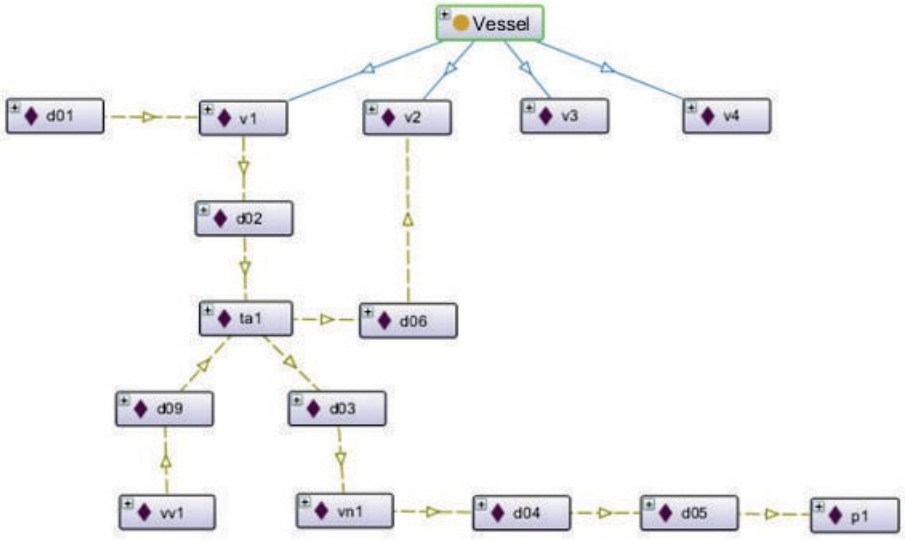


Figura 4.5: Representação do sentido do fluxo do processo.

out2_ta1, out3_ta1 são associadas, respectivamente, às entradas dos dutos d03, d06, d09. Cada entrada modelada nesse trecho de fluxo da Figura 4.5 foi associada ao tipo de fluxo de material (indivíduo da classe "FlowType").

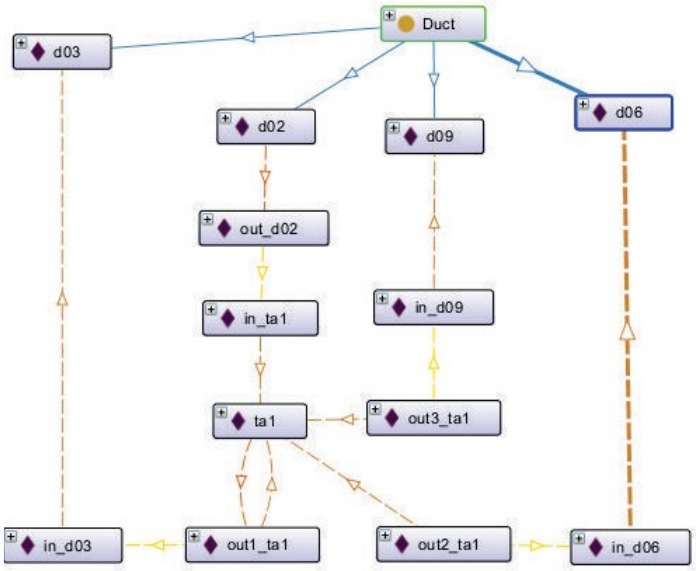


Figura 4.6: Representação de entradas e saídas.

O principal objetivo de realizar modelagens com ontologias é poder extrair informações semânticas, porém também é possível extrair informações quantitativas, assim como metodologias tradicionais de modelagem como diagramas de entidades e relacionamentos. A Tabela 4.1 apresenta um exemplo de extração de informação quantitativa sobre a

ontologia da DEA. No caso, são indicados os quantitativos de sensores e alarmes existentes na planta.

Tabela 4.1: Extração de informação quantitativa da DEA.

Indicador	Propriedade
Sensor de Temperatura	10
Sensor de Nível	4
Sensor de Pressão	7
Sensor de Vazão	3
Alarme H	23
Alarme L	18

4.2 Forno Industrial

Os fornos industriais são equipamentos bastante importantes nas refinarias e indústria petroquímica, visto que a utilização de chama oriunda da queima de combustível é uma ótima maneira de fornecer grande quantidade de energia para elevar grandes vazões de fluidos e altas temperaturas, viabilizando as operações de destilação, craqueamento, etc. A principal finalidade de um forno é fornecer o calor produzido pela queima de combustível ao fluido que circula em uma serpentina de tubos em seu interior.

O projeto de um forno pode ser feito de várias formas, ou seja, há uma grande variação na concepção de projetos de fornos. Em geral, as principais partes de um forno são [Moore & Moore 1943], [Dawe 2002]: câmara de combustão (onde há a queima de combustível); seção de radiação (semelhante à câmara de combustão, onde os tubos são diretamente expostos à radiação da chama); seção de convecção (os tubos não são expostos diretamente à radiação da chama. Eles entram em contato com os gases quentes vindos da câmara de combustão); serpentina (conjunto de tubos consecutivos através dos quais o fluido passa dentro do forno nas seções de radiação e convecção); chaminé (montada acima da câmara de combustão, é responsável pela tiragem e descarga dos gases). A Figura 4.7 apresenta um exemplo de forno industrial, constituído de duas câmaras de combustão.

Os fornos precisam de sistemas de alimentação compostos pelos sistemas de distribuição (anéis), de gás combustível, óleo combustível, de vapor de atomização, além do anel de vapor de abafamento (para purga do forno) e dos dutos e sopradores de ar, no caso da tiragem forçada.

Alguns equipamentos auxiliares também fazem parte dos fornos, por exemplo, os queimadores (responsáveis pela liberação de calor necessário para as diversas serpentinas de um forno, através da combustão de um ou mais combustíveis) e os sopradores de fuligem (equipamentos com a finalidade de realizar a limpeza periódica das serpentinas onde há acúmulo de fuligem devido à queima de óleos combustíveis).

Basicamente, nas refinarias e plantas petroquímicas são utilizadas duas categorias de

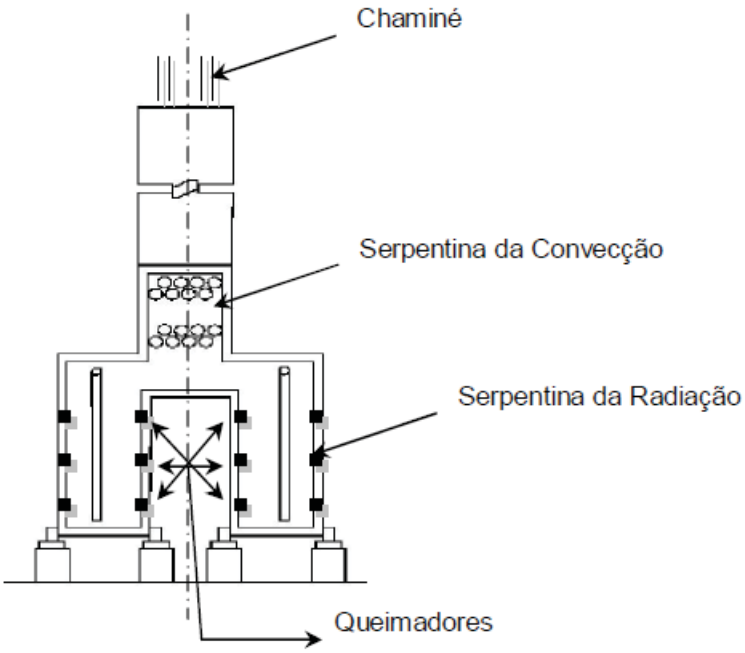


Figura 4.7: Exemplo ilustrativo de um forno industrial.

fornos:

- Fornos de Aquecimento: correspondem a todos os fornos em que a carga, ao passar pela serpentina de processo, sofre apenas um aquecimento, podendo ou não ser vaporizada.
- Fornos Reatores: correspondem a todos os fornos em que a carga, ao passar pela serpentina de processo, além de sofrer um aquecimento tem sua composição química alterada.

Com base na mesma ontologia OntoAuto apresentada anteriormente, uma nova instância foi criada para um forno industrial. A planta modelada possui dois fornos, onde um é uma redundância do outro. Cada forno formado por uma chaminé, duas câmaras de combustão, em que cada câmara possui duas serpentinas. Cada uma das serpentinas das duas câmaras é alimentada por um dos quatro passos de entrada do forno. Além disso, cada câmara possui o seu sistema de combustão.

A Figura 4.8 apresenta parte da instância da ontologia para os fornos. É possível observar que há dois indivíduos da classe “Kiln”: forno001 e forno002. Nesta mesma figura, há um detalhamento maior para o indivíduo do forno001 (semelhante a do forno002), onde é possível visualizar que ele se relaciona com um indivíduo da classe “Chimney” (cham001), quatro indivíduos da classe “Pass” (passo1A, passo2A, passo3A e passo4A), com dois indivíduos da classe “Camara” (cam001 e cam002) e que cada indivíduo “Camara” se relaciona com dois indivíduos da classe “Serpentine” (serp001, serp002, serp003 e serp004).

A Figura 4.9 descreve as propriedades ilustradas na Figura 4.8.

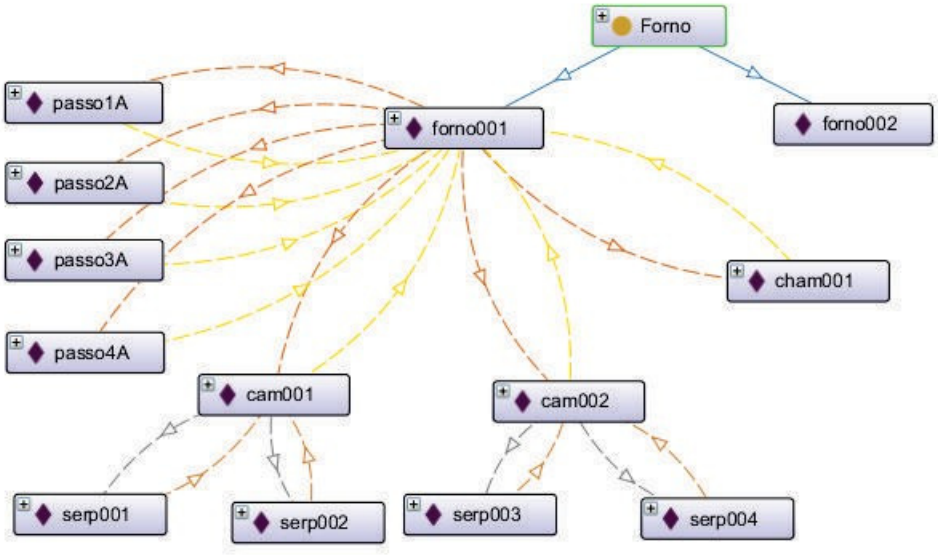


Figura 4.8: Exemplo da instância do forno industrial.

Seta	Propriedade	Significado
	hasComponents	Indivíduo a esquerda da seta é formada pelos indivíduos (componentes) da direita.
	isComponent	Indivíduo a esquerda da seta compõe o indivíduo da direita.
	hasInstruments	Indivíduo a esquerda da seta possui como instrumento o indivíduo da direita.

Figura 4.9: Propriedades exibidas na Figura 4.8

Na Figura 4.10 é possível ver que uma das serpentinas (serp001) de uma das câmaras do forno (cam001) se relacionam com 18 sensores de temperatura, que estão conectados ao longo da sua extensão através da propriedade “hasInstruments”.

Outra questão a ser ressaltada é que esses sensores estão conectados à serpentina em determinada ordem. Assim, através da propriedade “hasOrder” é possível representar esta ordenação física dos sensores ao longo da serpentina, por exemplo, o indivíduo sensor stemp001 possui valor “1” para a propriedade “hasOrder”, o sensor stemp002 possui o valor “2”, o sensor “stemp003” possui o valor “3” para esta mesma propriedade e assim

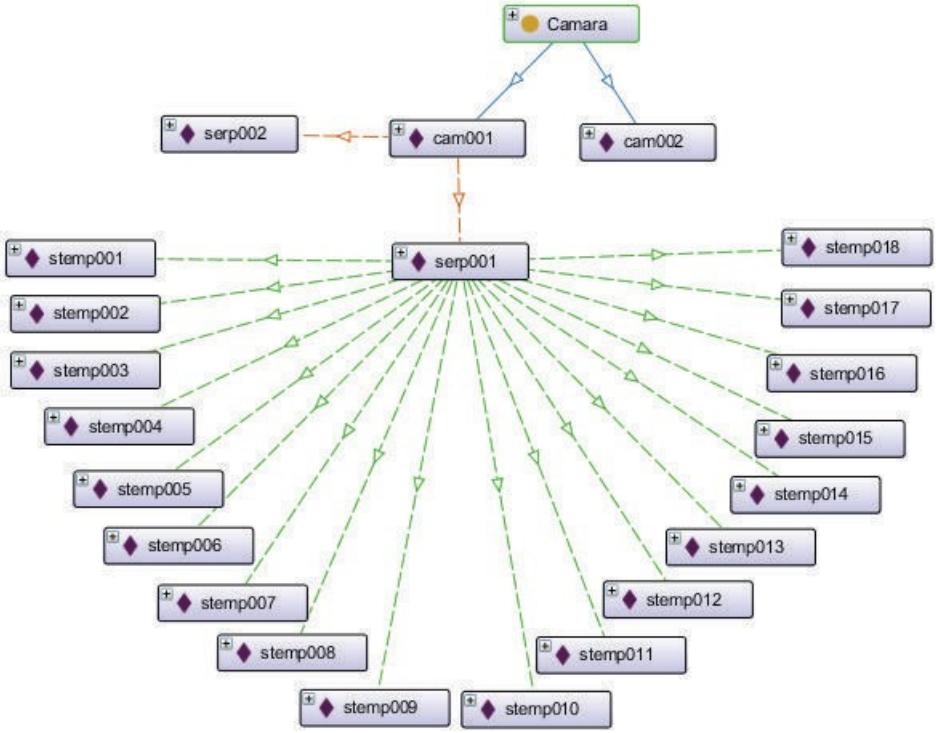


Figura 4.10: Exemplo dos indivíduos sensores de temperatura associados às serpentinas das câmaras de combustão.

por diante. Isso significa que através da instância da ontologia desenvolvida é possível associar os sensores à serpentina e em que ordem eles estão conectados ao longo dela, permitindo-se via inferências acompanhar o perfil de distribuição de temperatura ao longo da serpentina, desde o seu início (entrada da câmara) até seu término (saída da câmara).

A Figura 4.11 apresenta a relação entre os queimadores e uma das câmaras do forno, ou seja, apesar de não ter uma conexão física através de tubulação, por exemplo, a ontologia permite modelar a passagem do fluxo de energia dos queimadores para a câmara, possibilitando assim o mapeamento do fluxo de fluidos entre eles. Isso é possível porque a modelagem proposta para a ontologia permite que sejam representadas entradas (indivíduos da classe "Input") dos queimadores para as câmaras do forno associadas ao tipo de fluxo energia (propriedade "isFlowType"). A Figura 4.12 descreve as propriedades ilustradas na Figura 4.11.

Além de permitir modelar o fluxo do fluido e a passagem de energia ao longo da planta, a ontologia permite a extração de informações quantitativas. Como exemplo, a Tabela 4.2 apresenta um exemplo de extração de informação quantitativa sobre a instância da ontologia para o exemplo do forno. No caso, são indicados os quantitativos de sensores e alarmes existentes na planta, permitindo categorizar o seu tipo.

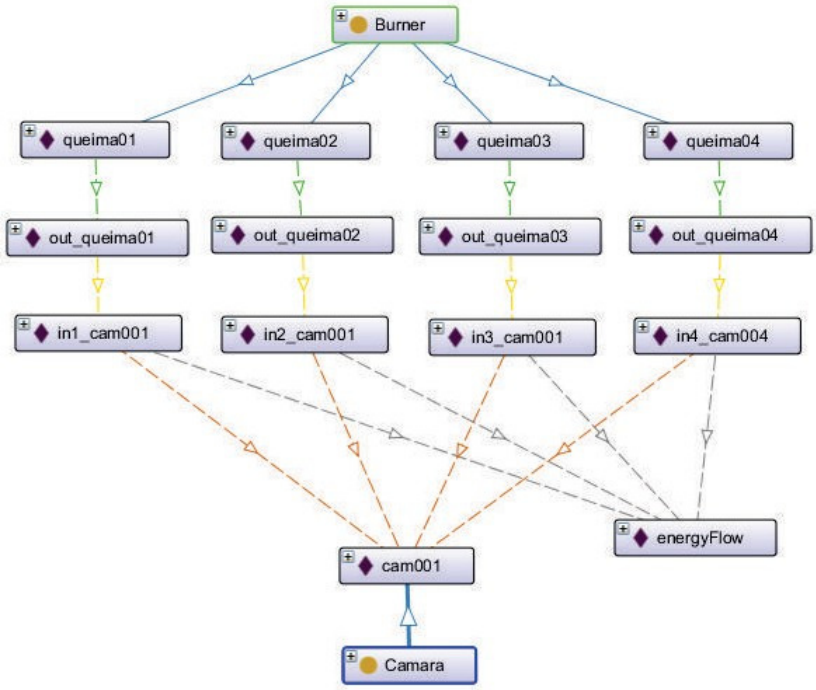


Figura 4.11: Exemplo de associação entre entradas e saídas com tipo de fluxo de energia.

Tabela 4.2: Extração de informação quantitativa do Forno.

Indicador	Propriedade
Sensor de Temperatura	87
Sensor de Nível	0
Sensor de Pressão	8
Sensor de Vazão	12
Alarme H	18
Alarme L	6

4.3 Considerações Finais

Neste capítulo foram apresentados dois processos importantes da área de automação industrial, Unidade de Tratamento DEA e Fornos Industriais, sendo modelados com a OntoAuto. Para ambos, observa-se que foi possível modelar como os seus componentes estão relacionados, como por exemplo quais equipamentos estão a jusante ou montante de outros, quais instrumentos estão associados a determinados equipamentos, entre outros. Além disso, permitindo ter uma definição de como o fluxo do fluido segue na planta.

No caso particular dos fornos, em algumas situações os componentes não são ligados fisicamente, mas possuem uma ligação com base no fornecimento de energia, como por exemplo, as chamas dos queimadores passam energia para aquecer as câmaras dos fornos. Além disso, no caso das serpentinas, é possível identificar a ordem em que os sensores de temperaturas estão instalados ao longo da mesma, permitindo, por exemplo, monitorar a variação de temperatura ao longo da mesma. Dessa forma, a OntoAuto também permitiu modelar esses comportamentos.

Seta	Propriedade	Significado
	hasOutput	Indivíduo da esquerda (Component) tem como saída o indivíduo da direita.
	hasAssociateInput	Indivíduo da esquerda (Output) está associado ao indivíduo da direita (Input).
	componentInputOutput	Indivíduo da esquerda (Input) está associado ao indivíduo da direita (Component)
	isFlowType	Indivíduo da esquerda (Input) está associado ao indivíduo da direita (FlowType)

Figura 4.12: Propriedades exibidas na Figura 4.11.

Capítulo 5

Aplicação 02: Correlação de Alarmes

Um alarme é uma notificação ao operador sobre a ocorrência de uma anormalidade que necessita de uma ação a ser tomada, mesmo que mental [Eng 1999]. Os sistemas de alarmes possuem a função de receber essas notificações e apresentá-las ao operador, porém em muitas situações é necessário, para a correta identificação de uma anormalidade, a interpretação de múltiplos alarmes.

Vários estudos estão sendo realizados para otimizar a carga de alarmes e eventos apresentados à operação. Alguns estudos elaboraram guias de boas práticas onde estabelecem limites para o número gerenciável de sinalizações de alarmes por operador, bem como definem estratégias de priorização em caso de avalanches de alarmes [Eng 1999]. Assim, técnicas como priorização, supressão e agrupamento de alarmes devem ser utilizadas pelos sistemas de alarmes [Pinto & Paula 2009].

Devido ao crescente número de alarmes apresentados aos operadores de plantas industriais, várias indústrias preocupadas com a segurança de seus processos estão em um árduo trabalho de minimização das ocorrências desses alarmes através de um estudo chamado de racionalização de alarmes [Gustavo Leitão & Aquinoo 2008]. Na atividade de racionalização, uma equipe multidisciplinar é selecionada para avaliar a necessidade de cada alarme configurado. O desafio é manter o sistema monitorado de forma que o número de ocorrências apresentadas ao operador seja condizente com a habilidade humana de processar e tratar informações [Eng 1999]. Essa equipe necessita de ferramentas para avaliar e encontrar alarmes configurados erroneamente.

Os sistemas de alarmes constituem um elemento fundamental em quase todos os modernos sistemas industriais, incluindo refinarias de petróleo, centrais elétricas, indústrias [Leitão 2008].

Conforme descrito na norma EMMUA 191, os sistemas de alarmes permitem o monitoramento automático da planta, atraindo a atenção do operador para mudanças significativas do processo que necessitam de avaliação ou ação. [Leitão 2008] apresenta algumas situação de como os alarmes auxiliam o operador:

- Manter a planta dentro de uma faixa segura de funcionamento. Um bom sistema de alarmes orienta ao operador sobre situações potencialmente perigosas antes que o ESS (*Emergency Shutdown System*) seja forçado a intervir. Isto melhora a avaliação da planta e ajuda a diminuir a demanda do ESS, aumentando a segurança.

- Reconhecer e agir para evitar situações perigosas. É o papel dos sistemas ESS intervir em uma situação perigosa, entretanto podem existir casos onde a planta desvie das suas condições de projeto para um estado onde o ESS não é capaz de agir eficientemente, como durante o start-up de uma planta.
- Identificar desvios de condições operacionais que poderia levar a perdas financeiras, tais como, produtos fora de especificação ou custo excessivamente alto.
- Compreender as complexas condições do processo. Alarmes podem ser uma importante ferramenta de diagnóstico e são uma das várias fontes de informação que um operador pode utilizar durante uma perturbação no processo.

Atualmente, os sistemas de gerenciamento de alarmes de vários processos industriais implementam algumas funcionalidades para análise e filtragem dos alarmes gerados. As principais funcionalidades desses sistemas são [Leitão 2008]:

- Contagem de alarmes: realiza a contagem de alarmes para encontrar os alarmes mais freqüentes. Segundo a norma EMMUA, em um típico processo industrial, em média os alarmes mais frequentes são responsáveis pela maior parte das ocorrência de alarmes. Estes alarmes são conhecidos como *Bad Actors*. Identificá-los é o papel deste tipo de análise, permitindo tomar possíveis ações.
- Contagem de duração de alarmes: realiza a contagem do tempo médio de duração de cada alarme. Esta análise permite identificar alarmes que ficaram ativo por períodos muito prolongados ou alarmes com tempo de atividade extremamente curto.
- Alarmes Recorrentes: esta análise tenta encontrar alarmes que se tornam ativos um número elevado de vezes em um curto período de tempo. É uma das maneiras de identificar alarmes com thresholds mal configurados e por esta razão entram e saem periodicamente em atividade.
- Correlação entre Alarmes: realiza a correlação estatística entre as ocorrências de alarmes, a fim de encontrar qual a influência de um alarme com relação aos demais.

Todas essas funcionalidades podem realizar suas atividades tanto com o processo *on-line* ou *offline*, objetivando identificar os alarmes problemáticos e permitindo que a equipe de gestão atue de forma a otimizar a análise dos alarmes. O gerenciamento de alarmes é uma atividade que deve ser realizada constantemente, a fim de manter os equipamentos de monitoramento da planta cada vez mais bem configurados, desta forma, reduzindo a curto e médio prazo, o aparecimento de possíveis problemas com impactos econômicos, pessoal e na segurança do processo [Habibi & Hollifield 2006].

Um fator bastante importante na identificação de alarmes é verificar o quanto as ocorrências de alarmes estão relacionadas. Isso significa que deve ser possível identificar, por exemplo, dado que um determinado alarme ocorreu quais outros alarmes estiveram relacionados com esta ocorrência, que impacto um alarme tem nos demais e se existem alarmes redundantes. Dessa forma, é essencial ter uma forma de identificar a relação entre os alarmes.

Para o cálculo desta análise se faz necessário escolher um alarme de referência, o qual deseja encontrar as relações com os demais e uma janela de tempo de ativação. Esta janela deve ser dimensionada por pessoas conhecedoras do processo, pois o valor da janela

consiste em quanto tempo após o alarme ter ocorrido será verificado a perpetuação dos seus efeitos no processo. O cálculo, no entanto, consiste em buscar todas as ativações do alarme escolhido e verificar quais outros alarmes aconteceram durante a janela de tempo fornecida. A janela, por sua vez, é posicionada a partir da ativação do alarme selecionado em diante. Uma vez identificados os alarmes que se ativaram dentro da janela, estes são contabilizados em relação ao total de ocorrências do alarme escolhido para a efetuação do cálculo da relação [Leitão 2008].

5.1 Estudo de Caso - Correlação Semântica de Alarmes

Nessa seção será apresentado um estudo de caso no tratamento avançado de alarmes com o objetivo de encontrar alarmes correlacionados temporalmente com a ajuda do conhecimento do fluxo do processo modelado via ontologia, auxiliando na busca por alarmes desnecessários. Os resultados foram coletados com base no processo industrial da DEA modelado a partir da OntoAuto.

A extração de conhecimento do fluxo do processo foi realizada através da utilização da instância da ontologia para a DEA, com o uso das ferramentas Jena e SPARQL. Para a aplicação de correlação semântica de alarmes, inicialmente, foi preciso ordenar os alarmes presentes na DEA. O algoritmo de ordenação é ilustrado através da Figura 5.1. Os retângulos representam alarmes que estão em algum instrumento, que podem estar conectados a dutos ou equipamentos. As setas sinalizam a direção de escoamento do fluido pelos dutos e equipamentos da planta DEA. As setas também representam equipamentos ou dutos que não possuem alarmes. Em termos de localização física, elas indicam que o equipamento ou duto está localizado fisicamente anterior ou após o duto/equipamento, cujo alarme está associado.

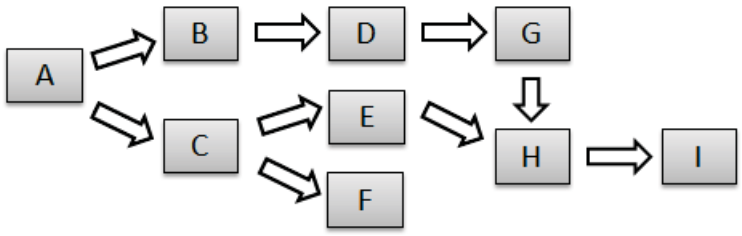


Figura 5.1: Ordenação física dos alarmes.

Dessa maneira, é possível ter mais de uma sequência de ordenação de causalidade para os alarmes, como por exemplo:

- Ordenação 1: A -> B -> D -> G -> H -> I
- Ordenação 2: A -> C -> E -> H -> I
- Ordenação 3: A -> C -> F

Com isso, é possível inferir que o alarme A está fisicamente antes de qualquer alarme na planta. Já o alarme B só está posicionado anterior aos alarmes D, G, H e I. O alarme C

está anterior ao E, F, H e I e assim por diante. Por outro lado, não é possível inferir que o alarme H está localizado antes ou depois do alarme F, pois em termos de sequência eles aparecem em filas de ordenação distintas.

Nessa lógica, é possível inferir um nível de vizinhança entre os alarmes, incluindo a distância entre eles. Por exemplo, para saber os vizinhos do alarme C, considerando-se apenas distâncias de uma unidade, o resultado seria: A, E e F. Por outro lado, se é desejado saber os vizinhos de D a uma distância de 2 unidades, teríamos: A, B, G e H.

O algoritmo de correlação de alarmes padrão tem como entrada o período de análise, um determinado alarme e a largura da janela de tempo. O cálculo busca todas as ativações do alarme escolhido e verifica quais outros alarmes ocorreram durante a janela de tempo fornecida. A janela é posicionada a partir de cada ativação do alarme selecionado. O resultado da correlação é dado pela razão entre o número de janelas que um determinado alarme ocorreu em relação ao total de ocorrências do alarme de referência.

Com a informação semântica adicional da ordenação dos alarmes e de sua vizinhança é possível filtrar do resultado da correlação de alarmes daqueles alarmes sem proximidade no processo, podendo inclusive informar o nível de vizinhança se assim for desejado.

Para os resultados, um período de tempo de um mês e janela de tempo de 10 minutos para cálculo das correlações foram utilizados. A Figura 5.2 apresenta o resultado obtido de correlação entre alarmes utilizando-se apenas o algoritmo básico. A Figura 5.3 apresenta um agrupamento de três filas de ordenação dos alarmes, através da instância da ontologia para a DEA.

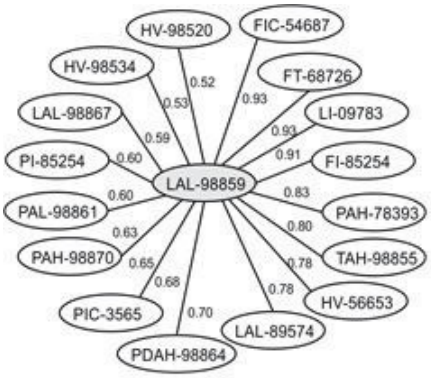


Figura 5.2: Correção de alarmes usando o algoritmo básico.

Agregando-se a informação semântica ao algorimo de correlação realizou-se uma filtragem dos resultados, retirando os alarmes que não tivessem relação de vizinhança com o alarme de referência. A Figura 5.4 mostra os resultados obtidos para o cenário descrito na Figura 5.2, porém considerando-se agora alarmes que possuem vizinhança de uma unidade com o alarme de referência. Já a Figura 5.5 apresenta o resultado considerando vizinhança de duas unidades com o alarme de referência.

Devido à grande quantidade de dados espúrios presentes nos logs de alarmes e eventos de um típico processo industrial complexo, ferramentas de filtragem através do conhecimento do fluxo do processo, tornam-se bastante relevantes como forma de excluir

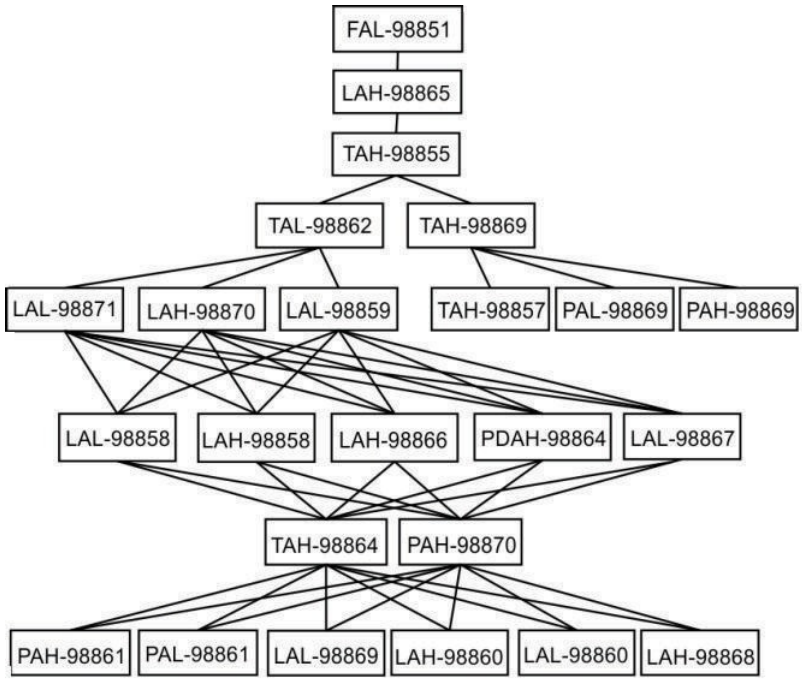


Figura 5.3: Exemplo de filas de ordenação de alarmes.

do resultado relações improváveis devido à disposição no fluxo de energia e fluidos do processo.

Assim, como pode ser observado pela análise dos resultados apresentados nas Figuras 5.4 e 5.5, o conhecimento do processo industrial consolidado em ontologia permite elevar a um novo patamar os algoritmos de análise e diagnóstico de processos industriais.

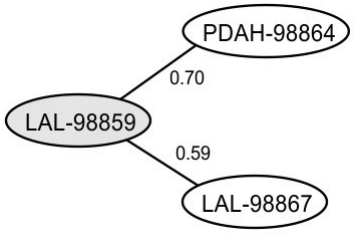


Figura 5.4: Correlação de alarmes com vizinhança de uma unidade.

5.2 Considerações Finais

Com a estruturação física de uma planta industrial usando a OntoAuto, foi possível desenvolver uma aplicação para realizar correlação semântica de alarmes. Com a inclusão de aspectos semânticos no procedimento de obtenção da correlação entre alarmes, ou seja, sem considerar apenas a temporalidade da ativação das ocorrências de alarmes, procedimentos de diagnósticos de anomalias em processos industriais podem ser aprimorados.

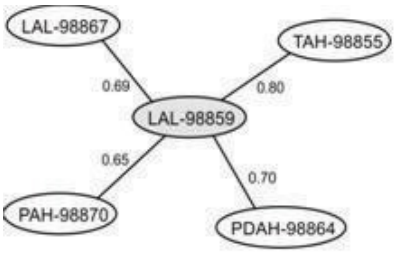


Figura 5.5: Correlação de alarmes com vizinhança de duas unidades.

Com essa nova abordagem semântica, permitiu-se um melhor auxílio de procedimentos de diagnósticos de anomalias em processos industriais, pois alarmes associados com equipamentos fisicamente desacoplados no processo industrial podem vir a apresentar alta correlação temporal por motivos alheios à causalidade dos eventos que descrevem a dinâmica do proceso. Deste modo, a análise baseada na simples correlação temporal entre alarmes pode levar a conclusões equivocadas, quando, por exemplo, há uma má configuração de alarmes.

A adição de informação vindas da estruturação do conhecimento permite melhorar bastante as análises de operação de processos industriais. No caso específico apresentado referente à correção de alarmes, a melhoria nos resultados foi proporcionada pela ordenação física causal dos alarmes e determinação do nível de vizinhança entre eles, que só foi possível através da estruturação do conhecimento da ontologia.

Capítulo 6

Aplicação 03: Avaliação Econômica

Na área de engenharia química, a engenharia de processos é a responsável por sistematizar os procedimentos de projeto, de modo a permitir maior rapidez, maior segurança e menor custo na execução dos projetos. Uma das atividades do projeto de processos é a avaliação econômica preliminar, que se trata de um procedimento adequado para a discriminação de alternativas de fluxogramas na fase preliminar do projeto.

Existem vários critérios para realizar a avaliação econômica de um processo, permitindo medir o seu desempenho econômico. Essa medição é feita através de critérios expressos por funções do tipo custo ou lucro [R. Turton & Shaeiwitz 2003]. Dentre os critérios existentes na literatura de avaliação econômica de projetos, o escolhido e utilizado nesta tese é um dos mais adotados, o *Venture Profit* [Rudd & Watson 1968], conhecido como Lucro do Empreendimento (LE). Nessa abordagem, o lucro corresponde a um lucro reativo que estima a vantagem de investir no processo industrial, sujeito a um risco comercial, em detrimento de um outro investimento que oferece uma taxa de retorno garantida, com risco zero [Perlingeiro 2005].

Nesta tese, foi criada uma nova ontologia, OntoEcon, que importa os conceitos existentes na OntoAuto para modelar o critério *Venture Profit* para avaliação econômica de processos, definindo formalmente os conceitos envolvidos.

6.1 Critério de Avaliação Econômica

O critério de avaliação econômica Venture Profit, Lucro do Empreendimento (*LE*) [Perlingeiro 2005], assume um investimento total, *Itotal*, no processo que deverá ser completamente recuperado pela empresa ao final da vida útil das instalações da planta. A receita (*R*) a ser gerada quando o processo tiver em execução corresponde ao preço de venda do produto (*PV*) multiplicado pela taxa de produção prevista (*PROD*):

$$R = PV * PROD \tag{6.1}$$

Então, o primeiro critério de avaliação do potencial econômico do processo é a margem bruta (*MB*). Se essa margem for maior do que zero, significa que o projeto pode prosseguir com o seu dimensionamento e a inclusão dos demais custos. A margem bruta é calculada pela fórmula descrita abaixo, ou seja, pela receita subtraída dos custos com

matéria prima e insumos (*Cmatprim*).

$$MB = R - Cmat\,prim \tag{6.2}$$

A próxima variável a ser analisada no procedimento de avaliação econômica é o lucro bruto (*LB*), que corresponde à receita subtraída de todos os custos do processo (*Ctotal*). O lucro bruto não corresponde ao desempenho final do empreendimento, pois não é o retorno total para a empresa, já que ainda é preciso debitar valores com depreciações (*D*) e imposto de renda (*Irenda*).

$$LB = R - Ctotal \tag{6.3}$$

Na sequência são calculados o lucro líquido antes do imposto de renda (*LA*) e o lucro líquido depois do imposto de renda (*LD*). Para que a operação do processo seja rentável, *LD* tem que ser maior do que zero.

$$LA = LB - D \tag{6.4}$$

$$LD = LA - Irenda \tag{6.5}$$

O cálculo do lucro do empreendimento deve considerar dados sobre o que a empresa lucraria se investisse em um outro empreendimento semelhante. Assim, para calcular o lucro líquido descontado o retorno sobre o investimento alternativo (*LL*), deve-se debitar de *LD* uma variável chamada de retorno sobre o investimento alternativo (*RI*), que corresponde à taxa de retorno (*TR*) multiplicada pelo *Itotal*.

$$LL = LD - RI \tag{6.6}$$

Para finalizar, deve-se debitar do lucro final uma parcela chamada de compensação pelo risco (*CR*), que é estimada pela taxa de risco (*h*) multiplicada pelo *Itotal*.

$$CR = h * Itotal \tag{6.7}$$

Então, o lucro final do empreendimento (*LE*) corresponde à:

$$LE = LB - (D + IR + RI + CR) \tag{6.8}$$

Se o valor de *LE* for positivo significa que o investimento no processo deverá ser vantajoso em detrimento do investimento alternativo. Para se chegar ao valor estimado de *LE*, precisa-se estimar os custos e investimentos.

Conforme descrito em [Perlingeiro 2005], os custos e investimentos são descritos pelas equações presentes nas Figuras 6.1 e 6.2.

$$C_{total} = C_{prod} + C_{gerais}$$
$$C_{prod} = C_{diretos} + C_{fixos}$$
$$C_{diretos} = (C_{matprim} + C_{util}) + 0{,}046 * I_{fixo} + 0{,}27 * C_{total}$$
$$C_{fixos} = 0{,}03 * I_{fixo}$$
$$C_{prod} = (C_{matprim} + C_{util}) + 0{,}076 * I_{fixo} + 0{,}27 * C_{total}$$
$$C_{gerais} = 0{,}025 * R$$
$$C_{total} = 1{,}37 * (C_{matprim} + C_{util}) + 0{,}104 * I_{fixo} + 0{,}034 * R$$

Figura 6.1: Equações para estimar custos.

$$I_{total} = I_{fixo} + I_{giro} + I_{partida}$$
$$I_{fixo} = I_{direto} + I_{indireto}$$
$$I_{direto} = 1{,}45 * ISBL$$
$$I_{indireto} = 0{,}25 * I_{direto}$$
$$I_{fixo} = 1{,}81 * ISBL$$
$$I_{giro} = 0{,}15 * I_{total}$$
$$I_{partida} = 0{,}10 * I_{fixo}$$
$$I_{total} = 2{,}34 * ISBL$$

$$ISBL = fT * fD * fL * \sum I_{ei}$$
$$OSBL = 0{,}45 * ISBL$$

fT = fator experimental de transferência de região.
fD = fator de atualização de preços para o ano vigente.
fL = Fator de Lang, que leva em conta a aquisição de outros itens indispensáveis à instalação dos equipamentos.
I_{ei} = preço de compra do equipamento.

$$LE = 0{,}48 * R - 0{,}68 * (C_{matprim} + C_{util}) - 0{,}54 * ISBL$$

Figura 6.2: Equações para estimar investimentos.

6.2 OntoEcon: Ontologia para Avaliação Econômica

Considerando o critério de avaliação econômica, Lucro do Empreendimento, apresentado anteriormente, foi necessário criar uma nova ontologia, OntoEcon, que importa a OntoAuto, acrescentando novos conceitos e novas propriedades. Para isso foram acrescentadas as classes descritas na Figura 6.3. Ressalta-se que um grande benefício proporcionado pela estruturação adequada da informação é justamente a facilidade de se estender ontologias.

A classe "EconomicEvaluation" define o conceito que representa uma determinada avaliação econômica de uma planta industrial. A Tabela 6.1 apresenta algumas propriedades relacionadas a ela e que são parâmetros definidos previamente para iniciar a avaliação.

Tabela 6.1: Propriedades associadas à classe "EconomicEvaluation".

Propriedade	Domínio	Tipo de Dados	Descrição
depreciationRate	EconomicEvaluation	float	Representa a taxa de depreciação das instalações físicas que se deteriorizam durante a vida útil do processo.
irRate	EconomicEvaluation	float	Representa a taxa do imposto de renda aplicada.
lifetime	EconomicEvaluation	float	Representa a vida útil da planta.
productionCapacity	EconomicEvaluation	float	Representa a capacidade de produção da planta.
returnRate	EconomicEvaluation	float	Representa a taxa de retorno ao se investir em um investimento alternativo.
riskRate	EconomicEvaluation	float	Representa a taxa de risco do empreendimento dar certo.
sellingPrice	EconomicEvaluation	float	Representa o preço de venda do produto final do processo industrial.

O cálculo das equações apresentadas na seção anterior é realizado basicamente considerando dados de custos e investimentos com equipamentos, tubulações, matéria prima e utilidades. Também são levados em conta alguns parâmetros definidos pelo fator de Lang. Dessa forma, foram modeladas na ontologia algumas classes que representassem essa associação. Na Figura 6.3 as subclasses da classe "EconomicEvaluationComponent" presentes na Tabela 6.2 representam esse conhecimento.

Tabela 6.2: Subclasses de "EconomicEvaluationComponent".

Classe	Descrição
EquipmentEvaluationComponent	Representa um item do custo com equipamentos e tubulações.
MaterialEvaluationComponent	Representa um item do investimento com matéria prima.

UtilityEvaluationComponent	Representa um item do investimento com utilidades, por exemplo, com água, vapor, etc.
LangItemEvaluationComponent	Representa um item que impacta no fluxo através do fator de Lang.

As propriedades criadas associadas às classes presentes na Tabela 6.2 estão apresentadas na Tabela 6.3.

Tabela 6.3: Outras propriedades da ontologia.

Propriedade	Domínio > Tipo de Dados	Descrição
isEconomicEvaluation	EconomicEvaluationComponent > EconomicEvaluation	Associa um item do componente de análise da avaliação econômica à respectiva avaliação.
isPlant	EconomicEvaluation > IndustrialProcess	Associa a avaliação econômica a um processo industrial instanciado na ontologia.
isComponent	EquipmentEvaluationComponent > Equipment, Pipe, Acessory	Representa o componente associado ao item do custo relacionado aos equipamentos e tubulações.
isRawMaterial	MaterialEvaluationComponent > RawMaterial	Representa a matéria prima associada ao item correspondente do investimento..
isUtility	UtilityEvaluationComponent > Utility	Representa a utilidade prima associada ao item correspondente do investimento..
isLangFactor	LangItemEvaluationComponent > LangFactor	Representa o tipo do fator de Lang associado ao item de análise de custo, podendo ser fatores de natureza física, de despesas adicionais ou demais parâmetros para o cálculo de investimentos.

hasCost	EconomicEvaluationComponent > float	Custo associado a um item da análise do investimento, podendo ser o valor de um equipamento, de uma quantidade de matéria prima, etc.

6.3 Estudo de Caso - Avaliação Econômica

Com base na representação do conhecimento do processo industrial e da metodologia de avaliação econômica, desenvolveu-se um aplicativo para auxiliar nessa atividade. O estudo de caso foi validado utilizando um processo de produção do lauril éter sulfato de sódio, cujo fluxograma está ilustrado na Figura 6.4.

O primeiro passo para avaliação econômica é criar uma instância da ontologia OntoEcon para modelagem de processos industriais mapeando o fluxograma da Figura 6.4. Na sequência, uma instância da classe "EconomicEvaluation" é criada, associando às suas propriedades os dados presentes na Figura 6.5.

A Figura 6.6 apresenta instâncias das subclasses da classe "EconomicEvaluationComponent". Cada item do painel "Investimento em Equipamentos" são instâncias da classe "EquipmentEvaluationComponent", cada item do painel "Custos com Matéria Prima" são instâncias da classe "MaterialEvaluationComponent", cada item do painel "Custos Utilidades" são instâncias da classe "UtilityEvaluationComponent" e, por fim, os itens do painel "Parâmetros" são instâncias das subclasses da classe "LangItemEvaluationComponent".

Uma vez que todo o conhecimento necessário para avaliação econômica está representado na ontologia, facilmente consegue-se aplicar as equações apresentadas no critério de avaliação escolhido. A Figura 6.7 apresenta o resultado dos cálculos das equações que auxiliam o especialista no projeto em relação a sua tomada de decisão quanto a avaliação econômica do processo industrial.

6.4 Considerações Finais

Nesta capítulo foi apresentada uma nova ontologia, a OntoEcon, aplicada ao domínio de avaliação econômica de processos industriais. A OntoEcon importou os conceitos e propriedades presentes na OntoAuto, acrescentando os seus próprios conceitos e propriedades específicas do domínio econômico. O método de avaliação econômica modelado foi o *Venture Profit*, mas também é possível incorporar outros métodos à ontologia.

Modelando processos com a OntoAuto e OntoEcon é possível obter informações úteis no procedimento de avaliação econômica, como por exemplo, na realização do orçamento dos componentes envolvidos. Com o processo modelado, é possível identificar facilmente quantos componentes de determinado tipo e com determinadas características existem e precisam se orçados.

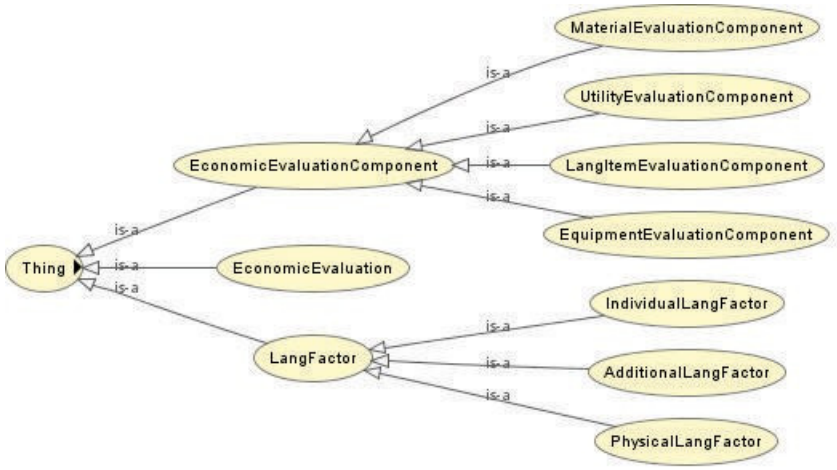


Figura 6.3: Novas classes da ontologia de avaliação econômica.

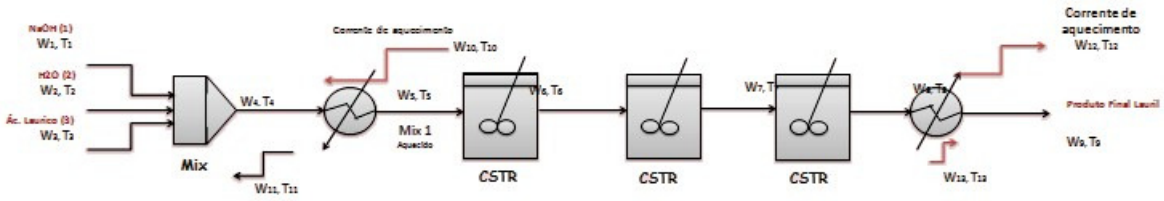
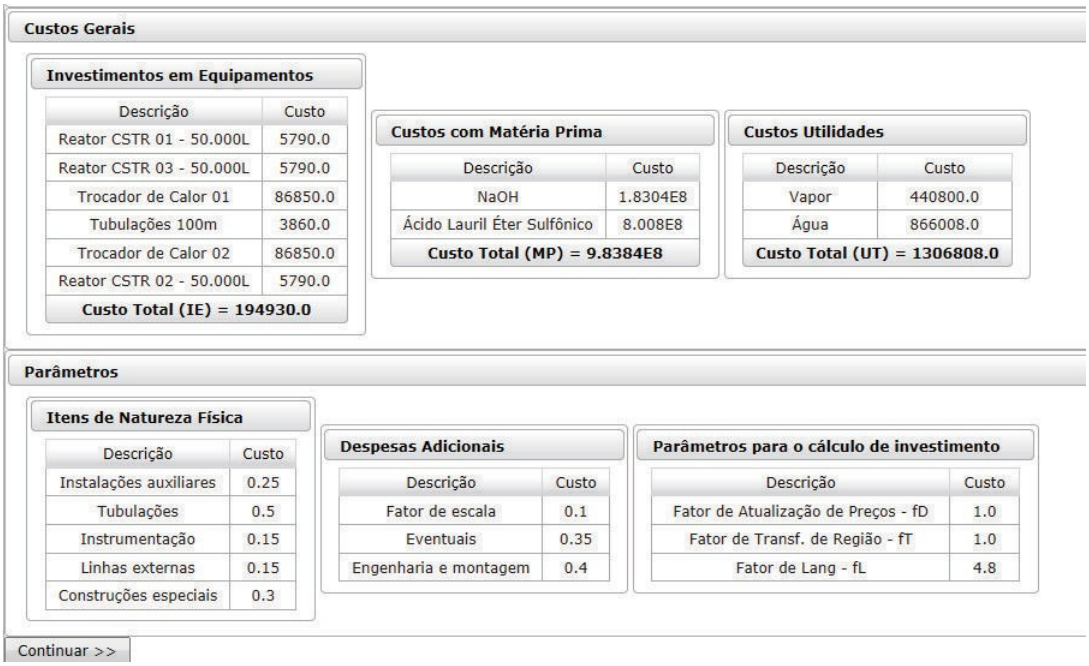


Figura 6.4: Fluxograma do Processo de Produção do Lauril Éter Sulfato de Sódio.

Dados da Produção

Capacidade de Produção (Kg/ano):	3.112E7
Preço de Venda: R$	45.0
Taxa de Depreciação ($/ano):	0.1
Taxa de Retorno i ($/ano):	0.17
Taxa de Risco h ($/ano):	0.01
Vida Útil (anos):	10.0

Figura 6.5: Primeiro passo do procedimento de avaliação econômica.



Custos Gerais

Investimentos em Equipamentos

Descrição	Custo
Reator CSTR 01 - 50.000L	5790.0
Reator CSTR 03 - 50.000L	5790.0
Trocador de Calor 01	86850.0
Tubulações 100m	3860.0
Trocador de Calor 02	86850.0
Reator CSTR 02 - 50.000L	5790.0
Custo Total (IE) = 194930.0	

Custos com Matéria Prima

Descrição	Custo
NaOH	1.8304E8
Ácido Lauril Éter Sulfônico	8.008E8
Custo Total (MP) = 9.8384E8	

Custos Utilidades

Descrição	Custo
Vapor	440800.0
Água	866008.0
Custo Total (UT) = 1306808.0	

Parâmetros

Itens de Natureza Física

Descrição	Custo
Instalações auxiliares	0.25
Tubulações	0.5
Instrumentação	0.15
Linhas externas	0.15
Construções especiais	0.3

Despesas Adicionais

Descrição	Custo
Fator de escala	0.1
Eventuais	0.35
Engenharia e montagem	0.4

Parâmetros para o cálculo de investimento

Descrição	Custo
Fator de Atualização de Preços - fD	1.0
Fator de Transf. de Região - fT	1.0
Fator de Lang - fL	4.8

Continuar >>

Figura 6.6: Segundo passo da avaliação econômica.

Investimentos

Cálculos dos Investimentos

ISBL:	R$ 4067799.2
OSBL:	R$ 1830509.6
Idireto:	R$ 5898309.0
Iindireto:	R$ 1474577.2
Ifixo:	R$ 7362716.5
Ipartida:	R$ 736271.7
Igiro:	R$ 1427797.5
Itotal:	R$ 9518650.0

Custos dos Investimentos

Custo sup. operacionais (R$/ano):	R$ 441762.97
Custo manutenção (R$/ano):	R$ 294508.66
Custo laboratórios (R$/ano):	R$ 1398030.4
Custo mão-de-obra (R$/ano):	R$ 2.7960608E7
Custo produção (R$/ano):	R$ 1.36821069E9
Custo matéria-prima (R$/ano):	R$ 9.8384E8
Custo Total (R$/ano):	R$ 1.39803046E9
Custo direto (R$/ano):	R$ 1.36295373E9
Custo Fixo (R$/ano):	R$ 220881.48
Custos Gerais:	R$ 3.501E7

Cálculos de Rentabilidade

Lucro Bruto:	R$ 2369536.0
Taxa de retorno (1/ano):	R$ 0.17
Depreciação:	R$ 736271.7
Lucro líquido antes IR:	R$ 1633264.2
IR:	R$ 408316.06
Lucro líquido depois IR:	R$ 1224948.2
Rentabilidade (%/ano):	R$ 12.868929

Figura 6.7: Passo final da avaliação econômica.

Capítulo 7

Aplicação 04: Confiabilidade

Cada vez mais a competitividade e a exigência do mercado demandam investimentos consideráveis para obtenção de qualidade e confiabilidade dos produtos. Baixa qualidade e baixa confiabilidade podem resultar em custos proibitivos. Um produto ou serviço de qualidade é aquele que atende adequadamente, de forma confiável, acessível, segura e no tempo certo ás necessidades do cliente [Campos 1992].

A confiabilidade é uma medida usada para caracterizar se um sistema está trabalhando corretamente em um período de tempo. Formalmente, é definida como a probabilidade que um sistema não falhe em um determinado intervalo de tempo [Campos 1992]. A análise da confiabilidade é uma parte importante da fase de projeto de sistemas complexos, em aplicações de alto custo e alto risco, já que alterar um projeto na fase inicial é mais simples e flexível, além de ser na fase de projeto onde se define a maior parte do custo do produto.

Para obter uma maior confiabilidade, deve-se investir em ferramentas relacionadas à análise de falhas, como por exemplo a Análise de Modo de Falhas e seus Efeitos (FMEA - *Failure Mode and Effects Analysis*), a Análise de Árvore de Falhas (FTA - *Fault Tree Analysis*) e o Diagrama de Ishikawa. O fato de se levantar as possíveis falhas do produto permite que, ainda na fase de projetos, possa-se definir soluções preventivas. Com isso, é possível estabelecer a manutenabilidade do produto, já que as falhas e seus possíveis efeitos são previamente conhecidos. Além disso, o tipo de manutenção (corretiva, preventiva, preditiva) pode ser escolhido [Sakurada 2001].

As ferramentas FTA e FMEA são muito utilizadas na área de confiabilidade. Com a FMEA é possível realizar uma análise local, procurando determinar os modos de falhas dos componentes e como afetam (efeitos) os níveis superiores do sistema. Já o FTA permite uma análise global, onde se inicia com a escolha de um evento topo (possível falha no sistema) e se parte em busca das falhas nos componentes. O Diagrama de Ishikawa é uma ferramenta mais simples, relaciona as causas que podem influenciar em um dado efeito [Sakurada 2001].

Nessa tese, realizou-se a representação do conhecimento de uma abordagem para geração automática de árvore de falhas. Para isso foi criada um nova ontologia, a OntoConf, que também importa os conceitos e propriedades da OntoAuto e acrescenta seus próprios específicos a esse novo domínio.

7.1 Conceitos Relacionados

É muito importante saber diferenciar os conceitos de falhas, defeitos e erros. Uma falha é um tipo de defeito que pode originar um erro. Ela é definida como ativa quando causa um erro, caso contrário é definida como dormente. Segundo [A. Avizienis 2004], as falhas podem ser agrupadas em três grupos: falhas de projeto (fases ocorridas durante a fase de desenvolvimento do sistema), falhas físicas (afetam o *hardware* dos equipamentos) e falhas de operação (ocorrem durante a utilização dos sistemas).

Um defeito pode ser definido como sendo uma manifestação de eventos que ocorre quando o sistema desvia do serviço correto. Já um erro pode ser definido como algo que caracteriza um estado incorreto por parte do sistema. Os defeitos podem ser causados por erros, que tem como causas as falhas [da Silva 2013]. Os defeitos aparecem quando os erros são propagados no sistema. Caso uma parte do sistema que tenha erros nunca seja usada, o defeito poderá nunca ocorrer.

A Figura 7.1 apresenta a relação entre falhas, erros e defeitos. A ativação de uma falha pode ocasionar um erro no sistema, que por sua vez pode propagar um defeito. A causa do defeito é a falha.

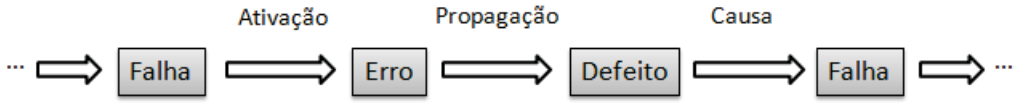


Figura 7.1: Relação entre falhas, erros e defeitos.

A habilidade de um sistema evitar falhas nos serviços mais críticos é conhecida como dependibilidade e pode ser caracterizada pela combinação dos seguintes conceitos [A. Avizienis 2004]:

- Confiabilidade: probabilidade de um defeito não ocorrer em um determinado período de tempo.
- Integridade: o serviço não pode ser modificado sem autorização.
- Manutenabilidade: capacidade de ser reparado ou sofrer manutenção.
- Disponibilidade: habilidade em fornecer o serviço correto quando solicitado. Em outras palavras, a probabilidade do sistema estar operacional quando solicitado.
- Segurança: ausência de consequências catastróficas para os usuários do sistema.

A confiabilidade e a disponibilidade podem ser utilizadas como medidas para definição de dependabilidade. A confiabilidade é uma medida cuja relevância está direcionada para os sistemas com alta sensibilidade em caso de defeitos, ou seja, sistemas cujo os serviços não podem ser interrompidos. Já a disponibilidade é relevante quando sistemas apresentam tolerância a pequenas interrupções.

Uma forma de avaliar a confiabilidade de um sistema é através dos valores médios/esperados das distribuições de defeitos. O tempo médio de funcionamento até a ocorrência de um

defeito (MTTF) é a medida mais utilizada [Shooman 1990]. Já no caso da disponibilidade são utilizadas as medidas de MTTF e MTTR (tempo médio até o sistema reparar um defeito).

7.2 Árvore de Falhas - FTA

O método da Análise da Árvore de Falhas (FTA) facilita a análise da confiabilidade dos sistemas, mapeando o relacionamento causa-efeito dos eventos, possibilitando um melhor conhecimento do funcionamento do sistema e dos mecanismos de falhas. Segundo [Contini 1995], a técnica FTA é uma técnica dedutiva formalizada, que permite a investigação das possíveis causas da ocorrência de estados pré-identificados indesejados do sistema. Esse estado, referido como evento de topo, está associado com o comportamento anormal do sistema, causado por uma falha no equipamento, ou erros humanos e /ou pertubações externas.

Uma árvore de falhas é formada por:

- Eventos: representam as condições normais e de falhas do sistema (defeitos nos componentes, condições ambientais, falhas humanas, etc). Representados graficamente por retângulos, círculos, losangos e triângulos.
- Evento topo: possível falha no sistema escolhida para análise. Corresponde à raiz da árvore de falhas.
- Evento repetido: aquele que ocorre mais de uma vez na árvore de falhas.
- Portas lógicas: mapeiam as relações causa-efeito entre os eventos, conectam os eventos de acordo com suas relações causais. As portas lógicas podem ter mais de um evento de entrada, porém possuem apenas uma única saída. Por exemplo, portas lógicas "E" ou "OU".
- Nível hierárquico: os níveis hierárquivos são definidos a partir do evento topo. O primeiro nível hierárquico é formado pelos eventos relacionados diretamente com o evento topo. Os eventos relacionados aos eventos abaixo do primeiro nível hierárquico representam os eventos do segundo nível hierárquico, e assim por diante.

A Figura 7.2 [Sakurada 2001] apresenta um exemplo de árvore de falhas. Observa-se que a mesma tem dois níveis hierárquicos, duas portas lógicas "E" e uma porta lógica "OU". Os eventos estão representados pelos seguintes símbolos [Henley & Kumamoto 1981]:

- Retângulo: representa um evento de falha resultante de uma combinação de falhas básicas que atuam através de portas lógicas.
- Círculo: denominado de evento básico, representa uma falha básica de um componente, delimitando o limite de resolução da árvore de falhas. São eventos que possuem informações métricas de confiabilidade (Tempo médio entre falhas, Tempo médio até a falha, confiabilidade, taxa de falhas, etc.)
- Losango: representa um evento não desenvolvido, ou seja, um evento que não se tem informação adicional para prosseguir com a análise detalhada.

- Triângulos: são eventos denominados *tranfer-out* e *tranfer-in*, que referem-se a parte idênticas de relações causais. Usados para simplificar a árvore de falhas, evitando a duplicidade de eventos iguais. O triângulo *tranfer-out* é o que tem uma linha lateral, já o *tranfer-in* é o que tem uma linha saindo do seu topo. O triângulo *tranfer-in* sai da porta lógica onde serão usados os eventos que serão copiados para o ponto onde está o triângulo *tranfer-out*.

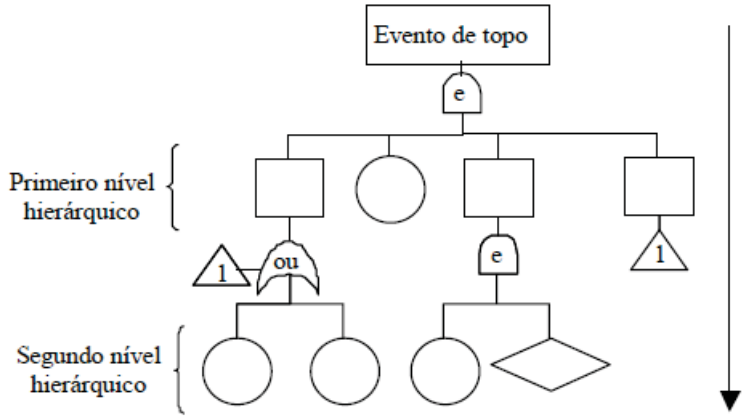


Figura 7.2: Exemplo de uma Árvore de Falhas.

A metodologia para construção de uma árvore de falhas inicia-se pela escolha do evento topo, que representa uma condição de defeito do sistema. A partir dele, é possível conduzir uma análise retroativa e encontrar as causas do defeito. Tendo construído a árvore de falhas, é possível realizar uma análise quantitativa, que consiste em calcular a probabilidade do evento TOPO baseado nas probabilidades dos eventos básicos.

7.2.1 Análise de Árvore de Falhas - FTA

Algumas das abordagens existentes para análise de árvore de falhas são a confiabilidade e a disponibilidade. Nessa subseção, elas serão apresentadas.

Confiabilidade

A confiabilidade, já descrita anteriormente, é formalmente definida como a probabilidade que um sistema não falhe em um intervalo de tempo $(0,t]$. A equação 7.1 representa a equação da confiabilidade $R(t)$, onde é assumido que o tempo de falha de um componente é uma variável aleatória definida por uma função de distribuição acumulativa $F(t)$.

$$R(t) = P(T > t) = 1 - F(t) \tag{7.1}$$

Uma outra forma de avaliar a confiabilidade do sistema é através dos valores médios/esperados ($E(t)$) das distribuições de defeitos. Em geral, o tempo médio de funciona-

mento até a ocorrência de um defeito (MTTF) é a medida mais utilizada [Shooman 1990] e está representada na equação 7.2.

$$MTTF = E(t) = \int_0^\infty t f(t)dt = \int_0^\infty R(t)dt \tag{7.2}$$

Disponibilidade

A disponibilidade do sistema está relacionada diretamente às ações de reparo, sendo definida como a probabilidade de um sistema funcionar no tempo t. A disponibilidade do instante t é referida como a disponibilidade instantânea $A(t)$, formalmente definida conforme equação 7.3. Essa métrica só é utilizada em um sistema que tem condição de probabilidade estacionária. Se $A(t) = 0$, significa que o sistema é não reparável, então o conceito disponibilidade iguala-se à definição de confiabilidade.

$$A = lim_{A \to \infty} A(t) \tag{7.3}$$

A função de confiabilidade é especificamente descrita com a função de taxa de falha $\lambda(t)$. Esta função (também conhecida como média do perigo) descreve a taxa de falha de um componente em um instante. Já a taxa de reparo $\mu(t)$ é a taxa que um componente danificado é reparado.

O MTTR (*Mean Time to Repair*) é definido como o tempo médio esperado para reparar um componente. Se as taxas de reparo e falha são assumidas constantes, respectivamente λ e μ, assumindo-se que n períodos ocorreram, podemos encontrar A_∞ através da equação 7.4 [Rausand & Hsyland 2004]:

$$A_\infty = \frac{MTTF}{MTTF + MTTR} \tag{7.4}$$

Analisar quantitativamente uma árvore de falhas consiste em calcular a probabilidade do evento topo baseado nas probabilidades dos eventos básicos. Este cálculo é realizado diferentemente para cada tipo de porta lógica. Assumindo n entradas/eventos independentes, onde a ocorrência do evento i é descrita pela sua função de distribuição acumulativa (CDF) $Fi(t)$, podemos descrever as saídas das portas lógicas (CDF) conforme exibido na Figura 7.3.

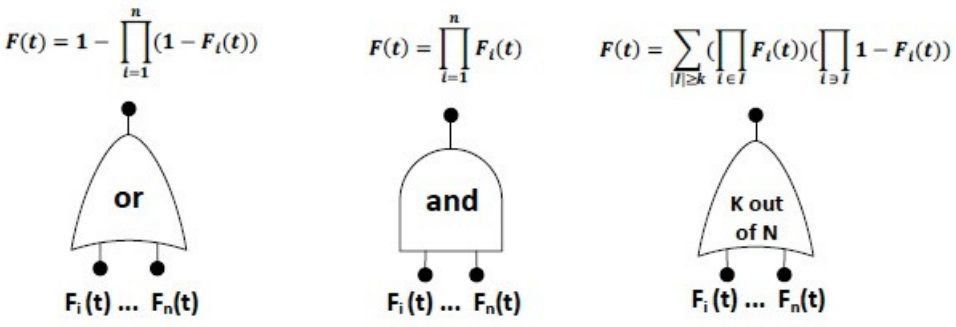


Figura 7.3: Função de distribuição acumulativa para as saídas das portas lógicas.

As equações descritas na Figura 7.3 somente são válidas quando a árvore de falhas não apresenta eventos repetidos. Quando há eventos repetidos, pode-se utilizar outras abordagens, como por exemplo, o princípio de inclusão-exclusão, soma de produtos disjuntos (SDP), fatoração, métodos recursivos diretos/indiretos [Limnios 2007].

7.3 Metodologia para Geração Automática de Árvore de Falhas

Para se desenvolver a aplicação para geração automática de árvores de falhas, inicialmente foram seguidos os passos presentes na Figura 7.4. A princípio foi criada uma ontologia, a OntoAuto, envolvendo os conceitos gerais relacionados aos processos industriais. Em seguida, analisou-se o algoritmo descrito no trabalho de [Majdara & Wakabayashi 2009] para geração automática de árvores de falhas. A partir daí, identificou-se que alguns conceitos e propriedades mais específicos para a área de confiabilidade deveriam também ter o conhecimento representado na ontologia. Com isso, foi criada uma nova ontologia, a OntoConf, que importa o que é representado na OntoAuto e acrescenta o que é específico para a geração de árvores de falhas. Por fim, desenvolveu-se uma aplicação que, dada uma instância da OntoConf para um dado processo industrial, ela gera automaticamente as árvores de falhas para os eventos topo pré-selecionados.

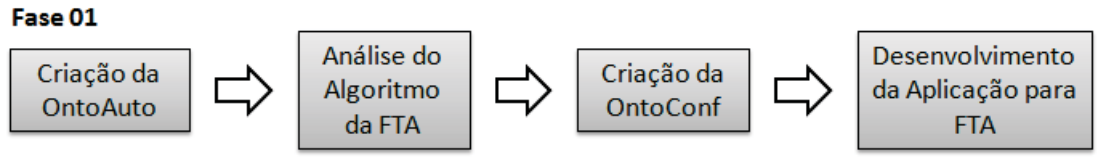


Figura 7.4: Fase 01: Metodologia para desenvolvimento da árvore de falhas automaticamente

O trabalho descrito em [Macedo et al. 2013] apresenta uma ferramenta para análise de árvore de falhas utilizando as abordagens apresentadas anteriormente, como por exemplo, confiabilidade e disponibilidade. Nessa ferramenta, tendo uma determinada árvore de falhas, é possível gerar gráficos para análises da árvore segundo essas abordagens. Essa ferramenta permite importar uma árvore de falhas através de um arquivo com extensão ".txt", que deve ser preenchido com a representação de uma árvore de falhas. Nesta tese, também foi feita uma integração com essa ferramenta, ou seja, a aplicação para geração automática de árvore de falhas gera a árvore e converte para o arquivo ".txt" que pode ser importado na ferramenta desenvolvida em [Macedo et al. 2013].

Essa integração pode ser resumida pela Figura 7.5. Inicialmente, para um dado processo é criada uma instância da OntoConf. Em seguida, é executada a aplicação para geração automática de árvore de falhas com base em um evento topo pré selecionado. Com a árvore de falhas gerada automaticamente, é possível convertê-la para o arquivo ".txt" que é importado na ferramenta para análise de árvore de falhas desenvolvida em [Macedo et al. 2013].

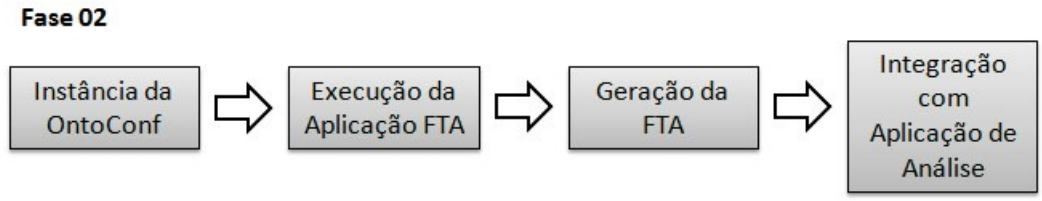


Figura 7.5: Fase 02: Metodologia para desenvolvimento da árvore de falhas automaticamente

A construção manual de uma árvore de falhas é uma tarefa que consome muito tempo, além de ser suceptível a erros humanos, visto que muitos passos precisam ser executados. No trabalho desenvolvido em [Majdara & Wakabayashi 2009], apesar do autor propor uma metodologia de automatizar a geração de árvore de falhas, não apresenta nenhuma aplicação desenvolvida que a gere automaticamente.

O algoritmo proposto por [Majdara & Wakabayashi 2009] para geração automática das árvores de falhas utiliza os seguintes conceitos:

- Componente: mesmo conceito definido na OntoAuto, representa equipamentos, instrumentos, tubulações, eventos externos, etc.
- Tabelas de funções: descrevem a relação entre as entradas e saídas dos componentes, podendo associar a uma condição de funcionamento ou estado de determinado componente. Para os componentes que a tabela de funções possui a indicação do estado, deve-se ter associada uma tabela de transição de estados.
- Tabelas de transição de estados: usadas para alguns componentes para os quais é necessário mapear transições de estados. São usadas em conjunto com as tabelas de funções e mapeiam as mudanças de estado em função das entradas.

Para descrever melhor o algoritmo, considere a representação de um sistema simplificado de controle de incêndio presente na Figura 7.6. Cada quadrado representa um componente no sistema e as setas como eles estão relacionados através de entradas e saídas.

Cada componente na Figura 7.6 possui uma tabela de funções. E os componentes “Switch” e “Valve” que necessitam de representação de estado possuem também tabelas de transição de estados. Para exemplificar, vamos considerar as tabelas dos componentes “Relay” e “Switch”.

A Tabela 7.1 apresenta a tabela de funções do “Relay”. Essa tabela de funções permite identificar os valores de saídas do componente através da combinação dos valores de suas entradas, condição de funcionamento e/ou estado. Por exemplo, o componente “Relay” assumirá como saída com valor 1, se a Entrada 1 for igual a 1 e a entrada 2 for igual a 1 e a condição de funcionamento for “OK”, ou então, se a entrada 1 for igual a 1 e a condição de funcionamento for “Stuck closed”.

Se nas células da tabela aparecer o valor “-”, significa que tanto faz o valor da entrada/condição de funcinamento para a combinação representada pela linha na tabela. Por

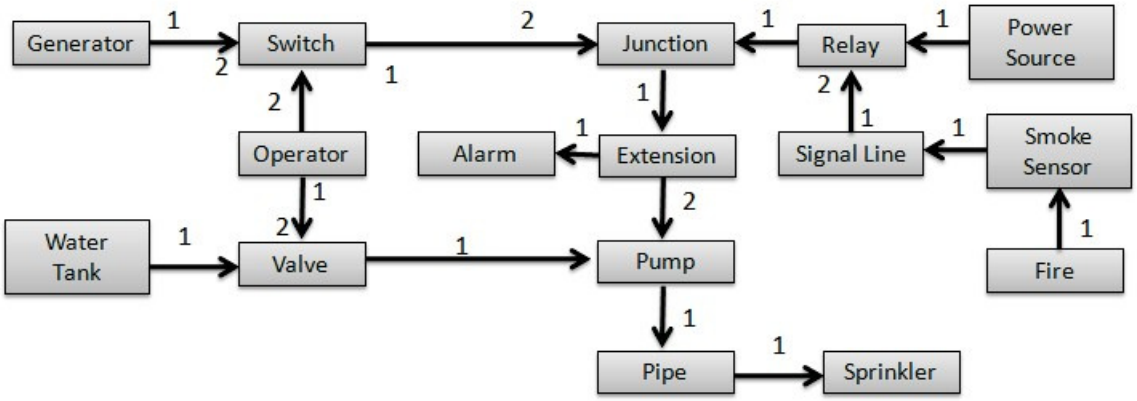


Figura 7.6: Esquema do sistema simplificado de controle de incêndio.

exemplo, na Tabela 7.1 a quarta linha representa uma das opções da saída do "Relay" ser igual a 0, que seria sempre que a condição de funcionamento for "Fail to close", independente do que tiver nos valores das entradas 1 e 2.

Tabela 7.1: Tabela de funções do componente "Relay"

Entrada 1	Entrada 2	Condição de Funcionamento	Saída 1
1	1	OK	1
0	-	-	0
1	-	Stuck closed	1
-	-	Fail to close	0
-	0	OK	0

No caso do "Switch", a representação é feita através da Tabela 7.2 de funções e da Tabela 7.3 de transição de estados. A leitura sempre parte da tabela de funções, por exemplo, a saída do "Switch" assumirá o valor 1, se a entrada 1 for igual a 1 e o estado dele for "Close". Analisando a tabela de transição de estados, verificamos que para o estado dele ser "Close", tem-se três possibilidades: entrada 2 igual a 1 e condição de funcionamento "OK", ou entrada 2 igual a 1 e estado inicial "Close" e condição de funcionamento "Fail to open", ou entrada 2 igual a 0 e estado inicial "Close".

O algoritmo para geração automática de árvore de falhas utiliza as informações contidas nessas tabelas e será detalhado na subseção seguinte.

Tabela 7.2: Tabela de funções do componente “Switch”

Entrada 1	Estado	Saída 1
0	-	0
1	Close	1
-	Open	0

Tabela 7.3: Tabela de transição de estados do componente “Switch”

Entrada 2	Estado Inicial	Condição de Funcionamento	Próximo Estado
2	-	OK	Close
2	Open	Fail to close	Open
1	Close	Fail to open	Close
1	-	OK	Open
0	Open	-	Open
0	Close	-	Close

7.3.1 Algoritmo para Geração Automática de Árvore de Falhas

O algoritmo para geração automática de árvore de falhas descrito em [Majdara & Wakabayashi 2009] está resumido na Figura 7.7. Ele inicia com a seleção do evento topo que vai está associado a um dos componentes do sistema. O próximo passo é definir se ele se refere a uma saída ou estado do componente. Se for a uma saída, segue para análise da tabela de funções, se for um estado, segue para a análise da tabela de transição de estados.

A análise da tabela de função inicia verificando quais linhas da tabela possuem saída com valor igual à saída em análise (inicialmente a saída em análise é a saída associada ao evento topo). Se tiver mais de uma linha na tabela correspondente, então uma porta lógica “OR” é inserida na árvore de falhas e segue-se analisando cada uma dessas linhas. Ao analisar uma determinada linha, verifica-se as suas colunas em busca das entradas e/ou a condição de funcionamento com valor “-”. Se tiver mais de uma coluna diferente de “-”, adiciona-se uma porta lógica “AND”.

Em seguida, analisa-se cada uma das colunas de entradas e condição de funcionamento/estado com valor diferente de “-”. Se a coluna na tabela de função se refere à uma condição de funcionamento e essa condição é diferente de “-”, adiciona-se um evento básico à árvore. Se a coluna for uma entrada, deve-se identificar a saída do outro componente vinculada à essa entrada e realiza-se a mesma análise da tabela de funções desse outro componente. Se a coluna for um estado, deve-se partir para a análise da tabela de transição de estados do componente relacionado.

A análise da tabela de transição de estados é semelhante à análise da tabela de funções, a diferença é que a busca se inicia verificando quais linhas da tabela possuem a coluna próximo estado com valor igual ao estado em análise. Se tiver mais de uma linha na tabela

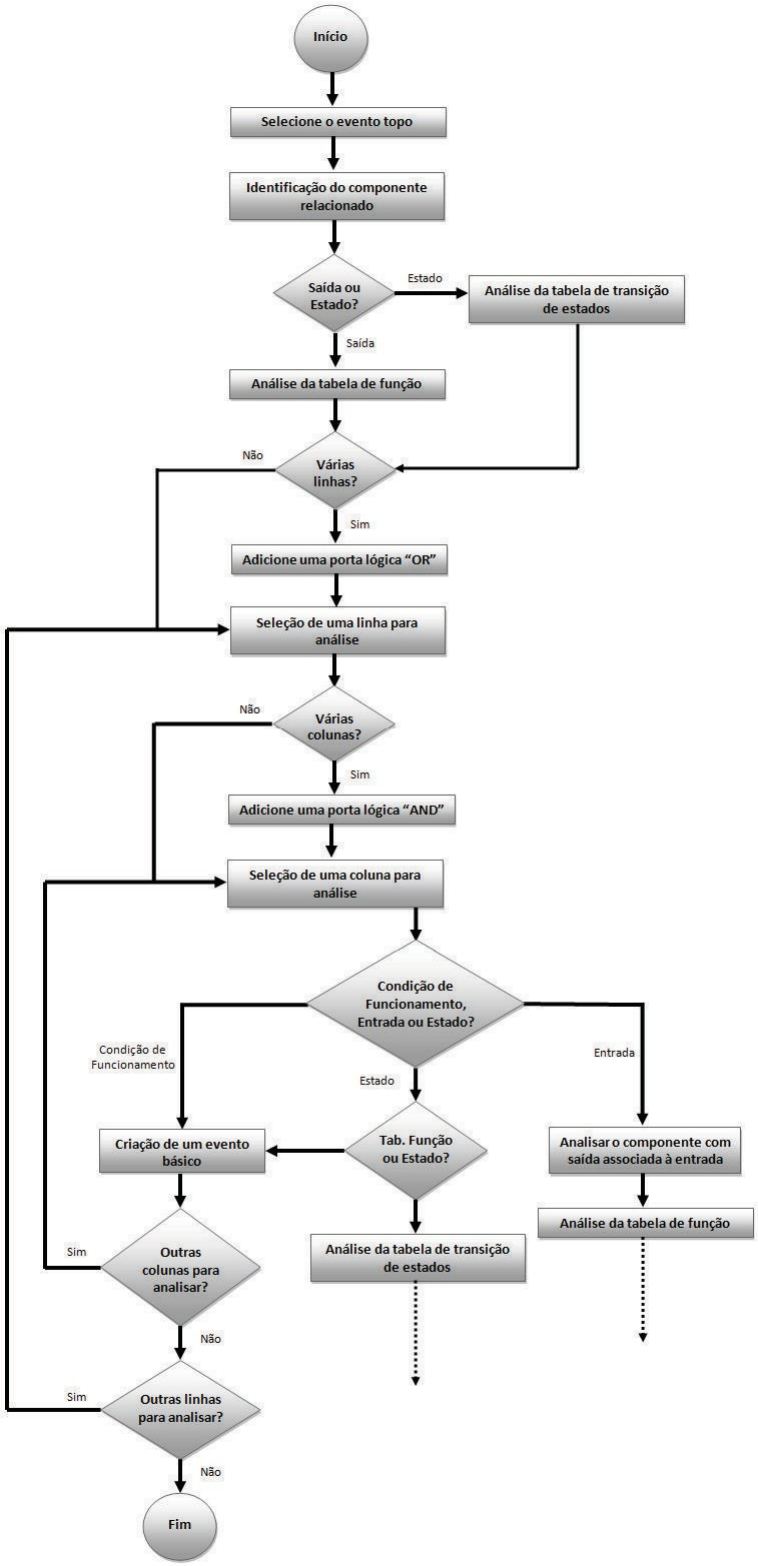


Figura 7.7: Fluxograma do algoritmo para geração automática de árvore de falhas.

correspondente, então uma porta lógica "OR" é inserida na árvore de falhas e segue-se analisando cada uma dessas linhas. Ao analisar uma determinada linha, verifica-se as suas colunas em busca das entradas e estado inicial com valor "-". Se tiver mais de uma coluna diferente de "-", adiciona-se uma porta lógica "AND". Em seguida, analisa-se cada uma das colunas de entradas e estado inicial. Se a coluna na tabela de função se refere a um estado inicial e esse estado é diferente de "-", adiciona-se um evento básico à árvore. Se a coluna for uma entrada, deve-se identificar a saída do outro componente vinculada à essa entrada e realiza-se a mesma análise da tabela de funções desse outro componente.

Mais adiante serão apresentados exemplos de árvores de falhas geradas automaticamente por esse algoritmo.

7.4 OntoConf: Ontologia para Confiabilidade

O algoritmo apresentado anteriormente foi utilizado com ontologias para implementar a aplicação de geração automática de árvores de falhas. O algoritmo realiza uma busca, a partir do evento topo selecionado, entre as tabelas de função e de transição de estados dos componentes através da relação entre suas entradas e saídas. Essa relação é mapeada na OntoAuto através das classes "Input" e "Output" e das propriedades "hasAssociateInput" e "hasAssociateOutput", conforme explicado no Capítulo 3. Porém, para a automatização desse algoritmo, foi necessário criar novas entidades e propriedades relacionadas ao conceito de confiabilidade e ao algoritmo proposto.

O primeiro grupo de classes criado está presente na Figura 7.8. As classes "ComponentState" e "FuncionalityCondition" foram criadas para representar, respectivamente, os estados e condições de funcionamento possíveis para os componentes. Observa-se que é possível especializar essas classes em subclasses específicas para os tipos de componentes. É importante ressaltar que nem todos os componentes precisarão ter estados mapeados.

As classes apresentadas na Figura 7.9 foram criadas para modelar os conceitos da tabela de função ("FunctionTable") e da tabela de transição de estados ("StateTable"). Além disso, também foi necessário mapear os itens dessas tabelas ("StateTableItem", "FuncionTableItem").

Já na Figura 7.10, temos a classe "InputOutputValue" criada para representar os possíveis valores de entrada e saída, por exemplo, os valores "1", "0", "Normal", "NW". Já as classes "InputTableItem" e "OutputTableItem" são utilizadas para representar a associação entre uma linha de uma das tabelas com suas respectivas entradas ou saídas e o valor assumido por elas naquela linha da tabela.

As Tabelas 7.4 e 7.5 apresentam as novas propriedades criadas para a representação da ontologia para geração automática de árvore de falhas (OntoConf). As novas classes e propriedades foram necessárias para permitir a execução do algoritmo conforme proposto em [Majdara & Wakabayashi 2009] sem intervenção manual, para isso foi preciso mapear todo o contexto das tabelas de função e de transição de dados, relacionando os itens da tabela com as entradas e saídas dos componentes mapeados pela OntoAuto.

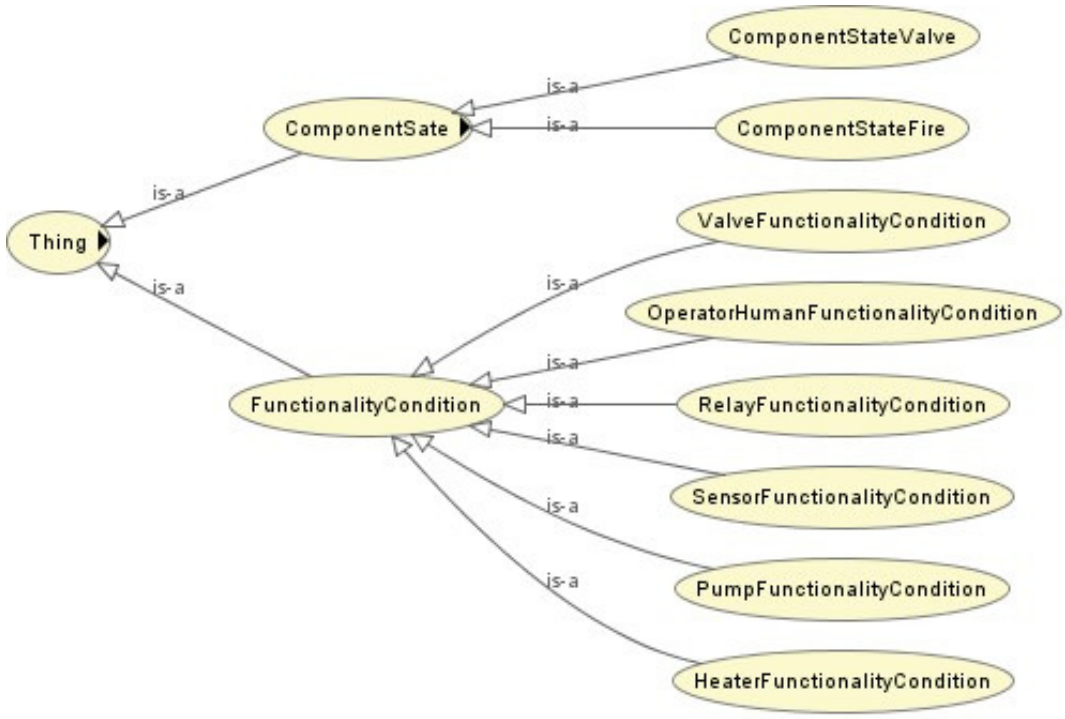


Figura 7.8: Classes da OntoConf - Parte 1.

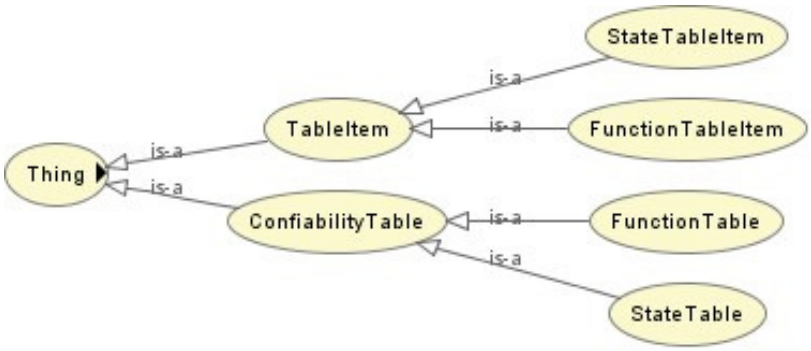


Figura 7.9: Classes da OntoConf - Parte 2.

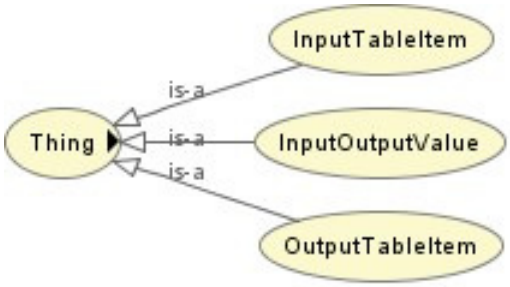


Figura 7.10: Classes da OntoConf - Parte 3.

Tabela 7.4: Propriedades da OntoConf - *Data Properties*.

Propriedade	Domínio	Tipo de Dado	Descrição
hasState	Component	boolean	Indica se o componente possui ou não estados a serem analisados, podendo assumir os valores true ou false. Para os componentes que tiverem valor true, significa que ele possuirá para a geração automática da árvore de falhas tabela de transição de estados.
isFault	Funcionality Condition	boolean	Indica se a condição de funcionamento representa ou não uma falha, podendo assumir os valores true ou false.
faultRate	Funcionality Condition	float	Representa a taxa de falha da condição de funcionamento.
repairRate	Funcionality Condition	float	Representa a taxa de reparo da condição de funcionamento.

Tabela 7.5: Propriedades da OntoConf - *Object Properties*.

Propriedade	Domínio	Tipo de Dado	Descrição
hasTable	TableItem	Confiability Table	Usado para associar os itens da tabela às suas respectivas tabelas de função e transição de estado
hasComponentTableItem	Confiability Table	Component	Usado para definição do componente associado às tabelas de função e de transição de estado.

hasFunctionalityCondition	TableItem	FunctionalityCondition	Usado na criação de uma tabela de função do componente para definir a condição de funcionamento associada à linha da tabela.
hasStateTableItem	Function TableItem	ComponentState	Usado na criação de uma tabela de função do componente, quando o componente possui também tabela de transição de estado. Define o estado associada à linha da tabela.
hasInitialComponentState	StateTable Item	ComponentState	Usado na criação de uma tabela de transição de estado para definir o estado inicial do componente associado à linha da tabela.
hasFinalComponentState	StateTable Item	ComponentState	Usado na criação de uma tabela de transição de estado para definir o próximo estado do componente associado à linha da tabela.
hasInputOutputValue	InputTable Item, OutputTable Item	InputOutputValue	Associa o valor de uma entrada ou saída a uma linha da tabela de função ou de transição de estados.
hasAssociateInputConf	InputTable Item	Input	Associa um determinado item de entrada da tabela de funções a entrada correspondente mapeada na OntoAuto.

hasAssociateOutputConf	OutputTable Item	Output	Associa um determinado item de saída da tabela de funções ou de transição de estados a saída correspondente mapeada na OntoAuto.
hasInputTable	TableItem	InputTableItem	Define para um item da tabela as entradas associadas.
hasOutputTable	TableItem	OutputTableItem	Define para um item da tabela as saídas associadas.

7.5 Estudo de Caso - Árvore de Falhas

Nessa seção serão exibidos alguns resultados obtidos com o desenvolvimento da aplicação para geração automática de árvores de falhas baseada em ontologias.

7.5.1 Exemplo 01: Sistema Simplificado de Aquecimento de Água

O primeiro exemplo corresponde a um sistema simplificado de aquecimento de água com esquema ilustrado na Figura 7.11.

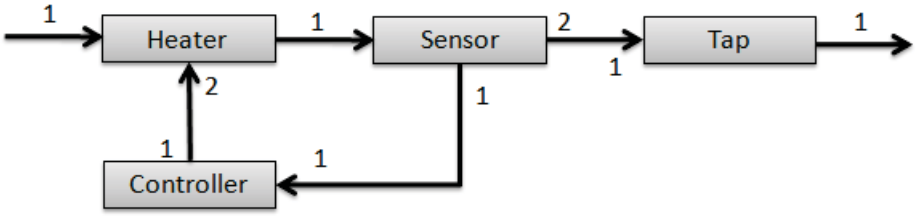


Figura 7.11: Esquema do sistema simplificado de aquecimento de água.

Esse diagrama de componentes e a ligação entre eles foi instanciado na ontologia. Além disso, também foram instanciadas na ontologia as tabelas de funções de cada um dos componentes. Nesse exemplo, não existem tabelas de transição de estados. Alguns exemplos de tabelas de funções para o exemplo da Figura 7.11 podem ser vistos na Figura 7.12. Cada tabela de função representada correspondente a um indivíduo da entidade "FuncionTable". Cada linha da tabela de função corresponde a um indivíduo da tabela "FunctionTableItem". As células das tabelas que possuem o valor "null" representam o

valor "-" conforme descrito anteriormente no algoritmo de geração automática de árvore de falhas.

Tabelas de Função

Tabela de Funções - Componente: Controller

Input 1	Functionality Condition	Output 2
N	OK	Norm.
null	Failed High	FP
null	Failed Low	Norm.
L	OK	FP

Tabela de Funções - Componente: Heater

Input 1	Input 2	Functionality Condition	Output 1
Normal	null	OK	W
Cold	FP	OK	W
Cold	Norm.	null	NW
null	null	Failed	NW
Too Cold	null	null	NW

Tabela de Funções - Componente: Sensor

Input 1	Functionality Condition	Output 1	Output 2
W	OK	N	N
null	Failed Low	L	N
null	Failed High	N	N
NW	OK	L	NW

Figura 7.12: Tabelas de funções para o exemplo da Figura 7.11.

No primeiro passo da aplicação desenvolvida, são exibidas as tabelas de função e/ou transição de estados conforme Figura 7.12. A partir dessa tela, o evento topo deve ser escolhido com base em uma das saídas. Por exemplo, se for escolhido como evento topo a saída 02 do componente "Sensor" como sendo o valor "NW", a aplicação irá gerar a árvore de falhas com base no algoritmo explicado anteriormente, explorando a associação entre saídas e entradas e analisando as linhas das tabelas de função.

A árvore de falhas gerada pela aplicação, para o caso em que o evento topo selecionado corresponde à saída 02 do componente "Sensor" assume o valor "NW", pode ser vista na Figura 7.13.

Na árvore de falhas gerada pelo sistema, pode-se considerar as seguintes legendas:

- E: evento a ser desenvolvido;

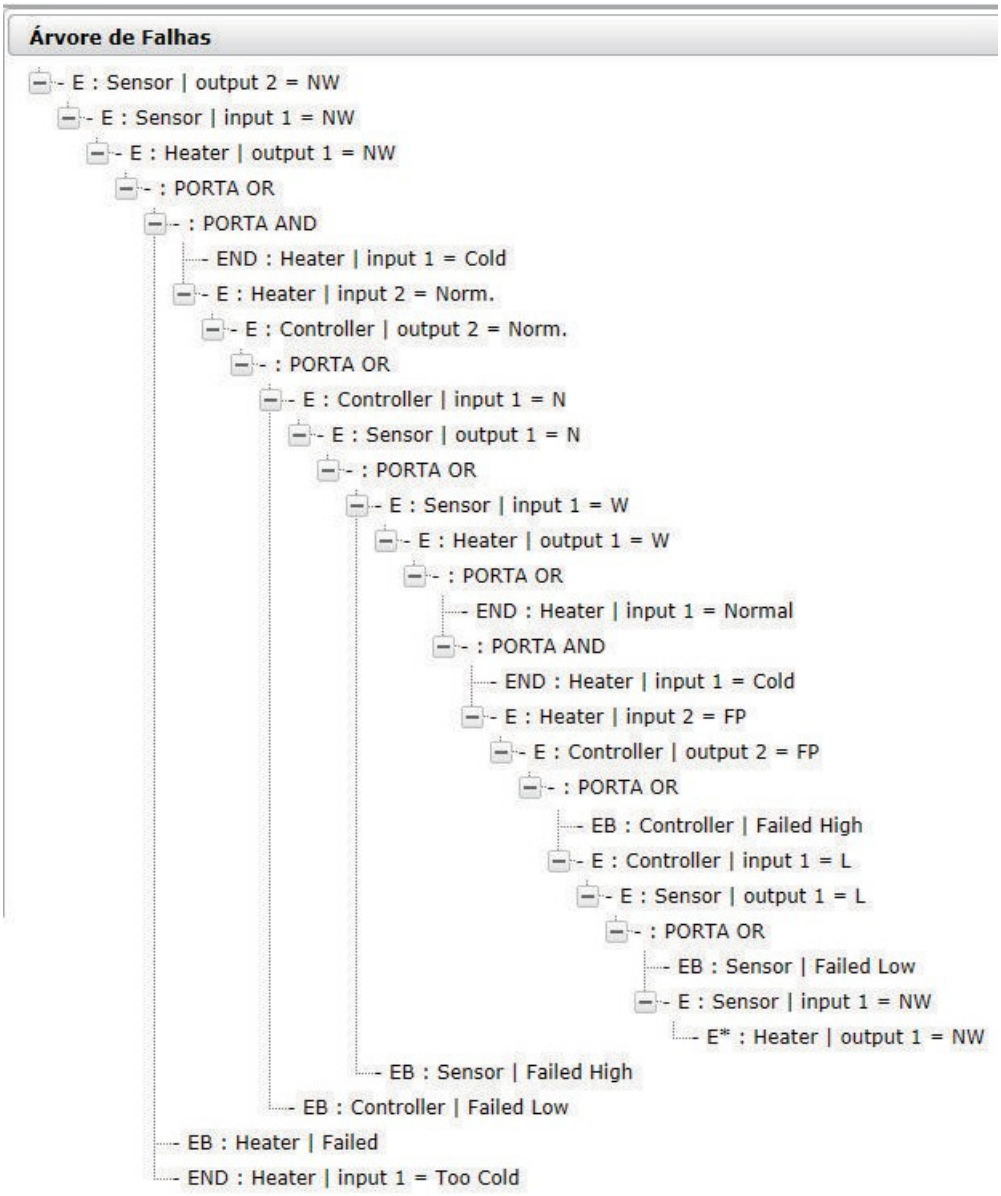


Figura 7.13: Árvore de falhas para o caso em que a saída 02 do “Sensor” assume valor igual a “NW”.

- EB: evento básico;
- END: evento não desenvolvido;
- E*: evento repetido;
- PORTA OR: porta lógica OR;
- PORTA AND: porta lógica AND.

Observe que para a saída 02 do “Sensor” ser igual a “NW”, a entrada 1 tem que ter valor igual à “NW”. A entrada 1 do “Sensor” corresponde a saída do componente “Heater” que também deve ser igual a “NW”. Para a saída do “Heater” ser “NW”, conforme

pode ser visto na tabela de função do mesmo, temos três linhas associadas, com isso, é adicionada uma porta lógica OR na árvore de falhas. Essa porta OR tem como entradas uma porta lógica AND (com entradas: entrada 1 do "Heater" igual a "Cold" e entrada 2 igual a "Norm."); a condição de funcionamento do "Heater" "Failed" (representado por um evento básico); e a entrada 1 do "Heater" "To cold". Quando se chega em uma condição de funcionamento, chega-se a um evento básico e não precisa mais desenvolver o algoritmo a partir desse ramo. Quando se chega em entradas de portas lógicas que são entradas, continua-se o desenvolvimento da árvore de falhas a partir das tabeças de funções das saídas associadas a essas entradas.

Outro exemplo de árvore de falhas gerada para esse sistema pode ser visto na Figura 7.14. Nesse caso, o evento topo corresponde a saída 01 do "Heater" assumindo valor "W".

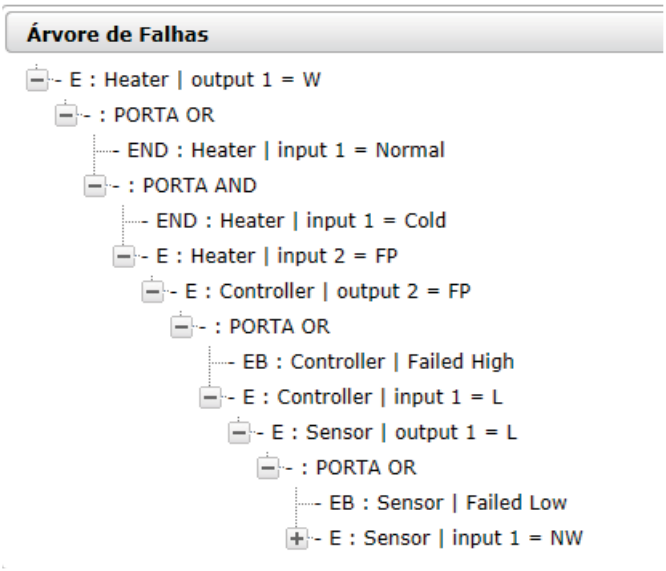


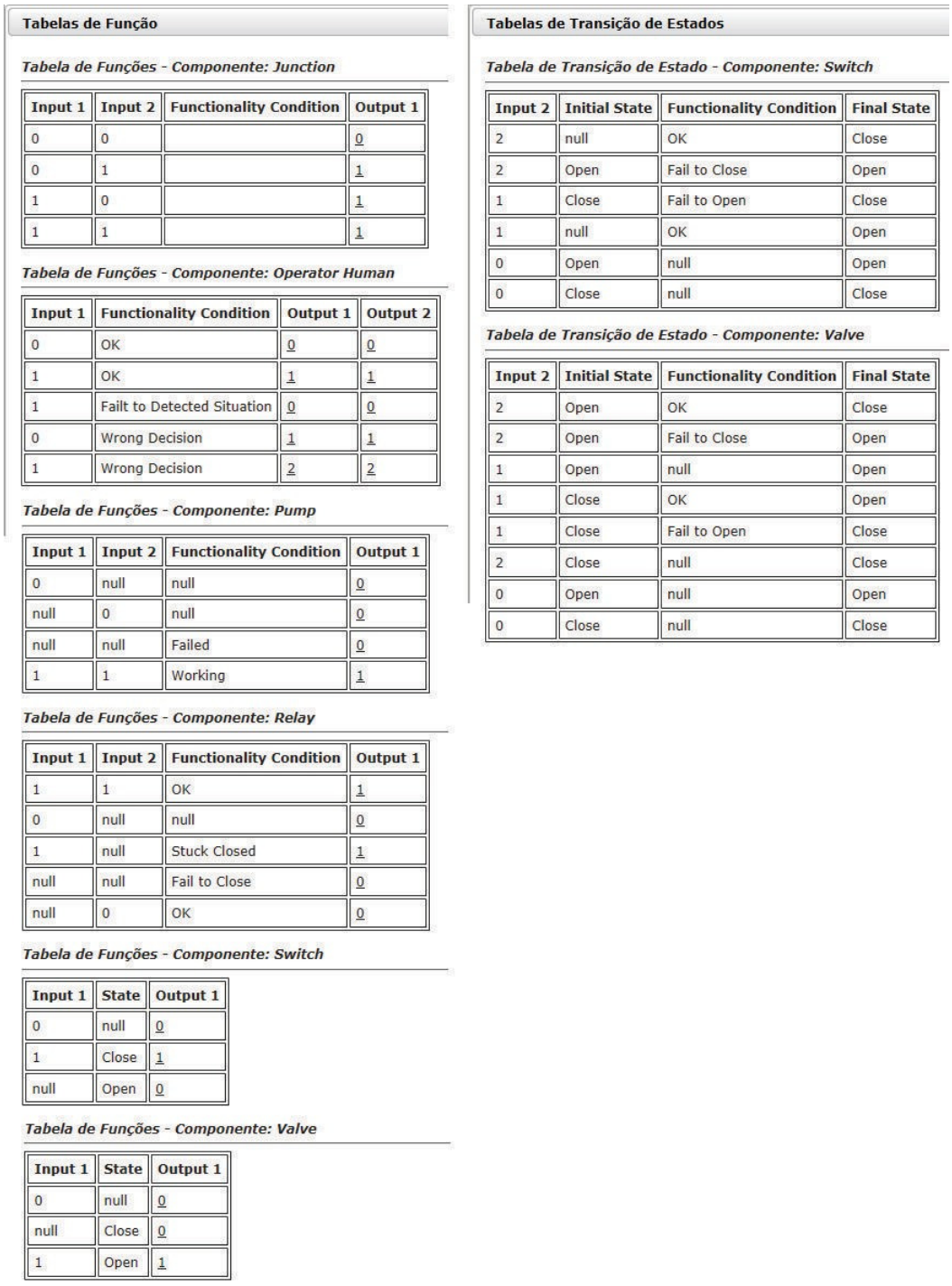
Figura 7.14: Árvore de falhas para o caso em que a saída 01 do "Heater" assume valor igual a "W".

7.5.2 Exemplo 02: Sistema Simplificado de Controle de Incêndio

O segundo exemplo corresponde ao sistema simplificado de controle de incêndio apresentado na Figura 7.6. Nesse exemplo os componentes "Switch" e "Valve" possuem também tabela de transição de estados. Semelhantemente ao exemplo anterior, o primeiro passo da aplicação exibe as tabelas de funções e de transição de estados mapeadas como indívidos da OntoConf para esse sistema, conforme pode ser visto na Figura 7.15.

Para esse exemplo, a árvore de falhas para o evento topo em que a saída 01 do "Relay" é igual a 1 pode ser vista na Figura 7.16. Na expansão dessa árvore de falhas não foi utilizada nenhuma tabela de transição de estados.

Já a Figura 7.17, exibe a árvore de falhas para o caso do evento topo em que a saída 01 do componente "Pump" assume valor "0". Analisando a tabela de funções do compo-



Tabelas de Função

Tabela de Funções - Componente: Junction

Input 1	Input 2	Functionality Condition	Output 1
0	0		0
0	1		1
1	0		1
1	1		1

Tabela de Funções - Componente: Operator Human

Input 1	Functionality Condition	Output 1	Output 2
0	OK	0	0
1	OK	1	1
1	Failt to Detected Situation	0	0
0	Wrong Decision	1	1
1	Wrong Decision	2	2

Tabela de Funções - Componente: Pump

Input 1	Input 2	Functionality Condition	Output 1
0	null	null	0
null	0	null	0
null	null	Failed	0
1	1	Working	1

Tabela de Funções - Componente: Relay

Input 1	Input 2	Functionality Condition	Output 1
1	1	OK	1
0	null	null	0
1	null	Stuck Closed	1
null	null	Fail to Close	0
null	0	OK	0

Tabela de Funções - Componente: Switch

Input 1	State	Output 1
0	null	0
1	Close	1
null	Open	0

Tabela de Funções - Componente: Valve

Input 1	State	Output 1
0	null	0
null	Close	0
1	Open	1

Tabelas de Transição de Estados

Tabela de Transição de Estado - Componente: Switch

Input 2	Initial State	Functionality Condition	Final State
2	null	OK	Close
2	Open	Fail to Close	Open
1	Close	Fail to Open	Close
1	null	OK	Open
0	Open	null	Open
0	Close	null	Close

Tabela de Transição de Estado - Componente: Valve

Input 2	Initial State	Functionality Condition	Final State
2	Open	OK	Close
2	Open	Fail to Close	Open
1	Open	null	Open
1	Close	OK	Open
1	Close	Fail to Open	Close
2	Close	null	Close
0	Open	null	Open
0	Close	null	Close

Figura 7.15: Tabelas de funções para o exemplo da Figura 7.6.

nente “Pump”, para a saída 01 com valor “0”, temos que incluir uma porta OR. Uma das entradas dessa porta lógica é a entrada 01 assumir o valor “0”. Essa entrada corresponde à saída do componente “Valve”, que possui tabela de transição de estados. Para a saída

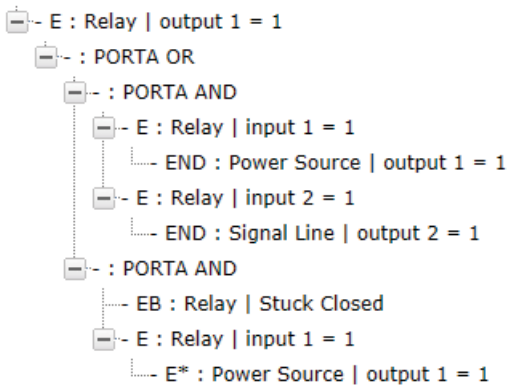


Figura 7.16: Árvore de falhas para o caso em que a saída 01 do "Relay" assume valor igual a "1".

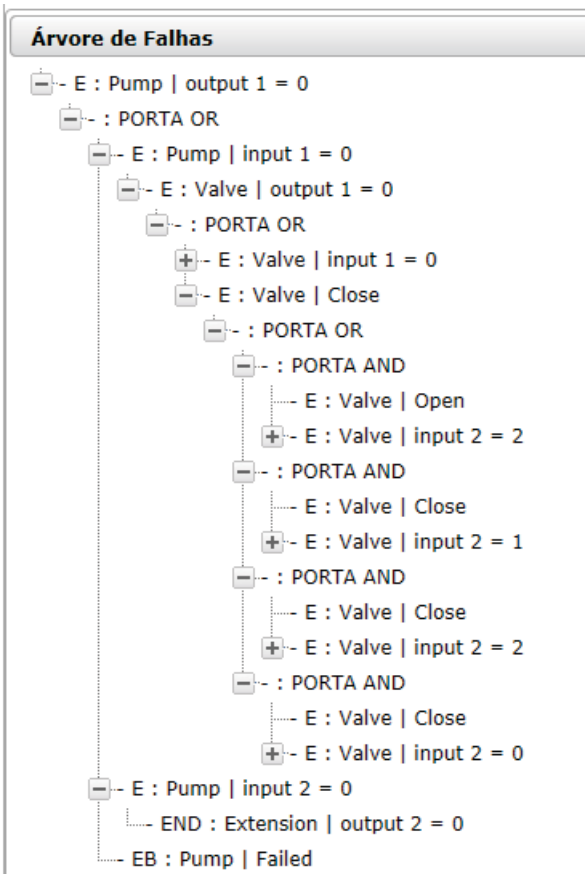


Figura 7.17: Árvore de falhas para o caso em que a saída 01 do componente "Pump" assume valor igual a "0".

da "Valve" assumir o valor "0", temos uma nova porta OR com as entradas entrada 01 da "Valve" como sendo "0" e o seu estado sendo "Close". A partir desse estado é que se analisa a tabela de transição de estados. Para o estado da "Valve" ser "Close", existem 04

linhas em sua tabela de transição de estados o que configura uma nova porta OR e cada entrada dessa porta corresponde às condições da entrada 02, estado inicial e condição de funcionamento de cada linha.

7.5.3 Exemplo 03: Integração com Ferramenta de Análise de Árvore de Falhas

A Figura 7.18 apresenta a ferramenta para análise de árvore de falhas desenvolvida no trabalho descrito em [Macedo et al. 2013]. A árvore de falhas a ser analisada deve ser criada no painel da lateral direito. Essa árvore pode ser criada manualmente através dos símbolos presentes na lateral esquerda ou ser importada através de um arquivo com extensão ".txt", que deve ser criado utilizando uma sintaxe pré-definida.

Na Figura 7.18 há um exemplo de arquivo a ser gerado para determinada árvore de falhas. Esse arquivo deve seguir o seguinte padrão:

- A descrição deve começar com o token STARTTREE e terminar com ENDTREE;
- Para se criar um evento simples usa-se a seguinte sintaxe: <basic> + <nome do evento> + <distribuição escolhida> + <parâmetro(s) dessa distribuição>;
- A criação de um evento que é clone de outro (ou seja, repetido) deve seguir a sintaxe: <basic> + < nome do evento > + <clone> + <nome do evento que é clonado>;
- A porta AND é descrita da seguinte forma: <and> + <nome da porta> + <nomes das suas entradas>;
- A porta OR é descrita de maneira similar a porta AND: <or> + <nome da porta> + <nomes das suas entradas>;
- O evento de topo da árvore é descrito por: <terminal> + <nome do terminal> + <tipo de analise a ser feita na FTA> + <intervalo de amostragem> + <ponto inicial da analise> + <ponto final da analise>.

Nesta tese, também realizou-se uma integração com a ferramenta apresentada em [Macedo et al. 2013]. Com isso, após gerar a árvore de falhas com base no algoritmo descrito anteriormente, gerou-se também um arquivo ".txt" no formato aceito pela ferramenta de análise de árvore de falhas.

Um dos exemplos para validar essa integração corresponde ao sistema de controle de nível presente na Figura 7.20 [M. Lampis 2009]. O sistema é formado por um tanque e uma bandeja. Existem dois sensores, $S1$ e $S2$, internos ao tanque que permitem analisar se o nível está adequado. Além disso, esses sensores enviam, respectivamente, informações a seus respectivos controladores ($C1$ e $C2$). Para a correta execução do processo, o valor do nível não deve ser menor que o estabelecido por $S1$ e nem maior que o estabelecido por $S2$. O controlador $C1$ é responsável por controlar a válvula de entrada ($V1$) enquanto que o controlador $C2$ é responsável pela válvula de saída de segurança ($V3$). Caso uma falha ocorra e o tanque transborde, a bandeja de segurança evitará que o líquido atinja outros componentes do sistema, a mesma está equipada com um sensor $SP1$ que indica se existe presença de água ou não. Por sua vez, a válvula de saída ($V2$) também está presente no sistema, porém esta é operada manualmente. Dutos numerados de $P1$ a $P6$ são os meios

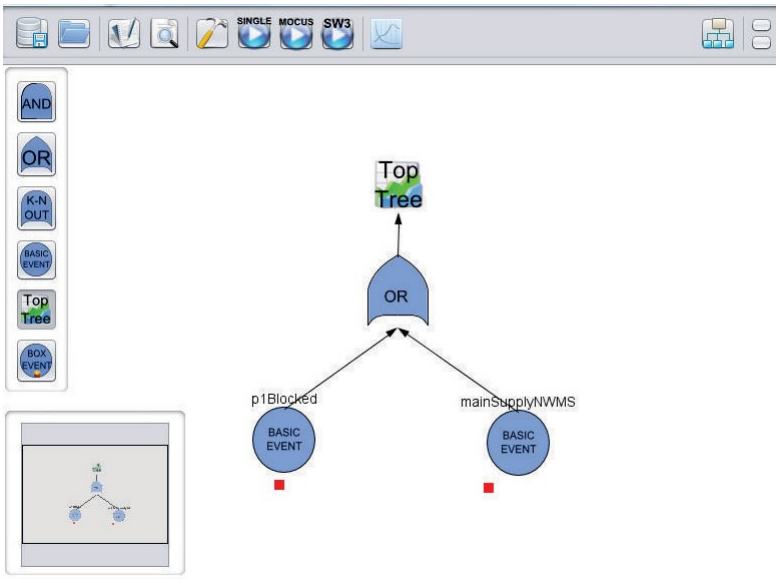


Figura 7.18: Ferramenta para análise de árvore de falhas

```
1  STARTTREE
2      basic C failureRate 0.2
3      basic A failureRate 0.1
4      basic B failureRate 0.1
5      basic AA clone A
6      and and2 C AA
7      and and1 A B
8      or or1 and2 and1
9      terminal terminal or1 reliability 1.0 0.0 10.0
10 ENDTREE
```

Figura 7.19: Exemplo de sintaxe do arquivo .txt para integração com a ferramenta.

por onde o líquido flui e estes também são alvos de falhas como rachaduras e obstruções. Neles estão embutidos três sensores de vazão $VF1$, $VF2$ e $VF3$.

Em condições normais, a válvula $V2$ encontra-se sempre aberta deixando o líquido sair, enquanto que a válvula $V1$ fica aberta para tentar substituir o que vazou da $V2$. Assim, o nível do tanque se mantém constante. A válvula $V3$ permanecerá fechada a não ser que o nível do tanque alcance um valor crítico.

A Figura 7.21 apresenta o esquema em blocos do sistema de controle de nível. Esse sistema também teve instância criada na OntoConf, incluindo suas tabelas de funções e de transição de estados.

Para a planta do sistema de controle de nível também foi criada uma instância na OntoConf, onde todo o esquema de blocos foi mapeado e também as tabelas de funções e de transição de estado dos componentes. Ao executar a aplicação desenvolvida nesta tese, é possível visualizar as tabelas utilizas pelo algoritmo para geração automática de árvores de falhas, conforme Figuras 7.22 e 7.23.

Um exemplo de árvore de falhas gerada automaticamente está presente na Figura 7.24.

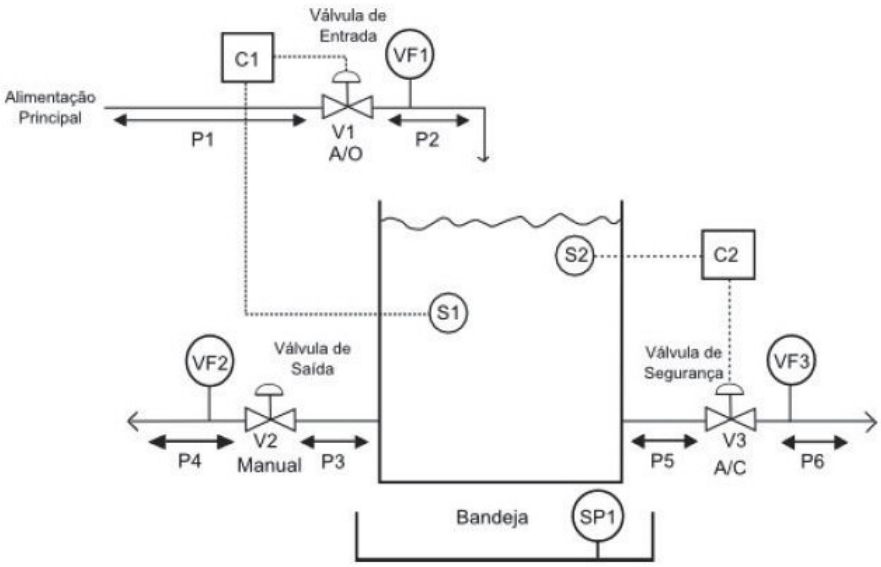


Figura 7.20: Sistema de controle de nível.

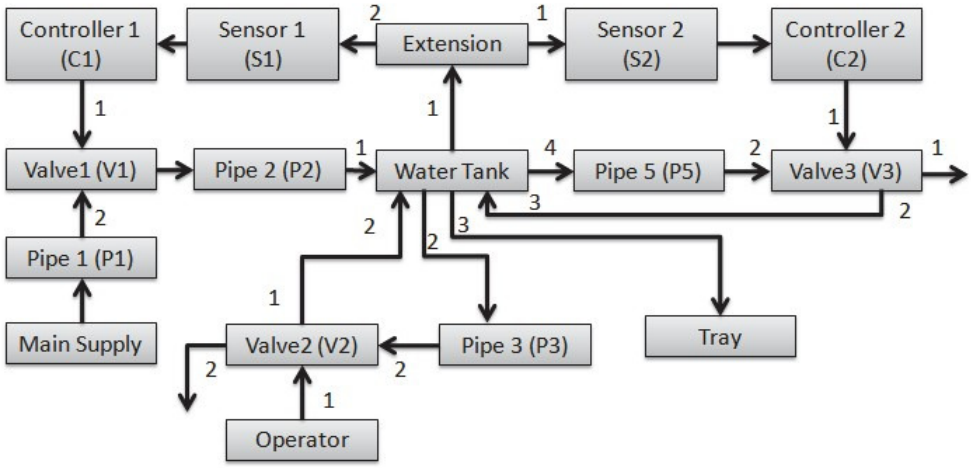


Figura 7.21: Esquema do sistema de controle de nível.

Nesse caso, o evento topo escolhido corresponde a situação do tanque de água transbordar, fazendo com que a bandeja presente na planta receba água.

Outro exemplo pode ser visto na Figura 7.25 e representa a ausência de fluxo no duto 01.

Tendo a árvore de falhas gerada através da aplicação desenvolvida nesta tese, é possível gerar o arquivo com extensão ".txt" que é aceito pela ferramenta descrita [Macedo et al. 2013] para análise de árvore de falhas. Para o exemplo da 7.25, foi gerado o arquivo presente na Figura 7.26. É importante ressaltar que os dados de taxas de falhas e de taxas de reparo para cada condição de funcionamento que representa uma falha também foram extraídos das instância da ontologia. Ao importá-lo para ferramenta de análise de falhas,

Tabelas de Função

Tabela de Funções - Componente: Controller 1

Input 1	Functionality Condition	Output 1
Below	OK	1
Above	OK	0
null	Failed Low	1
null	Failed High	0

Tabela de Funções - Componente: Controller 2

Input 1	Functionality Condition	Output 1
Above	OK	1
Below	OK	0
null	Failed High	1
null	Failed Low	0

Tabela de Funções - Componente: Extension

Input 1	Functionality Condition	Output 1	Output 2
Normal	OK	Normal	Normal
Low Level	OK	Low Level	Low Level
High Level	OK	High Level	High Level

Tabela de Funções - Componente: Operator

Input 1	Functionality Condition	Output 1
1	OK	1
0	OK	0

Tabela de Funções - Componente: Sensor 1

Input 1	Functionality Condition	Output 1
Low Level	OK	Below
Normal	OK	Above
High Level	OK	Above
null	Failed Low	Below
null	Failed High	Above

Tabela de Funções - Componente: Sensor 2

Input 1	Functionality Condition	Output 1
Low Level	OK	Below
Normal	OK	Below
High Level	OK	Above
null	Failed Low	Below
null	Failed High	Above

Tabela de Funções - Componente: Pipe 1

Input 1	Functionality Condition	Output 1
Flow (F)	OK	Flow (F)
No Flow (NF)	null	No Flow (NF)
null	Blocked	No Flow (NF)
Flow (F)	Fractured	Flow (F)

Tabela de Funções - Componente: Pipe 2

Input 1	Functionality Condition	Output 1
Flow (F)	OK	Flow (F)
No Flow (NF)	null	No Flow (NF)
null	Blocked	No Flow (NF)
Flow (F)	Fractured	Flow (F)

Tabela de Funções - Componente: Pipe 3

Input 1	Functionality Condition	Output 1
Flow (F)	OK	Flow (F)
No Flow (NF)	null	No Flow (NF)
null	Blocked	No Flow (NF)
Flow (F)	Fractured	Flow (F)

Tabela de Funções - Componente: Pipe 5

Input 1	Functionality Condition	Output 1
Flow (F)	OK	Flow (F)
No Flow (NF)	null	No Flow (NF)
null	Blocked	No Flow (NF)
Flow (F)	Fractured	Flow (F)

Tabela de Funções - Componente: Tray

Input 1	Functionality Condition
Water	OK
No Water	OK

Tabela de Funções - Componente: Main Supply

Functionality Condition	Output 1
OK	Flow (F)
NWMS (No water from main stream)	No Flow (NF)

Figura 7.22: Tabelas de Funções - Controle de nível - Parte I.

é gerada a árvore exibida na Figura 7.27.

Tabela de Funções - Componente: Water Tank

Input 1	Input 2	Input 3	Functionality Condition	Output 1	Output 2	Output 3	Output 4
Flow (F)	Flow (F)	No Flow (NF)	OK	Normal	Flow (F)	No Water	No Flow (NF)
Flow (F)	Flow (F)	No Flow (NF)	Water Tank Leaking	Normal	Flow (F)	Water	No Flow (NF)
Flow (F)	No Flow (NF)	Flow (F)	Water Tank Leaking	Low Level	No Flow (NF)	Water	Flow (F)
Flow (F)	No Flow (NF)	No Flow (NF)	Water Tank Leaking	High Level	No Flow (NF)	Water	No Flow (NF)
No Flow (NF)	null	null	Water Tank Leaking	Low Level	No Flow (NF)	No Water	No Flow (NF)
Flow (F)	No Flow (NF)	No Flow (NF)	OK	High Level	No Flow (NF)	Water	No Flow (NF)
No Flow (NF)	Flow (F)	No Flow (NF)	OK	Low Level	Flow (F)	No Water	No Flow (NF)
No Flow (NF)	No Flow (NF)	No Flow (NF)	OK	Low Level	No Flow (NF)	No Water	No Flow (NF)
null	Flow (F)	Flow (F)	OK	Low Level	Flow (F)	No Water	Flow (F)
null	No Flow (NF)	Flow (F)	OK	Low Level	No Flow (NF)	No Water	Flow (F)
Flow (F)	null	null	Water Tank Ruptured	Low Level	No Flow (NF)	Water	No Flow (NF)
No Flow (NF)	null	null	Water Tank Ruptured	Low Level	No Flow (NF)	No Water	No Flow (NF)
Flow (F)	Flow (F)	Flow (F)	Water Tank Leaking	Low Level	Flow (F)	Water	Flow (F)

Tabelas de Transição de Estados

Tabela de Funções - Componente: Valve 1

Input 2	State	Output 1
No Flow (NF)	null	No Flow (NF)
Flow (F)	Open	Flow (F)
null	Close	No Flow (NF)

Tabela de Funções - Componente: Valve 2

Input 2	State	Output 1	Output 2
No Flow (NF)	null	No Flow (NF)	No Flow (NF)
Flow (F)	Open	Flow (F)	Flow (F)
null	Close	No Flow (NF)	No Flow (NF)

Tabela de Funções - Componente: Valve 3

Input 2	State	Output 1	Output 2
No Flow (NF)	null	No Flow (NF)	No Flow (NF)
Flow (F)	Open	Flow (F)	Flow (F)
null	Close	No Flow (NF)	No Flow (NF)

Tabela de Transição de Estado - Componente: Valve 1

Input 1	Initial State	Functionality Condition	Final State
1	null	OK	Open
0	null	OK	Close
0	Open	Fail to Close	Open
1	Close	Fail to Open	Close

Tabela de Transição de Estado - Componente: Valve 2

Input 1	Initial State	Functionality Condition	Final State
1	null	OK	Open
0	null	OK	Close
0	Open	Fail to Close	Open
1	Close	Fail to Open	Close

Tabela de Transição de Estado - Componente: Valve 3

Input 1	Initial State	Functionality Condition	Final State
1	null	OK	Open
0	null	OK	Close
0	Open	Fail to Close	Open
1	Close	Fail to Open	Close

Figura 7.23: Tabelas de Funções e de Transição de Estados - Controle de nível - Parte II.

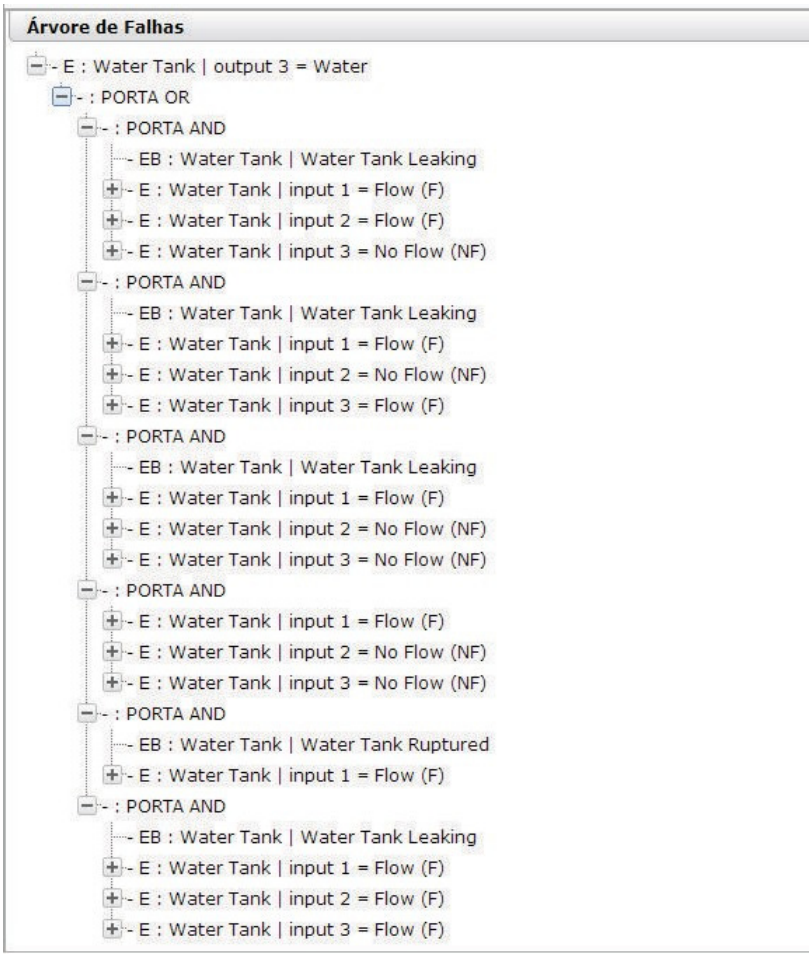


Figura 7.24: Árvore de falhas para o caso em que a saída 03 do “Tank” assume valor igual a “Water”.

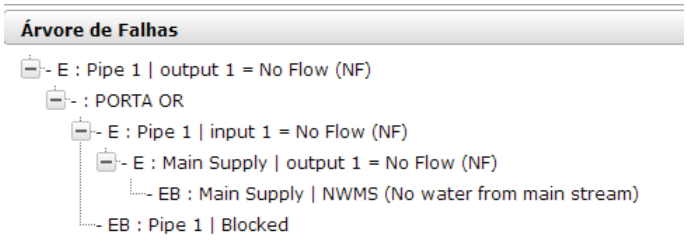


Figura 7.25: Árvore de falhas para o caso em que a saída do “Pipe 1” assume valor igual a “No Flow”.

```
STARTMODEL
    basic p1Blocked repairFailireRate 1.0 0.01
    basic mainSupplyNWMS repairFailireRate 0.2 0.01
    or  p1Blocked mainSupplyNWMS
terminal conf_03   reability 1.0 0.0 100.0
ENDTREE
```

Figura 7.26: Arquivo .txt para integração com ferramenta para árvore de falhas.

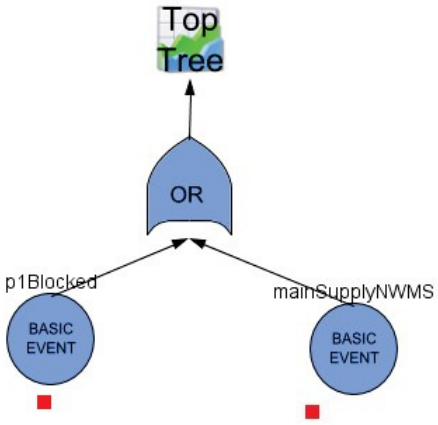


Figura 7.27: Árvore de falhas gerada para ferramenta de análise.

As Figuras 7.28 e 7.29 apresentam, respectivamente os gráficos para análise de confiabilidade e da disponibilidade em três situações:

- Gráfico azul: taxa de reparo para duto 1 bloqueado é igual a 1, taxa de reparo para falha no suprimento de água é igual a 0,2, taxa de falha para duto 1 bloqueado é igual a 0,01, taxa de falha no suprimento de água é igual a 0,01;
- Gráfico vermelho: taxa de reparo para duto 1 bloqueado é igual a 1, taxa de reparo para falha no suprimento de água é igual a 0,2, taxa de falha para duto 1 bloqueado é igual a 0,005, taxa de falha no suprimento de água é igual a 0,005;
- Gráfico verde: taxa de reparo para duto 1 bloqueado é igual a 1, taxa de reparo para falha no suprimento de água é igual a 0,2, taxa de falha para duto 1 bloqueado é igual a 0,001, taxa de falha no suprimento de água é igual a 0,001;

No gráfico da Figura 7.28, observa-se que a medida que a taxa de falha dos componentes diminui, aumenta-se a confiabilidade do sistema. Por exemplo, no instante t = 100 horas, considerando a taxa de falhas dos componentes de 0,01, a confiabilidade do sistema fica em torno de 13 % (gráfico azul); já para a taxa de falha de 0,005, a confiabilidade fica em torno de 35 % (gráfico vermelho); e para a taxa de falha de 0,001, menor taxa, a

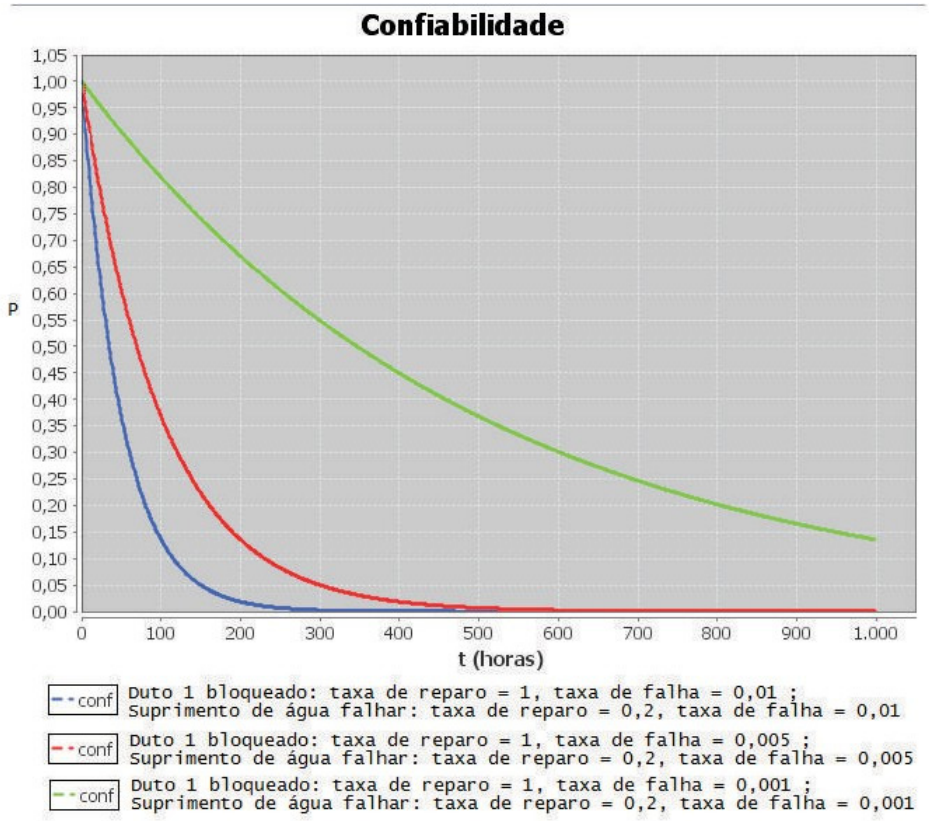


Figura 7.28: Gráfico de confiablidade gerado pela ferramenta de análise de árvore de falhas.

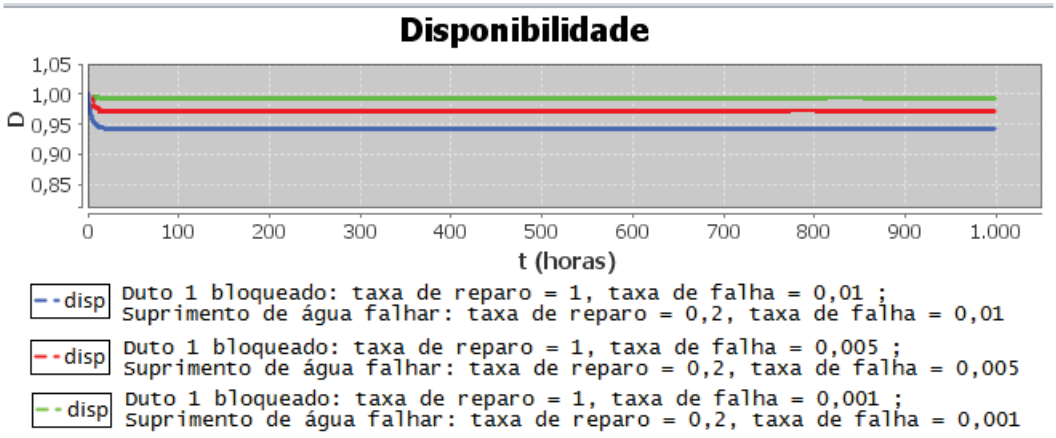


Figura 7.29: Gráfico de disponibilidade gerado pela ferramenta de análise de árvore de falhas.

confiabilidade fica em torno de 82 % (gráfico verde). Observa-se tambem que em torno do instante t = 450 horas, a confiabilidade, considerando as taxas de falhas 0,01 (gráfico azul) e 0,005 (gráfico vermelho) já são próximas de zero, enquanto se a taxa de falha for

0,001, ela ainda está em torno de 40 % (gráfico verde)

Já no gráfico da Figura 7.29, considerando que para cada gráfico as taxas de reparo foram as mesmas, observa-se a influência da taxa de falha na disponibilidade do sistema. Quanto menor a taxa de falha, mas disponível é o sistema.

7.6 Considerações Finais

Nesta capítulo foi apresentada uma nova ontologia, a OntoConf, aplicada ao domínio de confiabilidade de processos industriais. A OntoConf, assim como a OntoEcon, importou os conceitos e propriedades presentes na OntoAuto, acrescentando os seus próprios conceitos e propriedades específicas. A metodologia para geração automática de árvore de falhas apresentada em [Majdara & Wakabayashi 2009] foi modelada na OntoConf.

No trabalho de [Majdara & Wakabayashi 2009] a descrição do algoritmo foi apresentada, porém a aplicação para gerar árvores de falhas automaticamente não foi implementada. Nesta tese, o desenvolvimento da aplicação para geração automática de árvore de falhas só foi possível devido aos relacionamentos entre entradas e saídas dos componentes modelados na OntoAuto e da representação das tabelas de funções e de transições de estados específicas da OntoConf.

Por fim, também foi feita uma integração com a ferramenta para análise de árvore de falhas desenvolvida em [Macedo et al. 2013], onde o resultado da aplicação desenvolvida usando ontologias alimenta essa ferramenta.

Capítulo 8

Conclusões e Perspectivas Futuras

A principal hipótese desta tese é que a representação do conhecimento de forma estruturada na área de processos industriais facilita o desenvolvimento de aplicações de automação avançada. Mais especificamente, defende-se que a estruturação de conhecimento baseada em ontologias permite o desenvolvimento de ferramentas computacionais flexíveis, que podem ser reutilizadas ou adaptadas para o uso em diversos processos industriais, mesmo que eles apresentem suas próprias peculiaridades. Isto é possível porque a ontologia é a base consistente a partir da qual são desenvolvidas ou adaptadas diversas aplicações. Além disso, o uso de ontologias permite a extensibilidade, inclusão de informações mais estruturadas, possibilidade de incluir regras e fazer inferências automaticamente.

Então, com o objetivo de verificar essa hipótese, nesta tese desenvolveu-se uma ontologia de domínio denominada de OntoAuto para representação dos conceitos principais envolvidos nos processos industriais. Com essa ontologia, pode-se representar como os componentes das plantas industriais se relacionam e a sequência do fluxo dos materiais, informações, energia, dentre outros relacionamentos.

Objetivando validar a OntoAuto, dois processos importantes foram instanciados e analisados: unidade de tratamento DEA e fornos industriais. Observa-se que todos os seus componentes foram modelados e suas relações através das diversas propriedades. Os principais elementos de uma planta industrial foram representados através da classe "Component" e suas subclasses. As classes "Input" e "Output" e as propriedades "componentInputOutput", "hasAssociateInput", "hasAssociateOutput" foram fundamentais para a modelagem da sequência do fluxo na planta e permitiram o desenvolvimento das aplicações de correção semântica de alarmes e confiabilidade.

A partir da OntoAuto, foi possível desenvolver uma aplicação voltada para a área de gerência de alarmes. A aplicação permitiu realizar uma correlação semântica dos alarmes. Essa abordagem baseada em ontologia incluiu aspectos semânticos no procedimento de obtenção da correlação entre alarmes, sem considerar apenas a temporalidade da ativação das ocorrências de alarmes. Assim, com essa abordagem semântica foi possível um melhor auxílio de procedimentos de diagnósticos de anomalias em processos industriais, pois alarmes associados com equipamentos fisicamente desacoplados no processo industrial podem apresentar alta correlação temporal, por motivos alheios à causalidade dos eventos que descrevem a dinâmica do processo.

A aplicação de correlação semântica de alarmes foi executada com base em dados

reais de uma planta de unidade de tratamento DEA, mas a mesma pode ser reaproveitada em outros processos, bastando apenas criar a instância do processo na OntoAuto e criar a comunicação com a base de dados desse novo processo. A análise tradicional, baseada na simples correlação temporal entre alarmes, pode levar a conclusões equivocadas, quando, por exemplo, há uma má configuração de alarmes. A adição de informações vindas da estruturação do conhecimento permite melhorar as análises de operação de processos industriais. No caso específico apresentado, referente à correlação de alarmes, a melhoria nos resultados da correlação de alarmes foi proporcionada pela ordenação física causal dos alarmes e determinação do nível de vizinhança entre eles, que só foi possível através da estruturação do conhecimento da ontologia.

A OntoAuto também foi aplicada na área de projetos para avaliação econômica de plantas industriais. Nesse caso, identificou-se que apenas os conceitos presentes na OntoAuto não seriam suficientes. Dessa forma, foi criada uma nova ontologia de aplicação, a OntoEcon, que importa os conceitos e propriedades da OntoAuto e é acrescida de novos conceitos e propriedades mais específicos à essa área.

No contexto de projeto de processos industriais, a análise de custos é imprescindível, pois é a partir dela que se define a viabilidade econômica do processo. O processo apresentado para validar a OntoEcon desenvolvida foi o de produção de lauril éter sulfato de sódio. Considerando que o conhecimento está estruturado, a aplicação desenvolvida para avaliação econômica, baseada no método *Venture Profit*, pode-se realizar a análise econômica das várias opções de processos para se obter o mesmo resultado de produto, permitindo a comparação entre eles. Ressalta-se que a OntoEcon é abrangente o suficiente para ser utilizada na análise econômica de outros processos industriais.

Por fim, com o objetivo de validar a hipótese central desta tese, criou-se uma nova ontologia a partir da OntoAuto para o contexto de análise de confiabilidade de processos industriais, que é uma área bastante importante por tratar da criticidade da contenção de falhas em processos industriais. Mais uma vez, a OntoAuto teve seus conceitos e propriedades importados por uma nova ontologia de aplicação, incorporando novos conceitos e propriedades específicos da área confiabilidade de sistemas. A nova ontologia foi denominada de OntoConf.

Na OntoConf foram modeladas entidades com base no algoritmo de geração automática de árvore de falhas apresentado em [Majdara & Wakabayashi 2009]. No trabalho de [Majdara & Wakabayashi 2009] a descrição do algoritmo foi apresentada, porém a aplicação para gerar árvores de falhas automaticamente não foi implementada. A geração automática foi possível diante dos relacionamentos entre entradas e saídas dos componentes modelados na OntoAuto e da representação das tabelas de funções e de transições de estados específicas da OntoConf.

É importante ressaltar que a geração de árvore de falhas de forma manual é muito custosa. Assim, o fato de conseguir desenvolver uma aplicação que pode ser usada por vários processos apresenta um ganho grande em termos de esforço e tempo. Além disso, o fato da aplicação desenvolvida poder se comunicar com uma ferramenta já existente para análise de árvore de falhas acelera esse processo.

Uma das facilidades no uso de ontologias para o desenvolvimento de aplicações de automação avançada foi o uso da tecnologia Java e dos *frameworks* relacionados, como

o JENA. O JENA permite carregar a ontologia para a memória da aplicação e a partir da linguagem de consulta SPARQL realizar consultas sobre o domínio das instâncias da ontologia.

Por fim, com esta tese, foi possível mostrar a vantagem da estruturação do conhecimento na área de automação. Área essa, onde o conhecimento é pouco estruturado e o desenvolvimento de aplicações usualmente não é uma tarefa fácil, pois demanda muito conhecimento específico do funcionamento do processo em questão. Além disso, em geral para processos industriais diferentes são desenvolvidas aplicações diferentes para o mesmo fim. Dessa forma, com a estruturação do conhecimento permite em última análise o reuso de soluções.

As principais contribuições desta tese são listadas a seguir:

- Representação do conhecimento no domínio dos processos industriais de forma unificada através da ontologia de domínio denominada OntoAuto.
- Possibilidade de criação de novas ontologias de aplicação a partir da ontologia OntoAuto, abrangendo várias áreas da automação de insdustrial.
- Criação de aplicações em áreas distintas que podem ser reaproveitadas por vários processos.
- Desenvolvimento de uma aplicação para gerência de alarmes, melhorando o processo de correlação de alarmes via análise semântica.
- Desenvolvimento de uma aplicação para avaliação econômica de processos, área fundamental no projeto do processo que pode decidir a viabilidade ou não do mesmo.
- Desenvolvimento de uma aplicação para geração automática de árvore de falhas de forma automatizada a partir da modelagem da planta através da OntoAuto e sua extensão a OntoConf, o que permite a análise de confiabilidade e disponibilidade do processo.

8.1.1 Perspectivas Futuras

Através do trabalho desenvolvido nesta tese, visualiza-se as seguintes perspectivas futuras:

- Criação de novas ontologias de aplicações voltadas a outras áreas, como por exemplo para a área de diagnóstico de falhas.
- Criação de ontologia para modelar árvores de falhas, permitindo o desenvolvimento de aplicações de análise de árvore de falhas baseadas em conhecimento;
- Detalhamentos de regras das ontologias desenvolvidas através da linguagem SWRL (*Semantic Web Rule Language*);
- Expansão da aplicação de correlação semântica de alarmes para incorporar novas formas de filtragem, como por exemplo, considerar apenas alarmes de determinados componentes;
- Desenvolvimento de ferramenta que permita comparar avaliações econômicas de várias configurações do mesmo processo;
- Criação de uma ferramenta de alto nível para criar instâncias da OntoAuto, OntoEcon e OntoConf de forma mais amigável para o usuário;

- Realização de experimentos para avaliar o custo e benefícios do uso de ontologias na área de automação industrial por parte dos especialistas dos processos;
- Simulação de construções de ontologias com erros para avaliar impactos no processo.

8.1.2 Publicações Realizadas

- LIMA, R., G, F. ; LEITAO, G. ; Oliveira, Luiz A. H. G. ; BEZERRA, V. . Economic Evaluation of Industrial Processes with Ontology. In: 11th IEEE International Conference on Networking, Sensing and Control, 2014, Miami. IEEE International Conference on Networking, Sensing and Control, 2014.
- LIMA, R., G, F. ; LEITAO, G. ; Oliveira, Luiz A. H. G. ; MELO, J. D. ; Neto, Adrião D. D. . Semantic Alarm Correlation Based on Ontologies. In: Emerging Technologies e Factory Automation - ETFA, 2013, Cagliari. ETFA 2013, 2013.
- LIMA, R., G, F. ; LEITAO, G. ; Oliveira, Luiz A. H. G. ; MELO, J. D. ; Neto, Adrião D. D. . CORRELAÇÃO SEMÂNTICA DE ALARMES UTILIZANDO ONTOLOGIA. In: Simpósio Brasileiro de Automação Inteligente - SBAI, 2013, Fortaleza. SBAI 2013, 2013.

Referências Bibliográficas

A. Avizienis, J. C. Laprie, B. Randell C. Landwehr (2004), Basic concepts and taxonomy of dependable and secure computing, pp. 11–33.

A. Bernaras, I.Laresgoiti, J. Corera & N. Bartolomé (1996), An ontology for fault diagno- sis in electrical networks, *em* 'Proceedings of the Intelligent Systema Applications to Power System', pp. 199–203.

Abele, L., C. Legat, S. Grimm & A.W. Muller (2013), Ontology-based validation of plant models, *em* 'Industrial Informatics (INDIN), 2013 11th IEEE International Conference on', pp. 236–241.

Anthony A.R. Diniz, Rodrigo Silva, Adrião Duarte Dória Neto & Jorge D. de Melo (2012), Ontology for industrial petrochemical processes: Case study of a dea pro- cess.

Arnold, K. & M. Stewart (1999), *Surface Production Operations*, Vol. 2, Elseiver.

Bill Swartout, Ramesh Patil, Kevin Knight & Tom Russ (1996), 'Toward distributed use of large-scale ontologies', *USC/Information Sciences Institute* .

Blackburn, S. & D. Marcondes (1997), *Dicionário oxford de filosofia. Tradução D. Murcho et al. Rio de Janeiro*, Jorge Zahar.

Boyer, Stuart A. (2010), *Supervisory Control And Data Acquisition*.

Campos, Vicente Falcone (1992), *TQC: Controle de Qualidade Total (no estilo japonês)*, Fundação Christiano Ottoni, Escola de Engenharia da UFMG.

Carlan, Eliana (2010), Sistemas de organização do conhecimento: uma reflexão no con- texto da ciência da informação, Dissertação de mestrado, Universidade de Brasília.

Cavalcanti, C.R. (1978), *Indexação e tesauro: metodologia e técnicas*.

Contini, S. (1995), A new hybrid method for fault tree analysis, Vol. 49, pp. 13–23.

da Silva, Ivanovitch Medeiros Dantas (2013), Uma Metodologia para Modelagem e Ava- liação da Dependabilidade de Redes Industriais Sem Fio, Tese de doutorado, Universidade Federal do Rio Grande do Norte.

Dawe, R. A. (2002), *Modern Petroleum Technology*, Vol. 1, Institute of Petroleum.

de Oliveira, Leandro, Sandra Aluísio & Gladis Almeida (2007), Ontoeditor uma ferra- menta web para edição de ontologias, *em* 'V Workshop em Tecnologia da Informa- ção e da Linguagem Humana'.

Dunn, Willian R. (2002), *Pratical Design ofSafety-Critical Computer Systems*, Reliability Press.

Eng (1999), *Alarm Systems A Guide to Design, Management and Procuremen*, 191ª edição.

Filho, Constantino Seixa (1993), *Programação concorrente em ambiente Windows - Uma visão de automação*.

Genesereth, M. E. & R. Fikes (1992), Knowledge interchange format. reference manual. technical report logic-921., Relatório Técnico 3, Computer Science Department, Stanford University.

Gómez-Pérez, Asunción (2000), Evaluation of ontologies, *em* 'International Journal of Intelligent Systems', Vol. 16.

Gómez-Pérez, Asunción & M. D. Rojas (1999), Ontological reengineering and reuse, *em* '11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'99), Lecture Notes in Artificial Intelligence', Vol. 1621, p. 139156.

Grosso, William (1999), Formal aspects of protégé, *em* 'Fourth International Protégé Workshop'.

Gruber, Thomas R. (1993), A translation approach to portable ontology specifications, *em* 'Knowledge Engineering Review. Computer Science Department. Stanford Univer- sity.'.

Guarino, N. (1997), 'Understanding, building and using ontologies.', *International Journal of Human- Computer Studies* 46, 293–310.

Gustavo Leitão, A. Pifer, Luiz Affonso Guedes K. Saito & L. Aquinoo (2008), Petroche- mical plants alarms analysis and optimization system, *em* 'Rio Oil and Gas'.

Haav, M. H. & T. L. Lubi (2001), A survey of concept-based information retrieval tools on the web, *em* 'East European Conference ADBI'.

Habibi, Eddie & Bill Hollifield (2006), Alarm systems greatly affect offshore facilities amid high oil prices, World Oil Magazine.

Henley, E. J. & H. Kumamoto (1981), *Reliability Engineering and Risk Assessment*, Prentice Hall, Inc.

Hjelm, Johan (2001), *Creating the Semantic Web with RDF*, John Wiley and Sons; Pap/Cdr edition.

Jan Morbach, Aidong Yang & Wolfgang Marquardt (2007), Ontocape: A large-scale on- tology for chemical process engineering, *em* 'ScienceDirect. Engineering Applications of Artificial Intelligence', Vol. 20, pp. 147 – 161.

Jan Morbach, Andreas Wiesner & Wolfgang Marquardt (2009), Ontocape: A (re)usable ontology for computer-aided process engineering, *em* 'Computers and Chemical En- gineering', Vol. 33, pp. 1546 – 1556.

John Hebeler, Matthew Fisher, Ryan Blace & Andrew Perez-Lopes (2009), *Semantic Web Programming*, Wiley Publishing, Inc.

JU. Kietz, A. Maedche & R. Volz (2000), A method for semi-automatic ontology acqui- sition from a corporate intranet, *em* 'EKAW'00 Workshop on Ontologies and Texts, CEUR Workshop Proceedings, Juan-Les-Pins', Vol. 51.

Kim, YounHee, ByungGon Kim & HaeChull Lim (2006), The index organizations for rdf and rdf schema, *em* 'Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference', Vol. 3, pp. 4 pp.–1874.

Küçük Dilek, O. Salor, T. Inan & I. Cadirci (2008), Building an ontology for flexible power quality querying, *em* 'Computer and Information Sciences, ISCIS.', Vol. 33, pp. 1–6.

Klein, M. & D. Fensel (2001), Ontology versioning on the semantic web, *em* 'First Inter- national Semantic Web Workshop (SWWS01)'.

Knublauch, Holger, Ray W. Fergerson, Natalya F. Noy & Mark A. Musen (2004), The protégé owl plugin: An open development environment for semantic web applications, *em* 'The Semantic Web Conference ISWC 2004'.

Kupcik, M., M. Sir & P. Fiedler (2012), Interoperability in diagnostic systems, *em* 'Car- pathian Control Conference (ICCC), 2012 13th International', pp. 408–412.

Laallam, F. Z. & M. Sellami (2007), Gas turbine ontology for industrial processes, *em* 'Journal of Computer Science', pp. 113–118.

Leitão, Gustavo Bezerra Paz (2008), Algoritmos para análise de alarmes em processos petroquímicos, Dissertação de mestrado, Universidade Federal do Rio Grande do Norte.

Lenat, Douglas B. & R.V. Guha (1989), *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, 1st[a] edição, Addison-Wesley Longman Publishing Co., Boston, MA, USA.

Li, Guoqi (2012), Ontology-based reuse of failure modes for fmea: Methodology and tool, International Symposium on Software Reliability Engineering Workshops.

Limnios, N. (2007), *Fault trees, Control systems, robotics and manufacturing series*.

Lindoso, A. N. (2006), Uma metodologia baseada em ontologias para a engenharia de aplicações multiagentes, Dissertação de mestrado, UFMA.

M. Klein, D. Fensel, F. van Harmelen & I. Horrocks (2000), The relation between on- tologies and schema-languages: Translating oil-specifications in xml-schema, *em* 'Workshop on Applications of Ontologies and Problem-solving Methods, 14th Eu- ropean Conference on Artificial Intelligence ECAI'00'.

M. Lampis, and J. D. Andrews (2009), Bayesian belief networks for system fault diagnostics, Vol. 25, p. 409 426.

Macedo, Daniel, Ivanovitch Silva & Luiz Affonso Guedes (2013), Uma ferramenta para análise de dependabilidade de processos industriais, *em* 'Simpósio Brasileiro de Au- tomação Inteligente , 2013. SBAI 2013.'.

Majdara, Aref & Toshio Wakabayashi (2009), Component-based modeling of systems for automated fault tree generation, p. 1076 1086.

Mariano Fernández López, Asunción Gómez-Pérez, A. Pazos-Sierra & J. Pazos-Sierra (1999), Building a chemical ontology using methontology and the ontology design environment, *em* 'IEEE Intelligent Systems and their applications', Vol. 4, pp. 37– 46.

Martinez, Ana (2004), Las categorías o facetas fundamentales: una metodología para el diseño de taxonomías corporativas de sitios web argentinos, *em* 'Ci. Inf.', Vol. 33, pp. 106–111.

Matthew Horridge, Holger Knublauch, Alan Rector Robert Stevens & Chris Wroe (2004), *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools*, 1.0ª edição.

McBride, Brian (2009), *An Introduction to RDF and the Jena RDF API*.

Meisen, A. & M. L. Kennard (1982), 'Dea degradation mechanism', *Hydrocarbon Process* 61:10, 105–109.

Moore, B. J. & B. C. Moore (1943), *Industrial Kiln and Oven*, United States Patent Office.

Muñoz, E. & A. Espuña (2010), Towards an ontological infrastructure for chemical batch process management, *em* 'Computers and Chemical Engineering', Vol. 34, pp. 668
– 682.

Novak, P. & R. Sindelar (2013), Ontology-based industrial plant description supporting simulation model design and maintenance, *em* 'Industrial Electronics Society, IE- CON 2013 - 39th Annual Conference of the IEEE', pp. 6866–6871.

Noy, Natalya Fridman & Deborah L. McGuinness (2001), Ontology development 101 : a guide to creating your first ontology, *em* 'Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, Stanford Medical Informatics Technical Report SMI'.

O. Aizpurúa, R. Galán & A. Jiménez (2008), A new cognitive-based massive alarm management system in electrical power administration, *em* 'Proceedings of the 7th International Caribbean Conference on Devices, Circuits and Systems', pp. 28–30.

Olawande Daramola, Tor Stalhane, THomas Moser & Stefan Biffl (2011), A conceptual framework for semantic case-based safety analysis, Emerging Technologies e Fac- tory Automation - ETFA.

Oscar Corcho, Mariano Fernández López & Asunción Gómez-Pérez (2003), Methodolo- gies, tools and languages for building ontologies. where is their meeting point?, *em* 'Data and Knowledge Engineering', Vol. 46, pp. 41–64.

P. Karp, V. Chauhdri & J. Thomere (1999), Xol: an xml-based ontology exchange language, Relatório técnico, Artificial Intelligence Center SRI International , Menlo Park, CA, Technical Report.

Perlingeiro, C. A. (2005), *Engenharia de processos: Análise, simulação, otimização e síntese de processos químicos*, Blucher.

Pinto, P. F. B. & G. L. Paula (2009), Aplicações práticas de gerenciamento de alarmes em sistemas scada, Congresso Rio Automação.

Quintão, H. C. M. (2008), Especificação de um sistema multiagente de recomendação de ações em caso de falhas de sistema de automação e controle industriais, Dissertação de mestrado, UFMA.

R. Du, Y. Li, F. Shang & Y. Wu (2010), Study on ontology-based knowledge construc- tion of petroleum exploitation domain, *em* 'International Conference on Artificial Intelligence and Computational Intelligenc', Vol. 2, pp. 42–46.

R. Mizoguchi, J. Vanwelkenhuysen & M. Ikeda (1994), Task ontology for reuse of pro- blem solving knowledge, *em* 'ECAI94', IOS Press, Amsterdam, pp. 46–59.

R. Turton, R, Bailie W. B. Whiting & J. A. Shaeiwitz (2003), *Analysis, Synthesis and Design of Chemical Process*, Prentice Hall.

Rausand, Marvin & Arnljot Hsyland (2004), *System reliability theory: models, statistical methods, and applications*, John Wiley e Sons, Inc., Publication.

Reynolds, Dave (2009), *Jena 2 Inference support*.

Ricardo Falbo, Credine Menezes & Ana Regina Rocha (1998), A systematic approach for building ontologie, *em* 'Ibero-American Conference on Artificial Intelligence'.

Rudd, D. F. & C. C. Watson (1968), *Strategy of Process Engeneering*, J. Willey.

S. Natarajan, K. Ghosh & R. Srinivasan (2012), An ontology for distributed process supervision of large-scale chemical plants, *em* 'Computers and Chemical Engineering', pp. 124–140.

S. Staab, H.P. Schnurr, R. Studer & Y. Sure (2001), Knowledge processes and ontologies, *em* 'IEEE Intelligent Systems', Vol. 16.

Sachs, Eliza (2006), *Getting Started with Protege-Frames*, The Protege Project.

Sakurada, Eduardo Yuji (2001), As técnicas de análise de modos de falhas e seus efeitos e análise de Árvore de falhas no desenvolvimento e na avaliação de produtos, Dissertação de mestrado, Universidade Federal de Santa Catarina.

Sean Bechhofer, Ian Horrocks, Carole Goble & Robert Stevens (2001), *KI 2001: Advan- ces in Artificial Intelligence*, Vol. 2174/2001, Springer Berlin / Heidelberg, capítulo OilEd: A Reason-able Ontology Editor for the Semantic Web, pp. 396–408.

Shooman, M. (1990), *Probabilistic Reliability an Engineering Approach*, Krieger Publishing Company.

Sowa, John F. (1999), Knowledge representation: Logical, philosophical, and computati- onal foundations, *em* 'Brooks Cole Publishing Co., Pacific Grove'.

Uschold, M. & M. Gruninger (1996), Ontologies: principles, methods and applications, *em* 'Knowledge Engineering Review'.

Uschold, M. & M. King (1995), Towards a methodology for building ontologie, *em* 'IJ- CAI95 Workshop on Basic Ontological Issues in Knowledge Sharing', Montreal.