UNIVERSIDADE FEDERAL DE SANTA CATARINA

ARTEFATOS DA WEB SEMÂNTICA NO APOIO AOS PROCESSOS ITIL

MARCOS HENRIQUE DOS SANTOS

Florianópolis - SC

2008

UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

ARTEFATOS DA WEB SEMÂNTICA NO APOIO AOS PROCESSOS ITIL

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistema de Informação.

AUTOR: MARCOS HENRIQUE DOS SANTOS

ORIENTADOR: PROF. DR. RENATO FILETO

Florianópolis - SC

2008

MARCOS HENRIQUE DOS SANTOS

ARTEFATOS DA WEB SEMÂNTICA NO APOIO AOS PROCESSOS ITIL

Este Trabalho de graduação foi julgado adequado para a obtenção do grau de Bacharel em Sistemas de informação e aprovado em sua forma final pelo Curso de Sistemas de Informação da Universidade Federal de Santa Catarina.

_____

Prof. Renato Fileto

Orientador

_____

Prof. Fernando Ostuni Gauthier

Banca examinadora

_____

Prof. Roberto Carlos dos Santos Pacheco

Banca examinadora

Dedico este trabalho ao meu Pai in memoriam.

Agradeço à Thais por toda compreensão e cumplicidade; aos meu amigos que muitos encontrei na vida, outros trabalhando e muitos estudando; aos meus mestres por todo conhecimento e exemplo e em especial à minha Mãe a qual creio ter herdado sua força e alegria de viver.

RESUMO

O presente trabalho demonstra o uso de artefatos desenvolvidos para a Web Semântica para apoiar a implementação de processos do arcabouço de boas práticas ITIL (*Information Technology Infraestucture Library*), um dos mais conhecidos conjuntos de conceitos, diretrizes e técnicas para apoiar a governança de tecnologia da informação (TI). O trabalho faz uso de ontologias, arcabouços de manipulação de ontologias e raciocinadores autômatos para avaliar o impacto da falha de itens de configuração (ICs) nos processos de negócio que os utilizam. Os experimentos realizados sobre o protótipo implementado no âmbito deste trabalho demonstram a viabilidade e os benefícios do uso destes artefatos.

Palavras-chave: ontologias, ITIL, governança de TI, Web semântica, Jena, Pellet, processos de negócios.

ABSTRACT

This work shows the use of semantic Web tools for supporting the implementation of processes from ITIL (Information Technology Infraestucture Library), one of the best known set of concepts, guidelines and techniques for managing information technology (TI). The work employes ontologies, ontology management frameworks e autonomous reasoners to evaluate the impact of failed configuration items (CIs) in the business processes that use them. The experiments made on a prototype developed as part of this work show the viability and the benefits of using these artifacts.

Keywords*: ontologies, ITIL, TI governance, semantic Web, Jena, Pellet, business processes.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS E ABREVIATURAS

C# – *C Sharp*
CCTA – *Central Computer and Telecommunications Agency*
CI – *Configuration Item*
CMDB – *Configuration Management Database*
CMMI – *Capability Maturity Model Integration*
COBIT – *Control Objectives for Information and related Technology*
DHS – *Definitive Hardware Store*
DSL – *Definitive software library*
ETC – *Etcetera*
EXIN – *Examination Institute for Information Science*
GITIM – *Government Information Technology Infrastructure Management*
HTML – *Hyper Text Markup Language*
IBM – *International Business Machines*
IC – *Item de configuração*
ICT – *Information and communication technology*
IDE – *Integrated Development Environment*
IKVM – *Java Virtual Machine (I letra antes do J e K da letra depois do J)*
IndGen – *Individual Generator*
ISEB – *Information System Executive Board*
ISO – *International Organization for Standardization*
ITGov – *Information Technology Governance*
ITIL – *Information Technology Infraestucture Library*
itSMF – *Information Technology Service Management Forum*
MIT – *Massachusetts Institute of Technology*
MOF – *Microsoft Operations Framework*
MOM – *Microsoft Operations Manager*
MPS.BR – *Melhoria de Processo do Software Brasileiro*
NS – *Name Space*
OECD – *Organisation for Economic Co-operation and Development*
OGC – *Office of Government Commerce*
OGC – *Office of Government Commerce*
OMG – *Object Management Group*
OS – *Operational System*

OWL – *Ontology Web Language*

OWL – *Ontology Web Language*

OWL–DL – *Ontology Web Language Descriptive Logic*

POP3 – *Post Office Protocol version 3*

PRINCE2 – *Projects In Controlled Environments 2*

RDF – *Resource Description Framework*

RFC – *Request for Comments*

RUP – *Rational Unified Process*

SGML – *Standard Generalized Markup Language*

SLA – *Service Level Agreement*

SMTP – *Simple Mail Transfer Protocol*

SPARQL – *SPARQL Protocol And RDF Query Language*

SPARQL-DL – *SPARQL Protocol And RDF Query Language Descriptive Logic*

SVN – *Subversion (version control system)*

TI – *Tecnologia da Informação*

URI – *Uniform Resource Identifier*

URN – *Uniform Resource Names*

W3C – *Word Wide Web Consortium*

WFMC – *Workflow Management Coalition*

XML – *eXtensible Markup Language*

1 INTRODUÇÃO

A governança de TI é um dos grandes desafios das empresas, mesmo para aquelas cuja atividade fim não seja tecnologia da informação e comunicação, como empresas de manufatura, construtoras e transportadoras. Toda empresa moderna necessita de ferramentas computacionais para o controle de suas atividades e é na devida gestão dos departamentos de *TI* que figura a Governança de *TI*. [3, 40]

Neste cenário, durante anos surgiram centenas de boas práticas ao redor do mundo e estas foram compiladas pela *OGC* (*Office of Government Commerce*) em um *framework* (arcabouço) denominado *ITIL* (*Information Technology Infraestucture Library*) [9, 10, 40].

Este trabalho propõe, constrói e avalia uma arquitetura em camadas usando ontologias, as quais permitem modelar a infra-estrutura de *TI* (i.e., itens de configuração, incluindo hardware, software e recursos humanos) necessária ao funcionamento de uma instituição além de modelar os seus processos de negócio e processos relativos ao *ITIL*. O objetivo é utilizar esta modelagem juntamente com raciocinadores autônomos para avaliar o impacto da falha de itens de configuração (ICs) específicos nos processos de negócio dessas instituições, de modo a orientar políticas para correção e prevenção de falhas de falhas desses ICs. O trabalho inclui, a título de estudo de caso, a modelagem da infra-estrutura de uma empresa fictícia com um processo de negócio juntamente com um dos principais processos do *ITIL*, a fim de demostrar a capacidade da arquitetura desenvolvida. A partir dessa infra-estrutura são feitas inferências e respostas são obtidas com o uso de um raciocinador autômato. Também foram feitos dois experimentos de desempenho e seus resultados são mostrados no capítulo 7.

1.1 Delimitação do escopo

Este trabalho não se propõe a modelar todos os processos do ITIL nem chegar perto disso, também não se propões a testar todos os artefatos desenvolvidos para a Web Semântica como linguagens, raciocinadores, arcabouços e ferramentas de modelagem. O que é proposto aqui é avaliação de um conjunto pré-determinado de tecnologias, ferramentas, linguagens os quais também não são esgotados na extensão de suas potencialidades.

1.2 Objetivos

1.2.1 Objetivos gerais

Este trabalho tem como objetivo avaliar artefatos desenvolvidos para a Web Semântica que apóiem a implantação dos processos de negócios e de gerência de configuração do ITIL. Ele usa ontologias para modelar tais processos e raciocinadores autônomos (*reasoners*) parar avaliar o impacto de problemas em itens de configuração (*software*, *hardware*, etc.) nos processos de negócio.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Desenvolver uma arquitetura em camadas usando ontologias com o objetivo de formalizar semanticamente os conceitos envolvidos na Gerência de configuração bem como na definição de processos técnicos e de negócio. Esta arquitetura terá a capacidade de ser usado por raciocinadores na obtenção de respostas pertinentes ao domínio.

2. Realizar experimentos com a intenção de se averiguar a viabilidade do uso dos artefatos propostos conjuntamente com artefatos e ferramentas da Web Semântica.

1.3 Motivação

Com a quebra da bolsa eletrônica *Nasdaq* em 2002 e com os escândalos das empresas *Enron*, *Tyco International*, *Peregrine Systems* e *WorldCom*, os investidores mundiais começaram a procurar formas de garantir seus investimentos de maneira a não ter prejuízo, como muitas empresas tiveram na ocasião[1]. Governos começaram a encarar a Governança de TI como fator relevante para a conjuntura financeira e estratégica das nações, tendo como principal expoente a América do Norte na consolidação da lei "*Sarbanes-Oxley Act of 2002*"[1]. Neste cenário, usar um *framework* de melhores práticas pode garantir o retorno do investimentos dos departamentos de TI das empresas através de práticas comprovadas ao longo dos anos.

Visualizando este viés, foram pesquisadas artefatos e ferramentas que podem apoiar no processo de implementação do ITIL e constatou-se que muitos produtos baseiam-se em estruturas fixas em seus códigos fonte que quase sempre são proprietários. Isso torna o processo de implantação do ITIL um ato de amarração dos processos da empresa a esses produtos.

1.4 Objeto

1.4.1 O problema

Os livros que definem as melhores práticas do ITIL apenas definem o que deve ser feito, mas não trazem informações de "como" fazer. A idéia deste trabalho é fazer uso de artefatos e ferramentas da Web Semânticas definidas pela W3C dentre elas as ontologias na forma de arquivos OWL e persistidas em banco de dados na modelagem

dos processos do ITIL. Assim fornecendo uma abordagem alternativa, as quais qualidades como descentralização, especificação aberta, escalabilidade, adaptabilidade e flexibilidade podem ser distinguidas.

Sendo a especificação OWL pública, as empresas optantes estarão menos sujeitas aos efeitos negativos do uso de plataformas fechadas, tendo assim maior flexibilidade de mudanças e transparência. Com o uso de especificação aberta, empresas parceiras poderão interoperar e ter seus processos alinhados usando estas ontologias, assim, favorecendo a formação de um ecossistema de negócios.

Com a formalização de conceitos lógicos através de ontologias é possível, por meio do uso de *reasoners* (raciocinadores autômatos) obter respostas operacionais e estratégicas, como por exemplo: “quais processos de negócio serão afetados com o desligamento de um determinado servidor da meia noite à uma hora da manhã?”, “qual o custo de desligarmos este seguimento de rede da meia noite à uma hora da manhã no domingo?” ou “quais pessoas serão interessadas que serão afetadas com determinada mudança?”

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Governança

Uma publicação de 1999 da Organização para Cooperação Econômica e Desenvolvimento “*OECD Principles for Corporate Governance*,” define governança corporativa como:

“*Providing the structure for determining organizational objectives and monitoring performance to ensure that objectives are attained*.[5]”

Em uma tradução livre:

"*Prover uma estrutura para determinar objetivos organizacionais e monitoração de desempenho de maneira a assegurar que esses objetivos sejam alcançados*.[5]"

2.1.1 Governança de TI

A Governança de TI é um subconjunto da Governança, conforme pode-se observar nas áreas cinzas da figura 1.

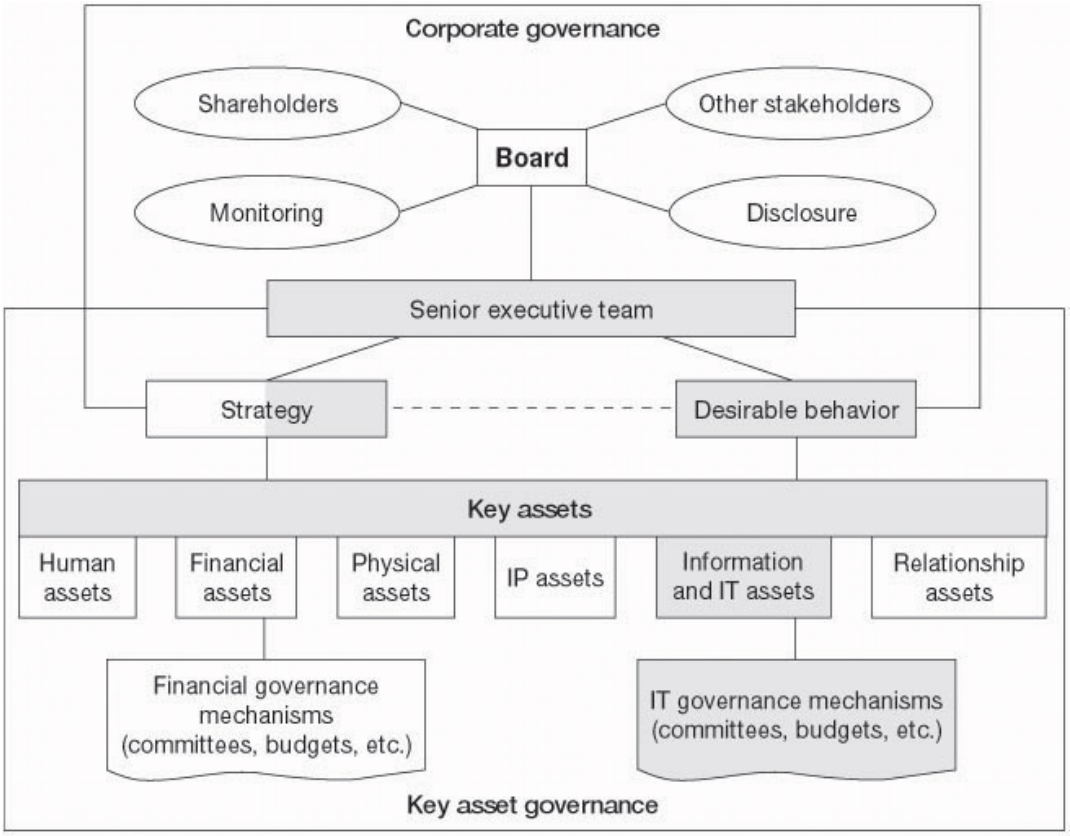

Figura 1 – Modelo de Governança Coorporativa – Fonte: IT Governance: How Top Performers Manage IT Decision Rights for Superior Results – Capítulo 1.

Através do diagrama pode-se notar a infiltração da governança de TI no modelo de governança coorporativa. Sendo assim, verifica-se sua vital importância na perfeita operação coorporativa, estando ligada aos níveis operacional, tático e estratégico.

Segundo o *IT Governance Institute's* "*COBIT 3rd Edition Executive Summary*" de Julho de 2000 a definição de Governança de TI é:

"*structure of relationships and processes to control the enterprise to achieve the enterprise's goals by adding value while balancing risk versus return over IT and its processes.*"

Em uma tradução livre:

"*estrutura de relacionamentos e processos para controlar a empresa para alcançar os objetivos corporativos através da adição de valor avaliando riscos versus retornos sobre TI e seus processos.*"

Uma boa governança pode prevenir escândalos e fraudes nas empresas, evitando danos morais e financeiros. A melhora na reputação de uma empresa torna-a mais atrativa para os clientes, investidores, fornecedores, e no caso de organizações sem fim lucrativos, voluntários[1].

Um trabalho do *MIT Sloan School of Management* feito em 250 empresas concluiu que, em média, os negócios implementados com as melhores práticas de governança de TI geram 25% mais lucro do que empresas cuja governança poderia ser considerada fraca [4].

Numa outra pesquisa liderada por Peter Weil da *MIT Sloan School of Management* envolvendo 147 empresas estudadas num período de quatro anos, as instituições foram classificadas em uma lista crescente de maturidade [4]. As sete do topo de maturidade figuram entre as 5% que obtiveram $250 de lucro para cada dólar investido em TI. Já as sete empresas da base da lista ordenada de maturidade, amargaram prejuízos de $900,00 para cada dólar investido em TI [4].

2.1.2 Ferramentas para Governança de TI

O mercado disponibiliza várias ferramentas para se atingir níveis aceitáveis de Governança de TI, dentre as quais podemos citar aquelas do conjunto ilustrado na figura 2.

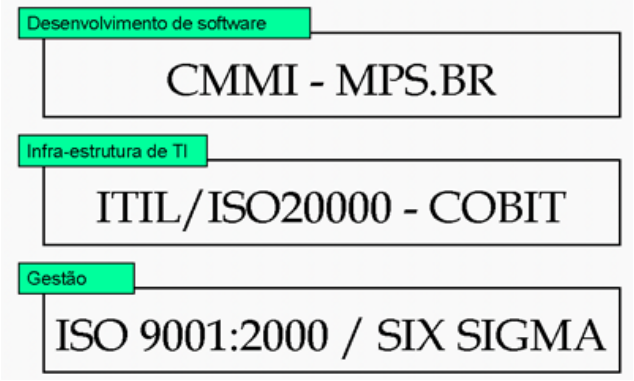

Figura 2 – Ferramentas para Governança de TI.

Para o desenvolvimento de *software* há o CMMI e o MPS.BR, modelos de maturidade do processo de desenvolvimento de software muito próximos em seus conteúdos. Outras ferramentas de desenvolvimento são o RUP (*Rational Unified Process*) da IBM e o PRINCE2 da ISEB.

Na área de infra-estrutura temos o ITIL, que é um conjunto de boas práticas padronizadas na norma ISO20000. Há também o COBIT, que se assemelha ao ITIL nos seus objetivos e áreas de processo. Para gestão é possível usar a ISO9001:2000 e o *framework Six Sigma*.

Todas essas ferramentas têm como objetivo auxiliar a corporação a alcançar níveis competitivos de maturidade. Este trabalho foca no ITIL que é um *Framework* de boas práticas com mais de 25 anos.

O ITIL possui uma abordagem sistêmica que vai desde o nível operacional, passando pelo tático e estratégico. Um breve resumo de cada um dos livros foi feito no capitulo 3.

2.2 Web Semântica

Em maio de 2001 Tim Berners-Lee, James Hendler e Ora Lassila publicam na revista *Scientific American* um artigo entitulado: *The Semantic Web*: "*A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*". O qual inspirou muitos pesquisadores, entidades e leigos a partir daí.

Esse artigo é iniciado na ambientação de uma cena futurista onde ao tocar o telefone, o sistema de áudio de Peter, um dos seus personagens, reduz o volume de som automaticamente e durante a conversa com sua irmã Lucy, ambos usam seus agentes de maneira a agendar uma séries de sessões médicas para a mãe dos personagens [24]. Nesse enredo é introduzido de forma acessível as pretensões da Web Semântica na qual agentes inteligentes poderão consultar uma base global de informações e interagir com esta e com outros agentes, com a finalidade de oferecer serviços aos seus usuários. No final da história, os autores introduzem alguns conceitos e ferramentas no sentido de concretizar tais idéias como: Ontologias, RDF, URI, e o Teorema da Incompletude de Gödel [24].

De 2001 aos dias de hoje, esta ficção vem se tornando cada vez mais materializada a partir de inúmeros esforços de pesquisadores, empresas e entidades.

2.2.1 Motivação

A intenção da Web Semântica é fornecer sentido a conhecida Web atual caracterizada por sua abordagem sintática [25]. A atual Web é dirigida a agentes humanos, com sua linguagem natural e representações na forma de imagens, o que provê pouco ou quase nenhum suporte à inferência de máquinas. A idéia por trás da Web Semântica é fornecer subsídios semânticos aos atuais *Web-Sites* de maneira que estes sejam igualmente entendidos por agentes autômatos.

2.2.2 Camadas

A proposta da Web Semântica consistia da construção de 7 camadas integradas as quais são ilustradas na figura abaixo:

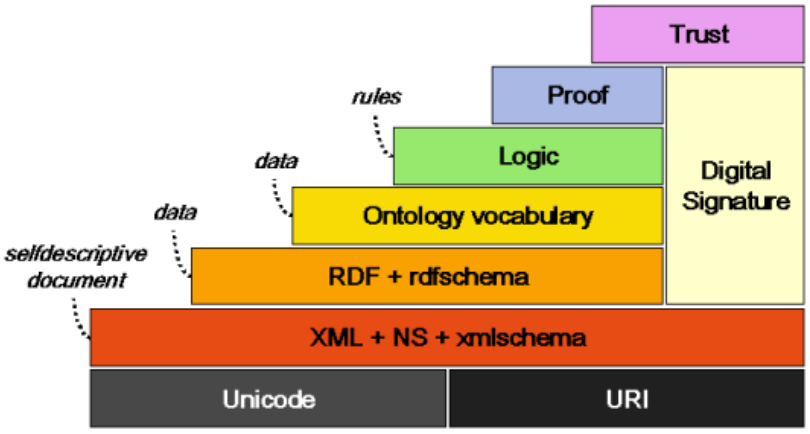

Figura 3 – Pilha de padrões original da Web semântica – http://www.w3.org/2001/12/semweb-fin/w3csw Acessado em 19/11/2007

Em 2005 foi atualizada pilha da da Web Semântica a qual segue abaixo:

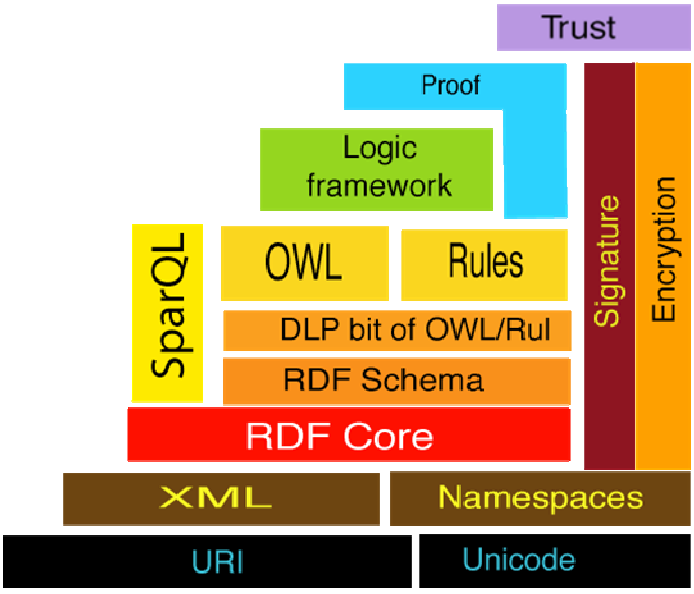

Figura 4 – Pilha de padrões revisada da Web semântica - http://www.w3.org/2005/Talks/0511-keynote-tbl/#[17] acessado em 18/05/2008

A seguir serão explanadas as principais camadas de forma sucinta.

2.2.2.1 Unicode

O Unicode é um padrão universal de codificação de caracteres para a escrita de caracteres e textos [27]. Definido como uma forma consistente de codificar textos multilíngüe, os quais podem ser intercambiados internacionalmente, criando uma base para desenvolvimento e uso de *software* globalmente[27].

O padrão Unicode tem uma importância primordial na fundamentação da Web Semântica, fornecendo um conjunto padronizado de caracteres, os quais as *URIs* são definidas.

2.2.2.2 URI

URI (*Uniform Resource Identifiers*) traduzindo: "Identificadores Universais de Recurso", provêem meios simples e extensíveis para identificação de recursos [28]. A especificação das URIs foram desenhadas em conformidade com as *RFCs* (*Request for Comments):* "*Functional Recommendations for Internet Resource Locators*" [RFC1736] e "*Functional Requirements for Uniform Resource Names*" [RFC1737] [28].

As URI*s* têm duas características vitais: Uniformidade e Recursos. A uniformidade pode ser definida por:

"*permite diferentes tipos de identificadores de recursos serem usados no mesmo contexto, mesmo que os mecanismos de acesso sejam diferentes; permitindo uma interpretação semântica uniforme de conversões sintáticas comuns sobre tipos de recursos diferentes sem a interferência com a forma com que estes identificadores de recursos sejam usados; permitem que identificadores sejam reusados em diferentes contextos; desta forma permitindo que novos softwares e protocolos façam uso de uma ampla gama de identificadores de recursos pré-existentes e amplamente usados.* [28]"

Recursos podem ser definidos como:

"*qualquer coisa que possua identidade*"[28]

Podemos citar alguns exemplos, como um documento eletrônico, uma imagem e um serviço, (Ex: “relatório da previsão do tempo de Los Angeles para hoje”). Recursos também podem ser coleções de outros recursos.

É importante frisar que não apenas recursos que possam ser buscados numa rede sejam considerados “Recursos”; incluem-se nesta categoria seres humanos, empresas e livros em uma biblioteca, por exemplo[28].

2.2.2.3 XML + NS + xmlschema

XML – (*Extensible Markup Language*) Consiste de um subconjunto da linguagem SGML (*Standard Generalized Markup Language*), sua meta é de ser servida, recebida e processada na *Web* da mesma maneira que hoje é feito pela HTML. A linguagem XML foi projetada para ser de simples implementação e de fácil interoperabilidade com a SGML e a HTML [29].

Sendo uma meta-linguagem, a XML permite seu uso na definição de outras linguagens, podendo servir tanto como especificação, quanto dados.

NS – (*Namespaces*) provêem um método simples de qualificação de nomes de elementos e atributos usados por documentos XML através de sua associação com os identificadores de *namespaces* e através das referências URI [30].

Xmlchema – Seu propósito é prover um conjunto de construções de marcação XML, os quais servem para escrever esquemas (*schemas*) [31].

O propósito de um *schema* é descrever e definir uma classe de documentos XML de maneira a especificar formalmente o sentido destes documentos, seus usos e relacionamentos de suas partes constituintes: *datatypes*, elementos e seu contexto, atributos e seus valores, entidades e seus conteúdos, e finalmente, notações[31].

Desta forma, a linguagem de esquema XML pode ser usada para definir, descrever e catalogar vocabulários XML para as classes de documentos XML [31].

Os *scheamas* XML definem a sintaxe necessária para o alicerce da Web Semântica. O uso de XML como linguagem base e os *namespaces* para a definição do contexto, formam o lastro léxico e sintático, as quais serão completadas pelas demais camadas.

2.2.2.4 RDF + rdfschema

RDF (*Resource Description Framework)* é um framework para a representação de informações na *Web* [32].

Possui uma sintaxe abstrata que reflete um modelo de dados simples de grafos e uma semântica formal. Possui um rigoroso e bem definido conceito de implicação, o que promove os fundamentos para deduções bem definidas a partir dos dados RDF[32].

A semântica é outro fato inerente ao RDF, a qual provê uma base confiável para o raciocínio sobre expressões RDF através do conceito de implicação e regras confiáveis [32].

A expressão definida por uma tripla é o conceito por trás do RDF, o qual forma um grafo. Esta tripla é composta por Sujeito e Predicado (também conhecido como (Propriedade) que denota o relacionamento e Objeto [32].

Figura 5 – Tripla RDF – Fonte: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ Acessado em 22/11/2007

O esquema RDF (RDF *schema*) tem o objetivo de especificar um vocabulário descritivo das triplas RDF, fornecendo uma base para sua especificação.

2.2.2.5 Ontologias

Segundo Gruber, Ontologias são:

"*is an explicit specification of a conceptualization*[33]"

Que pode ser traduzido como:

"*uma especificação explícita de um conceitualização*"

Esse termo foi emprestado da filosofia, embora que para os filósofos exista apenas uma ontologia e para a Web Semântica existam várias ontologias.

Nas camadas da Web Semântica, ontologias são representadas pelo padrão OWL, onde são definidos três tipos, os quais diferenciam-se pela sua expressividade; *OWL-Lite* (menos expressiva), *OWL-DL* e *OWL-Full* (mais expressiva) [25].

Ontologias são feitas por meio de triplas RDF, estas fornecem o sentido das coisas nas camadas da Web Semântica, provendo aos conceitos relações do tipo todo-parte, fraternidade e herança, entre outros. É nesta camada onde o conhecimento humano é especificado, formando-se assim, um vocabulário estruturado em que mecanismos autômatos podem fazer inferência acerca de seus conteúdos.

2.2.2.6 Logic, Proof, Trust

As camadas de Lógica (*Logic*), Prova (*Proof*) e Confiança (*Trust*) ainda estão sendo pesquisadas e algumas aplicações simples de demonstração estão sendo construídas. A camada de lógica permite a escrita de regras, enquanto a camada de prova executa estas regras e as avalia juntamente com o mecanismo de confiança, o qual assegura ou não a prova dada [26].

2.2.2.7 SPARQL

SPARQL é uma linguagem de consulta em bases RDF [41]. Assim como o SQL, o SPARQL funciona de forma análoga sobre as triplas RDF. Existem algumas variantes do SPARQL como o SPARQL-DL, onde seu poder de busca é mais assertivo com relação às inferências por *entailment*, retornando blocos consistentes de OWL-DL [42].

Abaixo segue uma consulta exemplo em SPARQL:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acme: <http://www.inf.ufsc.br/~marcoshs/ontologies/acme.owl#>

SELECT ?x WHERE {

?x a cmdb:Medias .

}
```

Quadro 1 – Exemplo de consulta SPARQL.

As primeiras duas linhas especificam os prefixos usados na consulta, assim evitando a desnecessário uso de *URI* longas dentro das clausulas seguintes.

A terceira linha contém a instrução *SELECT* a qual define a projeção [43] das variáveis seguintes, no caso do exemplo, ?x. Após isso, encontra-se a instrução *WHERE* a qual permite fornecer critérios para a projeção de ?x.

Dentro do bloco *WHERE*, delimitado por chaves estão as cláusulas de consulta onde é definido o padrão do grafo[43], o qual consiste de triplas *RDF* finalizadas por um ponto final. Note que é a relação *RDF is-a* é abreviada simplesmente pela letra a.

A consulta acima retorna todas as *Medias* da ontologia *Acme*.

A linguagem SPARQL é ainda uma recomendação da W3C [41] e nem todos os mecanismos de consulta suportam todas a produções de sua especificação, bem como existem dialetos próprios sendo estudados.

2.3 Anotações Semânticas

Anotações são comentários, notas, explanações ou outro tipo de anotação que possa ser anexada em um documento *Web* ou em uma parte específica de um documento. Como eles são externos, é possível anotar qualquer documento *Web* independentemente, sem a necessidade de editar o referido documento[36]. Do ponto de vista técnico, anotações são meta dados que fornecem alguma informação adicional a algum dado existente[36].

2.4 Intranet Semântica

Na bibliografia pesquisada para este trabalho, não foi encontrada uma definição formal de *Intranet* Semântica, apenas alguns artigos que forneciam certo tipo de ferramenta para seu uso. Livremente, com base na *Web* convencional originou-se o termo *Intranet* como sendo uma rede interna de um meio restrito, como um ambiente coorporativo ou acadêmico. Sendo assim, *Intranet* semântica pode ser definida como:

"*um conjunto de ferramentas e técnicas da Web Semântica usadas em um ambiente de rede com acesso controlado e restrito*"

2.5 ITIL

O *Information Technology Infrastructure Library* [39], pode livremente ser traduzido por Biblioteca de Infra-estrutura da tecnologia da informação, é um trabalho de compilação de melhores práticas propostas desde a década de 80, desenvolvido pelo (OGC) *Office of Government Commerce*, uma instituição do Reino Unido. Como é um *framework*, o ITIL é versionado a cada nova edição dos livros que o compõem. Até a data de escrita deste texto, sua última versão era a 3.0. Devido ao caráter recente desta versão, lançada em maio de 2007, falta de experiência e maturação e juntando-se ao fato deste trabalho ter sido iniciado antes do lançamento da versão 3.0 do ITIL, a fundamentação deste trabalho foi balizada na versão 2.2, cabendo um capítulo adicional com informações complementares da versão 3.0 e uma menção aos trabalhos futuros.

A versão 2.2 do ITIL é documentada em 8 livros:

- ***Service Delivery***
- ***Service Support***
- *ICT Infrastructure Management*
- *Security Management*
- *Management of Risk*
- *Planning to Implement Service Management*

- *Application Management*
- *Software Asset Management*

Os conteúdos desses livros foram abordados com mais detalhes nos capítulos seguintes.

Além dos livros, juntamente à OGC existe o Fórum de Gerência de Serviços itSMF (*Information Technology Service Management Forum*), entidade onde são realizadas discussões, resoluções e publicações acerca do ITIL. O itSMF conta com comitês em 42 países, constituindo uma forte base para a endosso das práticas do ITIL.

Somando-se a isso, entidades também associadas ao OGC/itSMF, como o EXIN (*Examination Institute for Information Science*) e a ISEB (*Information System Executive Board*), provêem provas de certificação para profissionais da área de TI que queiram provar seus conhecimentos e serem reconhecidos por isso. Empresas lançam mão da contratação de profissionais certificados, a fim de reduzir riscos em seus projetos.

2.6 História

No inicio da década de 80 o governo Britânico detectou que os serviços de TI providos a ele não tinham nível de qualidade suficiente para suas necessidades [13]. A CCTA (*Central Computer and Telecommunications Agency*) hoje OGC (*Office of Government Commerce*) ficou encarregada do desenvolvimento de um arcabouço que pudesse assegurar o provimento de serviços eficientes e financeiramente responsáveis para o governo Britânico e o setor privado. Este esforço acabou dando origem ao ITIL. Nas primeiras versões o ITIL era chamado de GITIM (*Government Information Technology Infrastructure Management*). Ambos possuem seu foco na entrega e suporte a serviços (*Service Delivery* e *Service Support*) [13].

Nos primeiros anos da década de 90, grandes empresas européias e agências governamentais adotaram o GITIM . Desta forma, ele se espalhou, atingindo cada vez

mais empresas e órgãos governamentais ao redor do mundo. Até que em determinado momento começou a evoluir por si só nos departamentos de TI (Tecnologia da Informação) das empresas e organizações, tornando-se assim, o conhecido ITIL. [13]

No ano 2000 o CCTA tornou-se parte do OGC (*Office of Government Commerce*) e neste mesmo ano a Microsoft usou o ITIL como base para seu *framework* proprietário, o MOF (*Microsoft Operations Framework*) [13], cujos processos são integrados com produtos da própria Microsoft. Entre os principais produtos usados pelo MOF, figura o MOM (*Microsoft Operations Manager*) que procura centralizar a administração de TI em um ponto central [14].

Em 2001 o ITIL 2.0 foi lançado; os livros *Services Delivery* e *Service Support* foram recompilados em volumes mais concisos e práticos. A partir desse momento, o número de adeptos ao redor do mundo foi crescendo cada vez mais [13].

Em 2007 foi lançada a versão 3, que adota uma abordagem voltada ao ciclo de vida para a gerência de serviços, com grande ênfase na integração da TI aos objetivos de negócios [13].

2.7 Livros

O ITIL 2.2 é um conjunto de 8 livros cujos principais são *Service Delivery* (Entrega de Serviços) e *Service Support* (Suporte a serviços). As próximas seções discorreram sobre cada livro.

2.7.1 Service Support

O *Service Support* é um dos principais livros da coleção, pois possui um ponto de vista mais operacional e tático das operações de um departamento de TI. Segue um resumo de cada processo.

2.7.1.1 Central de Serviços (Função)

A Central de Serviços (*Service Desk*) não é considerado um processo e sim uma função dentro do ITIL. Tem por objetivo ser um único ponto entre os serviços providos e os usuários nas atividades do dia-a-dia. Uma de suas atividades é manter os usuários informados a respeito de eventos, ações e oportunidades que possam impactar nas atividades cotidianas dos usuários[10].

Por exemplo, ao receber uma ligação do usuário, o atendente torna-se responsável por todo o ciclo de vida do incidente, sendo responsável por mantê-lo ciente do andamento de sua solicitação até o fechamento do chamado. Sendo assim o único ponto de contato do cliente com a empresa[10].

2.7.1.2 Gerenciamento de Incidentes

Seu principal objetivo é a restauração da normalidade dos serviços o mais rápido possível e a minimização do impacto negativo nas operações do negócio. Também assegura a entrega de serviços no melhor nível possível de qualidade e disponibilidade[10].

"Normalidade dos serviços" é definido nos livros como operação dentro dos níveis limites do SLA (*Service Level Agreement*)[10].

Na terminologia ITIL um incidente é definido como:

"*qualquer evento o qual não faz parte da operação padrão de um serviço o qual causa, ou pode causar uma interrupção ou redução na qualidade de serviço.*"[10]

2.7.1.3 Gerenciamento de Problemas

Seu objetivo principal é minimizar os impactos adversos dos Incidentes e Problemas nos processos de negócio, os quais podem ser causados por erros dentro da infra-estrutura de TI. O Gerenciamento de Problemas também previne a recorrência de incidentes relacionados a esses erros. Para atingir estes objetivos, a Gerência de

Problemas procura encontrar as causas raiz dos incidentes e então iniciar as ações de melhoria ou correção para a situação[10].

O processo de Gerência de Problemas tem uma abordagem reativa com relação ao incidentes já ocorridos e pró-ativa em relação aos possíveis incidentes, baseando-se nos problemas conhecidos. [10]

2.7.1.4 Gerenciamento de Mudanças

Seu objetivo é assegurar que métodos e procedimentos padronizados sejam usados para a eficiente e pronta execução de todas as mudanças, de maneira a minimizar o impacto de incidentes decorrentes de mudanças. O Gerenciamento de Mudanças trabalha em prol da melhoria da qualidade e consequentemente da melhoria das atividades cotidianas da organização[1].

2.7.1.5 Gerenciamento de Configuração

Negócios necessitam serviços de TI economicamente viáveis. Para ser eficiente e efetivo, todas as organizações precisam controlar seus serviços e infra-estruturas de TI. A Gerência de configuração provê um modelo lógico de infra-estrutura que mantém a identificação, controle, manutenção e verificação de versões dos Itens de Configuração (*Configuration Itens - CI*) que existam no âmbito da empresa[10].

2.7.1.5.1 Objetivos da Gerência de Configuração

- Contabilizar todo o patrimônio de TI e suas respectivas configurações e serviços associados.
- Prover informações confiáveis sobre a configuração e documentação de toda infra-estrutura para ajudar nos processos de Gerência de Serviços.
- Prover subsídio para os processo de Gerência de Incidentes e Problemas.

- Verificar os registros de infra-estrutura frente ao real parque patrimonial e corrigir quaisquer incorreções.

A literatura do ITIL não traz uma definição precisa do termo Item de Configuração (*Configuration Item*) mas livremente, através de inferência dos textos podemos definir que um IC é:

"*qualquer item da infra-estrutura que pode ser substituído individualmente*" e "*todos os itens controlados pela Gerência de Configuração*"

Dentro de sua disciplina, o processo de Gerência de Configuração especifica a adoção de um banco de dados contendo todos os itens de configuração e seus relacionamentos[10].

Outra estrutura especificada, é uma biblioteca definitiva de *software* onde todas as versões dos *softwares* usados pela empresa são armazenados de forma segura[10].

2.7.1.6 Gerenciamento de Liberação

A Gerência de Liberação precisa ter uma visão holística (sistêmica) de cada mudança em qualquer serviço de TI. Devendo assegurar que todos os aspectos de uma Liberação, sendo técnica ou não, sejam consideradas conjuntamente.

Tem como responsabilidade: projetar, executar, configurar e testar *hardware* e *software* de maneira a criar um conjunto de componentes para serem implantados no ambiente de produção (liberações). Essas atividades também cobrem o planejamento, preparação e agendamento de uma Liberação para vários Clientes, em várias localidades.[10]

2.7.2 Service Delivery

O *Service Delivery* é tão importante quanto o *Service Support*, possuindo um foco tático e estratégico. Segue um resumo de cada processo deste livro.

2.7.2.1 Gerenciamento do Nível de Serviço

O objetivo da Gerência do Nível de Serviço é manter e melhorar a qualidade de serviços de TI, através do ciclo constante de aceitação, monitoração e geração de relatórios sobre as atividades dos serviços de TI. Em paralelo, o Gerenciamento do Nível de Serviço preocupa-se com a erradicação de serviços pobres – alinhando os serviços com os negócios e em um custo justificável. Um relacionamento melhor entre clientes e a TI podem ser construídos com o uso desses métodos.[9].

O gerenciamento de níveis de serviço tem a responsabilidade de lidar com os contratos de prestadores de serviços, contratos do departamento de TI e seus clientes, que podem ser internos ou externos. Essa responsabilidade se estende desde a formulação dos contratos até a sua monitoria.

2.7.2.2 Gerenciamento da Disponibilidade

A Gerência de Disponibilidade (*Availability Management*) tem como meta otimizar a capacidade da infra-estrutura de TI, serviços e demais agentes da organização para que a entrega de serviços tenha um custo efetivo, sustentando os níveis de disponibilidade. Dessa forma, os objetivos do negócio são alcançados.[9].

A gerência de disponibilidade tem uma função estratégica na mensuração e estimação das disponibilidades dos serviços de TI, provendo um plano claro que possa restabelecer um serviço que minimize o impacto nos *SLAs* num caso infortúnio.

2.7.2.3 Gerenciamento de Capacidade

O processo de Gerência de Capacidade é responsável por assegurar que a capacidade da infra-estrutura de TI esteja alinhada com a evolução da demanda dos negócios, com custos e prazos aceitáveis[9]. Este processo abrange:

- Monitoria de desempenho e capacidades de respostas dos serviços e componentes da infra-estrutura.
- Responsabilidade do ajuste (*tuning*) para obter melhor custo-benefício dos recursos existentes.
- Responsabilidade de produzir estimativas a partir das demandas atuais.
- Influenciar na demanda de recursos, provavelmente em conjunto com a Gerência Financeira.
- Produzir os planos de capacidade, os quais poderão ser usados pelos provedores de serviços na criação dos *SLAs*.

2.7.2.4 Gerenciamento da Continuidade

O objetivo deste processo é assegurar a continuidade do negócio garantindo que em caso de desastre os processos de negócio sejam restabelecidos dentro de tempo e custos aceitáveis [9].

Se ponto chave está na gerencia de riscos, onde planos de contingência são criados de maneira a minimizar o impacto de um desastre.

2.7.2.5 Gerenciamento Financeiro

Os objetivos da Gerência financeira são divididos em dois tipos de organização, conforme o ITIL: aquelas cuja TI é interna, não sendo o foco da empresa, e para aquelas cujo fim é o fornecimento de serviços de TI.

Para o primeiro tipo, sua principal meta é:

"*prover serviços, recursos e adquirir patrimônio em custos aceitáveis pela diretoria para a o provimento de serviços de TI.*" [9]

Para aquelas empresas cujo TI é seu fim, temos as seguintes definições:

"*ser capaz de contabilizar totalmente os gastos com serviços de TI e atribuí-los nos custos dos serviços entregues para os clientes da organização.*"[9]

"*ajudar nas decisões gerenciais acerca de investimentos de TI através do fornecimento detalhado de casos de negócio para as Mudanças nos serviços de TI.*"[9]

Para o ITIL, todo serviço tem que ser cobrado, seja fornecido interna ou externamente, ou seja, qualquer departamento da empresa ao solicitar serviços de TI estará pagando por estes, mesmo que o serviço em questão seja provido internamente. Por exemplo, o departamento de manufatura solicita mais espaço em suas caixas de correio eletrônico, por sua vez, esta mudança resultará na cobrança adicional por parte do departamento de TI para com o depto. de manufatura.

Para o ITIL estas cobranças podem ser feitas em espécie real ou nocional. Na modalidade nocional não há efetiva troca de valores entre departamentos, porém o fato de serem evidenciados estes valores, o resultado final será uma melhor utilização dos recursos existentes, e este é o objetivo principal deste tipo de cobrança.

2.7.3 ICT Infrastructure Management

Este livro traz uma abordagem técnica dos processos descritos nos livros *Service Delivery* e *Service Support*, no que diz repeito à infra-estrutura dos serviços de informação e comunicação das empresas[15]. Vale lembrar, que o ITIL não é indicado apenas para empresas cujo foco é TI, e sim qualquer empresa que faça uso da TI em suas operações, como mencionado no capítulo de Governança. Conseqüentemente a implementação da gerência de serviços através dos processos ITIL requer uma instrumentação técnica que este livro fornece.

A Gerência de infra-estrutura de TIC tem foco nos seguintes processos:

2.7.3.1 Design and Planning

Estabelece estratégias e processos de desenvolvimento e implantação de soluções apropriadas de infra-estrutura de TIC por toda a organização.[15].

O processo é focado em coordenar a modelagem e planejamento de todos os aspectos de TIC e servir como interface entre a TIC e os processos de negócios. A abrangência vai desde tecnologias como *Main Frames*, Sistemas distribuídos, redes, *desktops* e dispositivos móveis, passando por *Frameworks* completos, arquiteturas, processos, procedimentos e métodos de gerência [15].

2.7.3.2 Deployment

Concentra-se na implementação e estabelecimento de soluções de negócio e/ou técnicas como foram planejadas, com o mínimo de impacto negativo nos processos de negócio[15].

2.7.3.3 Operations

Todas as atividades e medidas para possibilitar e/ou manter uma infra-estrutura de TIC como planejada[15].

2.7.3.4 Technical Support

Tem como meta o desenvolvimento do conhecimento necessário para manter, fortalecer e avaliar o estado atual e futuro de soluções de infra-estrutura de TIC[18].

O diagrama abaixo descreve os relacionamentos com o demais processo e livros do ITIL:

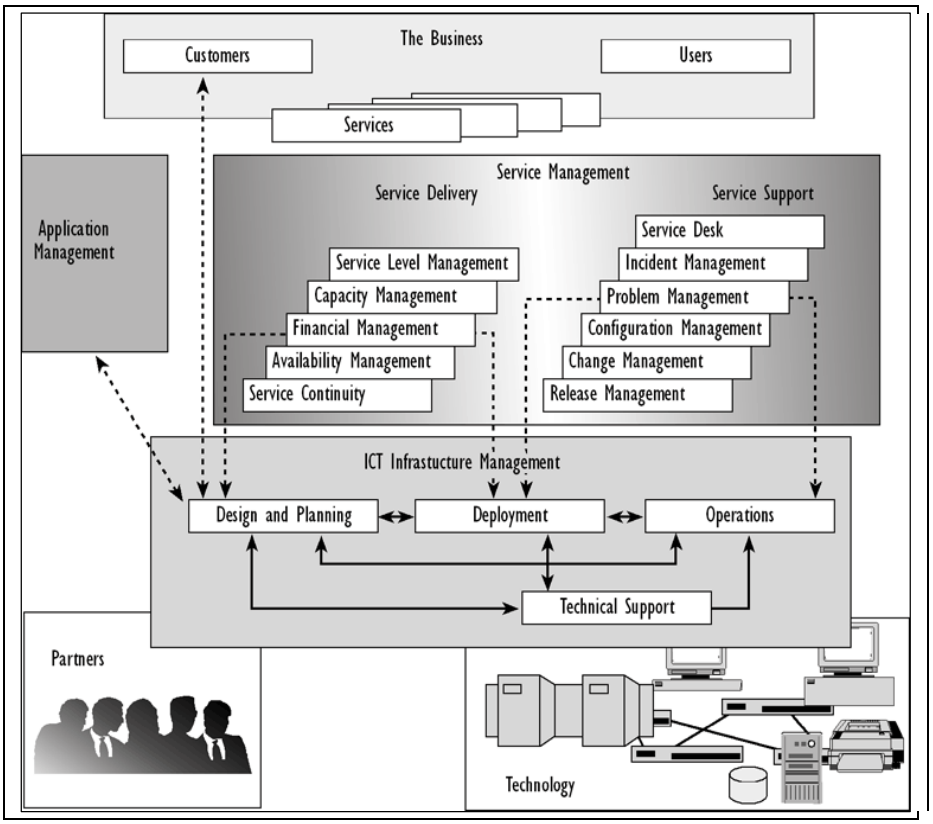

Figura 6 – Relacionamento do ICT Infrastructure Management com os demais processos – Fonte: ITIL ICT Infrastructure Management v.2.2 – Capítulo 1.5

2.7.4 Security Management

É o processo de gerência de um nível de segurança definido, incluindo respostas a incidentes de segurança [16].

A importância da segurança da informação vem crescendo drasticamente devido ao crescente uso de redes públicas e amplas (*Internet* e *Intranets*) para transações entre clientes e parceiros em operações de comércio eletrônico.

O amplo uso da informação e o processamento da informação vem crescendo juntamente com a necessidade desses processos terem uma política de proteção estruturada e organizada [16].

Nesse sentido, esse livro dirige-se a essas preocupações, pois descreve a Gerência de Segurança dentro das fronteiras do ITIL, no que tange segurança física e

dos bens intangíveis, como a informação. A segurança da informação atua nos três níveis de atuação do ITIL:

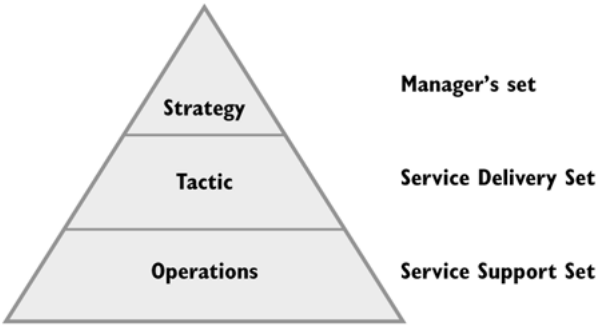

Figura 7 – Três camadas do ITIL – Fonte: ITIL ICT Security Management v.1.0 – Capítulo 3.1.1

A gerência de segurança é balizada em três fundamentos, os quais [16]:

- Confidencialidade
- Integridade
- Disponibilidade

2.7.5 Management of Risk

Para o guia da gerência de riscos do ITIL, um risco é definido como um resultado inesperado, tanto nos casos de oportunidades, onde o resultado é positivo, quanto em ameaças cujos resultados são negativos. O termo gerência de risco incorpora todas as atividades requeridas para identificar e controlar a exposição ao risco, as quais podem comprometer os objetivos de negócio da organização[17].

Este livro trata dos detalhes da gerência de risco, fornecendo um modelo de boas práticas para o eficaz gerenciamento dos mesmos.

2.7.6 Planning to Implement Service Management

A leitura deste livro é recomendável após a leitura dos livros *Service Delivery* e *Service Support*, devido a seu caráter estratégico. Existem várias formas de entrega de

serviços: como as terceirizadas (*outsourced*), parcerias(*partnership*) e as internas (*in-house*). A abordagem deste livro é mais orientada para empresas cuja entrega de serviços é feita de forma interna (*in-house*) [18], ou seja dentro das próprias empresas, pelos seus próprios departamentos de TI.

2.7.6.1 O arcabouço da mudança

A transição de um estágio para outro num processo de evolução, torna-se mais simples quando implementados e instalados os processos ITIL e as tecnologias relacionadas. Cada estágio requer uma combinação de mudanças cujo objetivo final é a completa transformação. Os elementos chaves do arcabouço da mudança podem ser caracterizados como [18]:

Visão e estratégia – Direcionamento global da TI frente ao seu papel dentro dos negócios.

Direção – Objetivos e metas da TI em relação a "Realizar a estratégia".

Processo – Os procedimentos necessários para alcançar as metas e objetivos.

Pessoas – Competências e habilidades necessárias para executar os processos.

Tecnologia – A infra-estrutura que irá viabilizar o processo a ser realizado.

Cultura – Os comportamentos e a atitudes relacionados ao papel da TI dentro dos negócios.

2.7.6.2 Teoria dos processos

Um processo pode ser definido como:

"uma série conectada de ações, atividades, mudanças, etc. Executadas por agentes com a intenção de satisfazer um propósito ou alcançar uma meta."[18]

O controle de processos pode similarmente ser definido como:

"o processo de planejamento e regulamentação, com o objetivo de executar um processo de forma eficiente e efetiva."[18]

Processos, uma vez definidos, devem estar sob controle; uma vez sob controle, podem ser repetidos e tornarem-se gerenciáveis. Graus de controle sobre o processo, e consequentemente métricas podem ser igualmente definidos e gerados no processo de controle e gerência [18].

A saída produzida por um processo deve estar em conformidade com as normas operacionais, as quais são derivadas dos objetivos de negócio. Se os produtos estão em conformidade com o conjunto de normas, o processo pode ser considerado efetivo (pois pode ser repetido, medido, e gerenciado). Se as atividades são executadas com o mínimo de esforço, então o processo pode também ser considerado eficiente. [18]

Métricas resultantes de processos definidos, como mencionado acima, devem ser incorporados em relatórios gerenciais[18].

2.7.7 Application Management

Este livro inicia sua abordagem tratando do alinhamento de TI aos negócios, pela ótica da gerência de aplicações. Nos capítulos decorrentes, trata do ciclo de vida das aplicações, abordando assuntos pertinentes à engenharia de software como: requisitos, modelagem, implementação, instalação, operação e otimização. Na ultima parte, aborda a gerência do conhecimento dentro das organizações, com sua importância e desafios.

Um dado interessante exposto neste livro, o qual serve como pilastra balizadora de sua concepção, é percebido no gráfico abaixo:

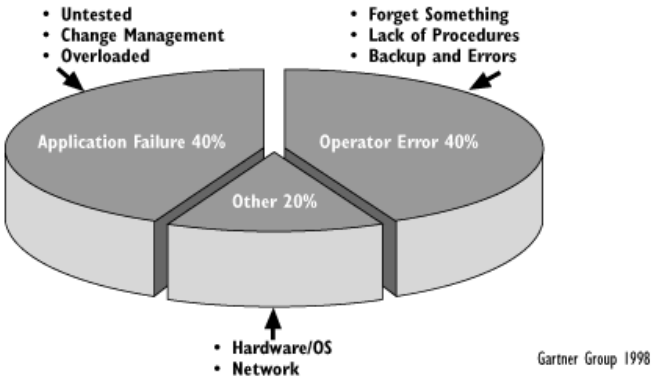

Figura 8 – Paradas na produção – Fonte: ITIL Application Management v.2.1 – Capítulo 1.2

Segundo o gráfico, pode-se perceber que 40% do problemas que afetam os negócios são decorrentes de falhas de *software*, 40% por falhas de operação e 20% por outras falhas incluindo *Hardware*, Sistemas Operacionais e rede [19].

2.7.8 Software Asset Management

Segundo o ITIL, SAM (*Software Asset Management*) pode ser definido por:

"é toda a infra-estrutura e processos necessários para a efetiva gerência, controle e proteção dos ativos de softwares dentro da organização, através de todo seu ciclo de vida.[20]"

O objetivo deste livro é lidar com a responsabilidade envolvida na gerência dos ativos de *software*, vistos como patrimônio da empresa. Neste processo verifica-se a preocupação do modelo de licenciamento, atualização e armazenamento de *software*.

É proposto a institucionalização de um banco de dados onde todos os *softwares* são registrados e também, é definido um local seguro para seu armazenamento.

2.8 Mudanças da versão 3.0

Na sua versão 3.0, a OGC ofereceu todo o conjunto de melhores práticas distribuído em 5 livros que compõem o núcleo do ITIL, os quais [22]:

- *Service Strategy*

- *Service Design*
- *Service Transition*
- *Service Operation*
- *Continual Service Improvement.*

Com esta mudança, a OGC procurou eliminar redundâncias encontradas nos diversos livros, formando um arcabouço mais coeso e mais orientado aos objetivos de negócio.

No geral, a versão 3 tornou mais clara e forte a ligação entre as melhores práticas e os benefícios do negócio. A principal contribuição desta versão está na abordagem baseada no ciclo de vida, ao contrário da abordagem baseada em setores de entrega relatado na versão anterior.[21]

3 AS ONTOLOGIAS

3.1 A arquitetura de 3 camadas

De maneira a promover o reuso e extensibilidade deste trabalho foi elaborada uma arquitetura em camadas, as quais as ontologias se conectam e se estendem formando uma pilha, onde as instancias estão no topo.

A figura 8 ilustra esta arquitetura proposta neste trabalho:

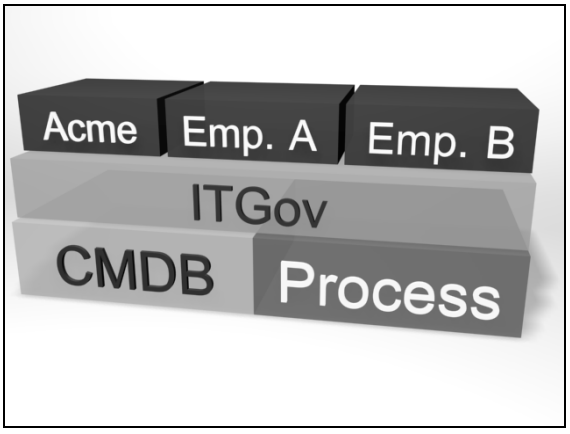

Figura 9 – Arquitetura em 3 camadas proposta

A ontologia *CMDB* define os itens de configuração do ITIL, bem como suas relações. Já, a ontologia *Process* define os elementos de processo, como decisões divisões e atividades.

A ontologia *ITGov* importa as duas ontologias e as estende, provendo as conexões necessárias para a interoperabilidade dos itens de configuração e os elementos de processo. Como exemplo, uma determinada atividade ou processo operacional poderá estar conectada a um agente, sendo este uma pessoa, um sistema, ou outro item de configuração relevante como documentação.

Na camada mais acima, encontram-se as ontologias das empresas que podem importar *ITGov*, e consequentemente por hereditariedade, *CMDB* e *Process*. Desta forma, a ontologia *Acme* conterá indivíduos de todo o arcabouço de governança.

Essa arquitetura possibilita a outras empresas o uso dessas ontologias para especificar seus processos, itens de configuração e unidades de informação, que são os dados que trafegam entre os processos. Assim, empresas parceiras poderão intercambiar informações, agentes e processos, gerando assim sinergia e consequentemente melhoria de desempenho e de resultados.

3.2 Perguntando às Ontologias.

A fim de verificar a funcionalidade do modelo, foi criado a ontologia de uma pequena empresa chamada *Acme Corp.* a qual, alguns processos de negócio e uma infra-estrutura foram definidos. A partir desse modelo foram realizadas algumas consultas, com o intuito de verificar o comportamento do *reasoner* sobre o arcabouço.

Os seguintes processos de negócio foram definidos: *Service Desk* e *Manufacturing*, bem como alguns servidores, funcionários, papeis e sistemas, os quais estão ilustrados a seguir.

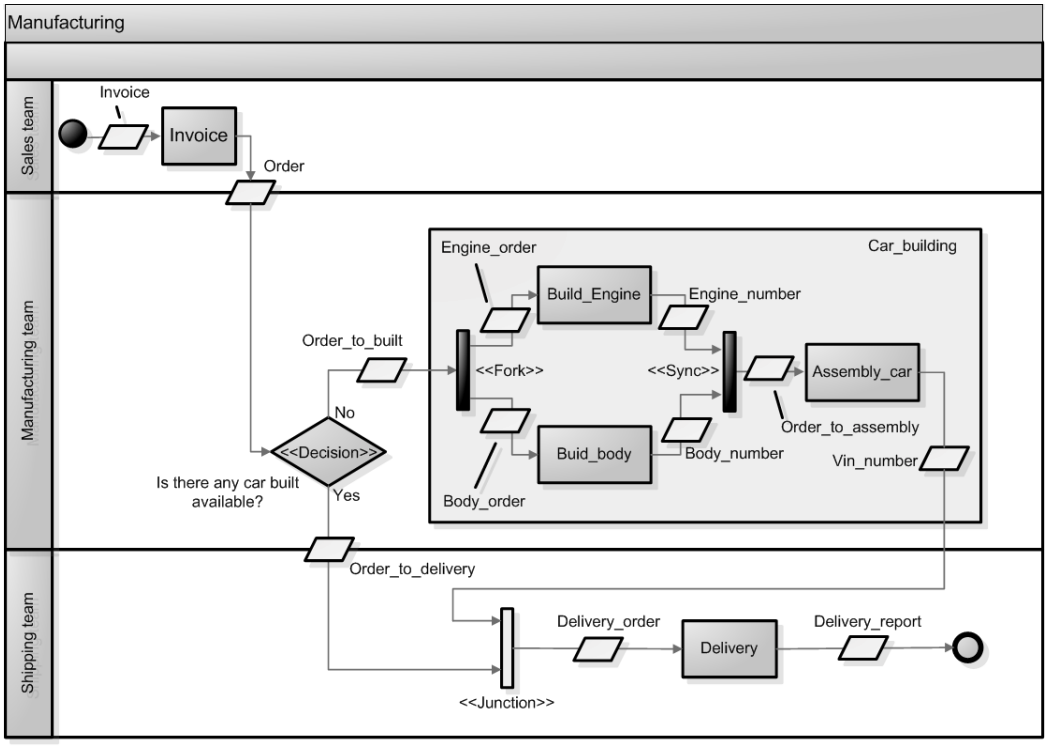

Figura 10 – Processo de manufatura.

O fluxograma da figura 9 mostra um processo simplificado de Manufatura, o qual faz uso dos elementos sintáticos básicos de um processo como: Processo, Atividades, Unidades de informação, Decisões, Separações, Sincronizadores e Junções. A idéia da criação desse processo foi justamente explorar esses elementos.

A figura 10 demonstra o processo de *Incident Handling* do ITIL, o qual define as atividades para o tratamento de um incidente.

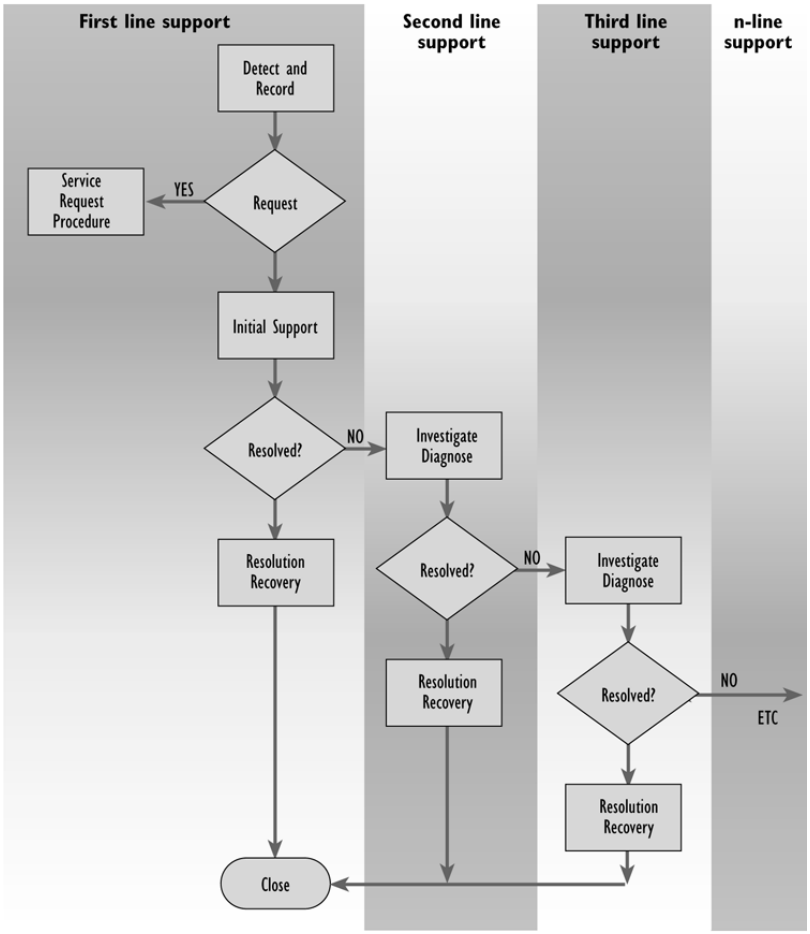

Figura 11 – Ciclo de vida de um incidente – Fonte: ITIL Service Support v2.3 – Capítulo 5.3.1 Incident Handling.

O ITIL sugere o uso de linhas de suporte, porém não obriga que este modelo seja implementado em várias linhas. Nesse caso, para a empresa *Acme Corp*. foi implementado um processo com apenas a primeira linha de suporte com realimentação da atividade *Resolved.* Segue, abaixo o diagrama do processo com as respectivas unidades de informação:

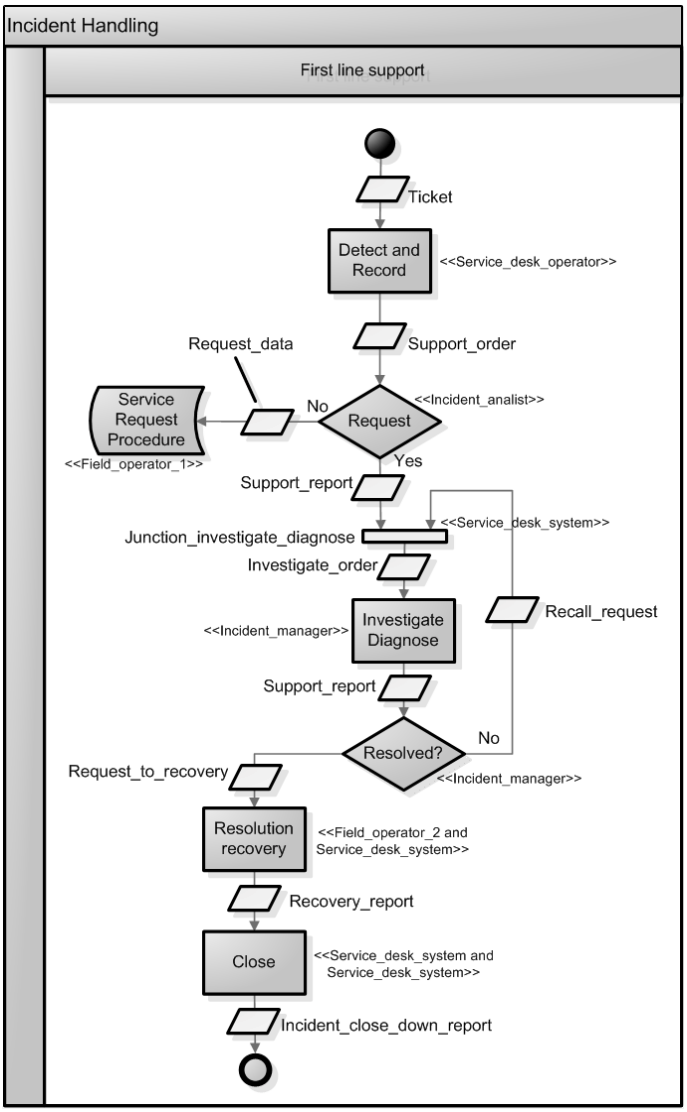

Figura 12 – Processo *Incident Handling Acme Corp*. – Baseado em: ITIL Service Support v2.3 – Capítulo 5.3.1 Incident Handling.

O diagrama da figura 11 representa o processo de *Incident Handling* o qual as atividades são representadas por retângulos; as unidades de informação são representadas por paralelogramas; e os processos operacionais como retângulos com lados arredondados. O círculo preto do topo representa a entrada do processo e o círculo oco da base, a saída do processo.

A seguir, segue um diagrama representando a infra-estrutura de TI da *Acme Corp*.

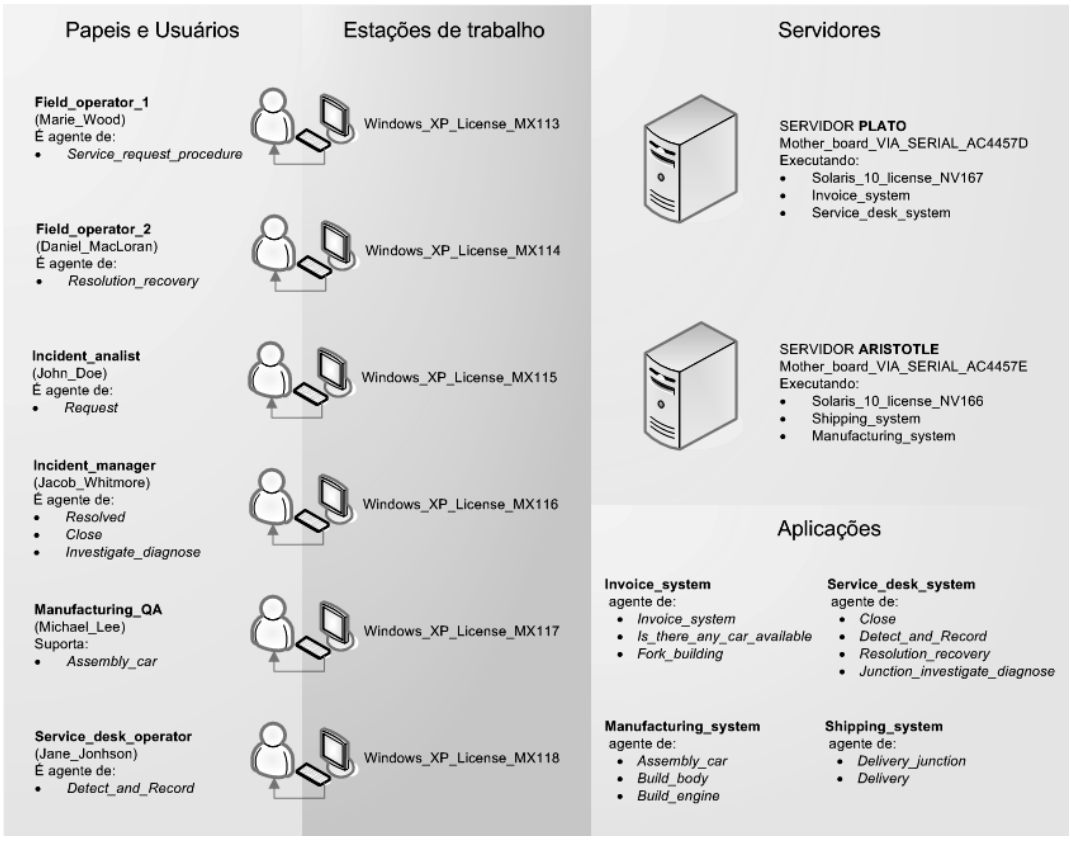

Figura 13 – Infra-estrutura da *Acme Corp*.

No primeiro quadro à esquerda estão os papéis e as respectivas pessoas responsáveis. Logo abaixo do nome, estão as listas das itens de processo que cada papel é responsável. No quadro à direita, na parte superior do diagrama, estão as representações dos servidores, bem como, seus respectivos componentes e programas instalados e rodando. Já na parte direita inferior, encontra-se uma tabela com as aplicações internamente desenvolvidas que dão suporte aos processos de negócio da empresa, juntamente com os respectivos itens de processo.

Uma vez definida a infra-estrutura, processos e suas dependências, como representado na figura 12, é possível lançar mão da linguagem SPARQL para fazer inferências sobre a base de conhecimento.

Para isso, foi construída uma pequena aplicação de testes em Java, a qual instancia o *reasoner Pellet*[38], carrega as ontologias em sua base de conhecimento e

executa dada consulta SPARQL. Nos quadros a seguir estão ilustrados os resultados obtidos com as respectivas instruções SPARQL e comentários.

a) Consulta de todos os elementos de processo que pararão consequentemente se um elemento "*Resolved*" parar:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX process: <http://www.inf.ufsc.br/~marcoshs/ontologies/process.owl#>
PREFIX cmdb: <http://www.inf.ufsc.br/~marcoshs/ontologies/cmdb.owl#>
PREFIX itgov: <http://www.inf.ufsc.br/~marcoshs/ontologies/itgov.owl#>
PREFIX acme: <http://www.inf.ufsc.br/~marcoshs/ontologies/acme.owl#>

SELECT ?x WHERE {

?x a process:Node_abstraction .
acme:Resolved process:has_next_statement ?x .

}
```

Quadro 2 – Consulta de todos os elementos de processo que pararão se um elemento *Resolved* parar.

Resultado da consulta:

```
x
================================
acme:Junction_investigate_diagnose
acme:Close
acme:Resolution_recovery
acme:Investigate_diagnose
acme:Resolved
```

Quadro 3 – Resultado referente ao quadro 2.

É possível alcançar o resultado desta pergunta devido a estruturação de todas as relações que ligam os elementos de processo (*Process_element_abstraction*) serem especializações de *has_next_statement* e sua inversa *is_previous_statement_of*. Assim, por exemplo, os indivíduos instanciados de *Data_container* e *Node_process_abstraction*, como a unidade de informação *Request_to_recovery*, está ligada a *Resolution_recovery* por uma propriedade especializada de *has_next_statement* (no caso *is_input_of*). Dessa forma, a máquina de inferência tem subsídios semânticos para inferir que *Resolution_recovery* é um elemento subseqüente à *Request_to_recovery*, consquentemente *Recall_request.* Da mesma maneira *Close* esta associada com *Request_to_recovery* através da unidade de informação *Recovery_report.*

Como todas as propriedades de *has_next_statement* e *is_previous_statement_of* são transitivas, o efeito cascata é alcançado e consequentemente os elementos: *Junction_investigate_diagnose*, *Close*, *Resolution_recovery*, *Investigate_diagnose* são detectados como sucessores de *Resolved*, inclusive ele próprio devida a recursão promovida pela resposta "não" que se liga na junção que é sua predecessora.

b) Consulta de todos os sub-itens de processo do processo *Incident_Handling*:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX process: <http://www.inf.ufsc.br/~marcoshs/ontologies/process.owl#>
PREFIX cmdb: <http://www.inf.ufsc.br/~marcoshs/ontologies/cmdb.owl#>
PREFIX itgov: <http://www.inf.ufsc.br/~marcoshs/ontologies/itgov.owl#>
PREFIX acme: <http://www.inf.ufsc.br/~marcoshs/ontologies/acme.owl#>

SELECT ?x WHERE {

?x a process:Process_element_abstraction .
?x process:is_aggregated_by acme:Incident_Handling .

}
```

Quadro 4 – Consulta de todos os sub-itens do processo *Incident_Handling*.

Resultado da consulta:

```
x
=================================
acme:Junction_investigate_diagnose
acme:Service_request_procedure
acme:Close
acme:Resolution_recovery
acme:Investigate_diagnose
acme:Resolved
acme:Request
acme:Detect_and_Record
```

Quadro 5 – Resultado referente ao quadro 4.

Alcançar o resultado desta pergunta é possível devido a relação *has_sub_items*, que liga os elementos de processo aos seus sub-elementos ser uma especialização de *aggregates_abstract*, assim estabelece-se a relação de agregação a qual a máquina de inferência se fia para retornar o resultado apresentado no quadro anterior.

c) Consulta de todas as pessoas envolvidas com processo *Incident_Handling*:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX process: <http://www.inf.ufsc.br/~marcoshs/ontologies/process.owl#>
PREFIX cmdb: <http://www.inf.ufsc.br/~marcoshs/ontologies/cmdb.owl#>
PREFIX itgov: <http://www.inf.ufsc.br/~marcoshs/ontologies/itgov.owl#>
```

```
PREFIX acme: <http://www.inf.ufsc.br/~marcoshs/ontologies/acme.owl#>

SELECT ?x WHERE {

?x a cmdb:Person . ?y cmdb:is_dependent_of ?x .
?y process:is_aggregated_by acme:Incident_Handling .

}
```

Quadro 5 – Consulta de todas as pessoas envolvidas com processo *Incident_Handling*.

Resultado da consulta:

```
x
====================
acme:Jane_Jonhson
acme:Daniel_MacLoran
acme:Marie_Wood
acme:Jacob_Whitmore
acme:John_Doe
```

Quadro 6 – Resultado referente ao quadro 6.

Nesta consulta, os elementos envolvidos fazem parte das ontologias *CMDB*, *Process*, *Acme* e *ITGov*. Os pontos chaves da inferência estão nas relações, *is_dependent_of* e *is_aggregated_by*. A inferência *is_dependent_of*, onde todas as demais propriedades que figuram uma relação de dependência são especializadas, juntamente com suas inversas, cujo topo da generalização está *has_dependent*, permite a navegação de dependência pela ontologia *CMDB.* Por outro lado, a propriedade *is_aggregated_by* e sua inversa, permitem a navegação nos níveis de agregação dos elementos de processo.

Para amarrar as ontologias *CMDB* e *Process* é feita uma especialização de *cmdb:has_dependent* pela propriedade *itgov:supports* da ontologia. A *ITGov* liga os itens de configuração do *CMDB* com os itens de processo de *Process.* Descendo o nível de especialização de *itgov:supports* está a propriedade *itgov:is_agent_of.*

Nesse enlace de *CMDB* com *Process* por *ITGov*, toda semântica necessária é fornecida para o dispositivo de inferência deduzir quais itens de configuração do tipo pessoa (Person), dependentes de itens são agregados ao processo de negócio *Incident_Handling*.

d) Consulta de todos os itens de configuração envolvidos diretamente com o processo *Incident_Handling*:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX process: <http://www.inf.ufsc.br/~marcoshs/ontologies/process.owl#>
PREFIX cmdb: <http://www.inf.ufsc.br/~marcoshs/ontologies/cmdb.owl#>
PREFIX itgov: <http://www.inf.ufsc.br/~marcoshs/ontologies/itgov.owl#>
PREFIX acme: <http://www.inf.ufsc.br/~marcoshs/ontologies/acme.owl#>

SELECT ?x WHERE {

?x a cmdb:CI .
?y itgov:is_supported_by ?x .
?y process:is_aggregated_by acme:Incident_Handling .

}
```

Quadro 7 – Consulta de todos os *CIs* envolvidos diretamente com o processo *Incident_Handling*.

Resultado da consulta:

```
x
=========================
acme:Service_desk_system
acme:Service_desk_operator
acme:Incident_analist
acme:Field_operator_2
acme:Field_operator_1
acme:Incident_manager
```

Quadro 8 – Resultado referente ao quadro 8.

Essa consulta funciona da mesma forma como na consulta anterior, porém não são filtrados os itens de configuração pelo tipo *Person*. Assim, o *reasoner* retorna todos os itens de configuração que dão suporte a itens que compõe o processo de negócio *Incident_Handling.*

c) Por fim para averiguar quais processos de negócio são afetados na falha ou degradação de um CI, foi adicionado o *status* ***Degraded*** ao item de configuração ***Mother_board_VIA_SERIAL_AC4457E.*** Os quadros 10 e 11 apresentam respectivamente, a consulta e o resultado de todos os processos de negócio afetados:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX process: <http://www.inf.ufsc.br/~marcoshs/ontologies/process.owl#>
PREFIX cmdb: <http://www.inf.ufsc.br/~marcoshs/ontologies/cmdb.owl#>
PREFIX itgov: <http://www.inf.ufsc.br/~marcoshs/ontologies/itgov.owl#>
PREFIX acme: <http://www.inf.ufsc.br/~marcoshs/ontologies/acme.owl#>

SELECT ?x WHERE {
```

```
?x a process:Business_process .
?y process:is_sub_item_of ?x .
?y cmdb:is_dependent_of ?z .
?z cmdb:has_status cmdb:enum_degraded .

}
```

Quadro 9 – Consulta de todos os processos de negócio afetados por itens degradados.

Resultado da consulta:

```
x
==================
acme:Manufacturing
```

Quadro 11 – Resultado referente ao quadro 10.

Como nas ultimas duas consultas, as propriedades chaves são *cmdb:is_dependent_of,* e *is_aggregated_by*, as quais fornecem sentido para que a máquina de inferência localize os itens de configuração que possuam o *status Degraded* associado. Por sua vez, os itens de configuração que são interdependentes transitivamente, e conjuntamente, dão suporte a itens de processo, que são processos de negócio. Assim, todos os processos de negócio que estão impactados por itens de configuração degradados são retornados..

Esse arcabouço permite efetuar o mais amplo conjunto de consultas à base de conhecimento, a qual, a exaustiva demonstração está fora do escopo deste trabalho, coube aqui apenas evidenciar a versatilidade desta abordagem, demonstrando o resultado de algumas perguntas à base.

3.3 As ontologias em detalhe

3.3.1 A ontologia da Gerência de Configuração do *ITIL - CMDB*

3.3.2 Relações *is-a* da ontologia *CMDB*

Esta ontologia possui seu foco na área dos itens de configuração. Vale ressaltar que não inclui-se no escopo deste trabalho a tarefa de cobrir toda a extensão do processo de Gerência de Configuração.

O diagrama das relações is-a da ontologia dos itens de configuração é ilustrado na figura 12.

Figura 14 – Relações *is-a* das classes da ontologia dos itens de configuração.

Segue agora uma descrição de cada classe da ontologia:

DHS – *Definitive Hardware Store* – define o conceito de local definitivo onde todas as peças sobressalentes devem ser armazenadas. Dentro da empresa podem existir várias dessas unidades distribuídas geograficamente.

CI – Configuration Item – define a abstração de item de configuração, onde quase todas as demais classes são derivadas. Um item de configuração deve ter uma identificação única e possuir versão. Todos os itens de configuração devem ser capazes de se inter-relacionar de alguma forma.

Role – define o conceito papel, o qual as pessoas estarão relacionadas. Por exemplo: “Gerente de configuração”, “Atendente de *helpdesk*”, “Gerente financeiro”, etc.

Application – representa todas as aplicações que a empresa usa, como por exemplo: “SAP”, “Microsoft Office”, “Sistema de Pedidos”, etc.

Third_part – é uma especialização de classe *application* cuja semântica é ser uma aplicação produzida e mantida por uma empresa terceirizada, como por exemplo, sistemas licenciados em regime de locação de licença ou aquisição de licença, como é o caso de sistemas operacionais, SGBDs, suítes de escritório, entre outros.

Internally_developed – esta especialização de *application* tem como semântica representar aplicações desenvolvidas por uma equipe interna da empresa com o intuito de atender as necessidades do negócio.

Media – representa todas as mídias eletrônicas onde as aplicações estão armazenadas. Sua especificação é importante para a rastreabilidade destas cópias de *software*.

System – esta classe representa uma abstração de sistema que é entendida por outras classes especializadas. Um sistema é qualquer conjunto de *hardware* que pode rodar uma ou mais aplicações.

Computer_system – define o que conhecemos como computador, micro-computador, *mainframe*, super-computador. Esta classe é derivada em duas mais especializadas, sendo elas *Server* e *Workstation*.

Server – representa todo computador, cujo uso é distribuído para vários usuários através da rede.

Workstation – define todo computador, cujo uso é feito por apenas um usuário por vez.

Documentation – representa todo artefato de documentação, como arquivos eletrônicos de todas as sortes; documentos físicos como contratos, certidões, certificados; manuais; livros; e em outras mídias como: vídeos, gravações, documentários.

Place – representa lugares físicos onde outros itens de configuração possam estar alocados, como prédios, salas, cidades, paises, andares e regiões geográficas.

Service – define serviços disponibilizados na *intranet* ou na *internet*, por exemplo: serviços de e-mail, *web-services*, páginas http, *date and time*, etc.

Externally_provided – esta classe especializa o conceito de *Service,* de maneira a adicionar a semântica que os serviços definidos por este conceito são providos de forma externa à empresa, ou seja por uma empresa terceira. Podemos exemplificar o uso de servidores externos para o provimento do *web site* da empresa.

Internally_provide – estende o conceito de *Service* de maneira a representar os serviços providos internamente da infra-estrutura da empresa. Podemos citar como exemplo, as páginas *web* da *intranet* que estão apenas disponíveis para os funcionários da empresa.

Hardware_component – define as partes componentes de um *System*, por exemplo: um servidor é composto de placa mãe, memória, *HD*, etc.

Person – representa de forma única as pessoas envolvidas nos processos das empresa, assim como todo item de configuração, na falha de uma pessoa, uma *Role* falhará e consequentemente desencadeará uma falha em cascata, podendo afetar um processo de negócio.

DSL - *Definitive software library* – biblioteca definitiva de *software* – representa um conceito abstrato de biblioteca de *software* cujo objetivo é agrupar todas as cópias mestres de *software* homologadas pela empresa.

Software_master_copy – Representa uma cópia mestre de um determinado *software* o qual foi homologado pelos processos de qualidade da empresa. Desta forma, qualquer instância de *software* deve ser derivada de uma cópia mestre dentro da *DSL*.

Vendor – representa os fornecedores de *software*.

CI_Status – esta classe é um enumerador onde os valores aceitos são: *enum_available, enum_degraded, enum_unavailable, enum_working_under_workaround*. Cada item de configuração tem por definição um status que pode ser um dos quatro permitidos pelo *CI_Status.* Cada estado desses tem um significado próprio, os quais:

enum_available: Quando o item de configuração está disponível.

enum_degraded: Quando o IC está com algum problema e seu funcionamento está degradado.

enum_unavailable: Quando o IC esta completamente indisponível.

enum_working_under_workaround: Quando o IC está funcionando através de alguma medida de contorno. Por exemplo, uma falha geral de energia por um período longo de tempo em um conjunto de servidores que funcione a partir de um gerador de energia elétrica de emergência.

3.3.3 Propriedades da ontologia da Gerência de Configuração do ITIL

Para definir os relacionamentos entre os itens de configuração e demais classes, foram criadas uma série de propriedades, as quais estão listadas e descritas abaixo.

Locates – relaciona um *Place* (Lugar) com um *CI (Item de configuração)*, é transitiva e possui a inversa funcional e transitiva *is_located_in.* Esta relação permite especificar a localização de cada item de configuração.

***has_status* –** Define o estado de cada item de configuração, relacionando o IC com um dos indivíduos que compõe as possibilidades de estado.

has_instance – relaciona *Software_master_copy* com *Application*, possuindo a inversa funcional *is_instance_of* . Esta propriedade permite especificar de qual tipo de *software* uma instancia (instalação) é. Uma vez que pelo ITIL todo *software* em uso dentro da empresa deve ser homologado e ter seus instaladores definitivos guardados em local seguro (*DSL*).

is_owned_by – propriedade transitiva que define que cada *CI* possui alguém responsável (*owner*), o qual é definido por uma Role, ou seja cada item de configuração pode ter um ou mais donos. Sua inversa é a propriedade transitiva *owns.*

***has_dependent* –** propriedade abstrata que não relaciona nenhuma classe. Seu objetivo é ser especializada por uma série de outras propriedades, as quais herdaram sua semântica. A semântica trata de informar que determinada classe tem dependente e conseqüentemente seus indivíduos, assim permitindo ao *reasoner* inferir as relações de dependências. Esta classe, por um outro lado, possui uma inversa que é *is_dependent_of,* cuja funcionalidade é fornecer um caminho de retorno para que o *reasoner* consiga deduzir dependências nos dois sentidos.

runs - propriedade transitiva a qual define que um sistema pode rodar uma aplicação, sendo assim qualquer especialização de *system* pode rodar qualquer especialização de *application.* Essa propriedade é uma especialização de *has_dependent* e possui uma inversa transitiva chamada *is_running_by*, que por sua vez é uma especialização de *is_dependent_of.*

has_application_dependent – está é um propriedade transitiva que liga dois indivíduos *application,* o que significa que uma aplicação pode ter dependência de outras aplicações. Ex: um gerenciador de banco de dados tem dependência de um sistema operacional (que também é uma aplicação), para funcionar. Esta propriedade é

uma especialização de *has_dependent* e possui uma inversa igualmente transitiva chamada *is_application_dependent_of* que por sua vez é uma especialização de *is_dependent_of.*

is_used – propriedade transitiva a qual define que uma aplicação(*application*) é usada por um papel (*role*). Assim um atendente de *helpdesk* usa a aplicação *front end* do atendimento ao cliente. Esta propriedade é uma especialização de *has_dependent* e possui uma inversa igualmente transitiva chamada *uses*, que por sua vez é uma especialização de *is_dependent_of.*

has_service_dependent – propriedade transitiva, a qual relaciona que um serviço possui dependências de um ou mais serviços ou, de uma ou mais aplicações. Por exemplo, o serviço de entrega de e-mails *POP3* pode depender do serviço *SMTP* e ambos são dependentes de um sistema operacional, *FreeBSD*. Esta propriedade é uma especialização de *has_dependent* e possui uma inversa transitiva chamada *is_service_dependent_of* que por sua vez é uma especialização de *is_dependent_of.*

plays – propriedade transitiva a qual relaciona uma pessoa (*Person*) a um ou mais papeis (*Roles*). Por exemplo, um funcionário pode acumular os papeis de Gerente de projetos e Gerente de configuração ao mesmo tempo. Esta propriedade é uma especialização de *has_dependent* e possui uma inversa transitiva chamada *is_played_by*, que por sua vez é uma especialização de *is_dependent_of.*

provides – propriedade transitiva a qual relaciona uma aplicação (Application) a um serviço internamente provido (*Internally_provided*). Por exemplo, o serviço de *POP3* da empresa é provido pela aplicação *Qpopper*. Esta propriedade é uma especialização de *has_dependent* e possui uma inversa transitiva chamada *is_provided_by*, que por sua vez é uma especialização de *is_dependent_of.*

compounds – propriedade transitiva e funcional que está relacionada *Hardware_component* com *System*, ou seja um sistema pode ser composto por

componentes de *hardware* e um componente de *hardware* pode compor apenas um sistema. Por exemplo, um servidor pode ser composto de placa mãe, memória e discos rígidos. Esta propriedade é uma especialização de *has_dependent* e possui uma inversa transitiva não funcional chamada *is_compound_of,* que por sua vez é uma especialização de *is_dependent_of.*

previous_version – propriedade transitiva a qual relaciona duas cópias mestre de *software* (*Software_master_copy*). Seu objetivo é prover uma rastreabilidade entre versões de *software* de maneira que se possa saber todas as versões passadas de um mesmo programa e eventualmente suas versões atualizadas. Esta propriedade possui uma inversa igualmente transitiva chamada *next_version*.

Supplies **–** esta propriedade relaciona uma *Vendor* com um *Software_master_copy*, possibilitando que um distribuídor esteja relacionado com um ou mais produtos (*Softwares*). Esta propriedade possui uma inversa chamada *is_supplied_by*.

3.4 A ontologia *Process*

3.4.1 Relações *is-a* da ontologia de Processos

Como objetivo específico foi proposto a modelagem de uma ontologia base para processos. Essa ontologia foi testada usando-se dois processos de negócios. Abaixo segue as relações *is-a* da ontologia.

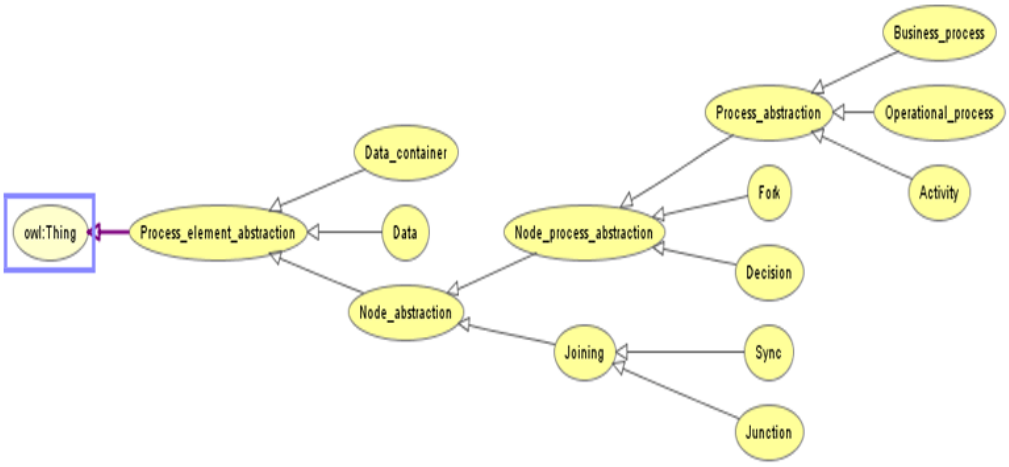

Figura 15 – Relações *is-a* das classes da ontologia de processo de negócio.

Segue agora uma descrição de cada classe da ontologia:

***Process_element_abstraction* –** classe em que todas a classes da ontologia de processos são especializadas, representa qualquer elemento de processo. Esta classe é importante na amarração com a ontologia dos Itens de configuração e com futuras ontologias.

***Node_abstraction* –** desta classe são especializadas todas as classes que representam nós num gráfico de processos.

Node_process_abstraction – define a abstração de nó, a qual são derivadas boa parte das classes que formam os elementos do processo.

***Process_abstraction* –** abstração a qual são especializadas as classes *Business_process, Operational_process,* e *Activity.*

***Business_process* –** desta classe são instanciados os indivíduos que representam os processos de negócio. Como por exemplo: Manufatura, Contabilidade e Suporte.

***Operational_process* –** desta classe são instanciados os indivíduos que representam os processos operacionais. Como por exemplo: Fabricação de carro, Pedidos, Entrega.

Activity – desta classe são instanciados os indivíduos que representam as atividades, como por exemplo: Montagem do Motor, Montagem do Chassi, Montagem do Carro.

Fork – desta classe são instanciados os indivíduos que representam as ramificações de um fluxo onde duas ou mais *Node_process_abstraction* continuam o fluxo do processo em paralelo. Como por exemplo, no processo operacional de montagem de um carro, as atividades montagem do motor e montagem do chassi podem ser feitas ao mesmo tempo.

Sync **-** desta classe são instanciados os indivíduos que representam as unificações por sincronismo ou conjunção, onde o *Node_process_abstraction* seguinte só é disparado quando os dados previstos pelas entradas do *Sync* são satisfeitas. Tem por objetivo sincronizar duas ou mais atividades paralelas como, por exemplo, a atividade de montar o carro que só é ser realizada após o término das atividades de montagem do chassi e do motor.

Junction – desta classe são instanciados os indivíduos que representam as unificações por disjunção. Tem função semelhante ao *Sync*, porém a junção funciona como um operador lógico **OU (**não exclusivo), ou seja, com qualquer das entradas satisfeitas são disparados os dados para o nó seguinte do processo. Normalmente uma junção é útil após separadores condicionais do tipo *Decision.*

Decision - desta classe serão instanciados os indivíduos que representarão as ramificações de um fluxo onde uma condição binária é testada. No caso positivo o fluxo segue pela saída "*yes*", no caso negativo o fluxo segue pela saída "*no*".

Joining – classe abstrata a qual as classes *Junction* e *Sync* são especializações. Sua semântica está no fato do seus elementos terem múltiplas entradas e uma única saída.

Data - desta classe são instanciados os indivíduos que representam as unidades de informação que compõe os indivívuos *Data_container*.

Data_container – desta classe são instanciados os indivíduos que representam os agrupamentos de unidades de informação compostos pelos indivíduos instanciados de *Data*. Os indivíduos de Data_container representam as unidades de informação no processo.

3.4.2 Propriedades da ontologia de Processos

Todas as propriedades da ontologia de processos são transitivas e derivadas de dois grandes blocos semânticos, o primeiro *has_next_statement* e sua inversa *is_previous_statement_of* que dão a semântica do sentido de fluxo do processo; já o segundo, composto por *aggregates_abstract* e sua inversa *is_aggregated_by*, compõe o bloco, cuja semântica, define as relações de composição. As relações de composição podem conter sub-processos, atividades, etc. Estas relações são explanadas em detalhe a seguir.

has_next_statement – propriedade abstrata, cuja funcionalidade é fornecer semântica às demais classes especializadas a partir dela. Todas as propriedades especializadas desta propriedade são conectoras, que ligam do lado onde o processo se inicia para o lado em que processo termina. Sua inversa é a *is_previous_statement_of* que analogamente generaliza todas a propriedades, cujo sentido *domain* para *range* apontam para o inicio do processo. Assim, fornecem uma relação de ordem nas duas direções para todos os elementos conectados.

***aggregates_abstract* –** desta propriedade são especializadas as propriedades *has_piece_of_data_of* e *has_sub_items.* as quais definem as relações de agrupamentos entre os *Data* sob os *Data_container* e *Node_abstraction* sob os *Process_abstraction*. Dessa forma possibilitando definição se sub-processos no caso da informação, dos

agrupamentos das mesmas. Esta propriedade pode estrategicamente especializada por outras propriedades em ontologias que importem a ontologia de processos de maneira a prover semântica de agrupamento para seus indivíduos. A inversa da propriedade *aggregates_abstract* é a propriedade *is_aggregated_by*, sendo que ambas são transitivas.

has_piece_of_data_of – estende a propriedade *aggregates_abstract*, sua função é agrupar indivíduos derivados de *Data* dentro de indivíduos derivados de *Data_container*. Sua inversa é a propriedade *is_data_part_of*, ambas transitivas.

has_sub_items **-** estende a propriedade *aggregates_abstract*, sua função é agrupar indivíduos derivados de *Node_abstraction* dentro de indivíduos derivados de *Process_abstraction.* Esta propriedade permite a definição recursiva de sub-itens de processo. Sua inversa é a propriedade funcional *is_data_part_of*, ambas propriedades são transitivas.

is_abstract_input_of **–** propriedade transitiva abstrata cuja semântica é definir que determinado indivíduo é a entrada de algum item de processo. Sua inversa e igualmente transitiva é *has_input_abstract*.

is_an_input_of **–** propriedade transitiva que relaciona os indivíduos de *Joining* e *Data_container* de maneira a especificar que um *Data_container* é uma entrada de um *Joining*. Como esta propriedade não é funcional, significa que um *Joining* pode ter várias entradas. Porém sua inversa, igualmente transitiva, *has_an_input* é funcional, o que indica que um *Data_container* pode apenas se ligar a um único *Joining.*

is_input_of – liga objetos *Node_process_abstraction* a *Data_container*, especificando assim que todos o indivíduos derivados de *Node_process_abstraction* tenham uma entrada, que é uma unidade de informação. Esta propriedade e sua inversa *has_input* são transitivas e funcionais.

is_abstract_output_of – propriedade abstrata cuja funcionalidade é ser especializada pelas classes que representam saídas de objetos *Node_abstraction*. Sua inversa igualmente transitiva é *has_output_abstract.*

is_an_output_of – sendo uma especialização *is_abstract_output_of*, define as saídas de unidades de informação em um objeto do tipo *Fork.* Esta propriedade é não-funcional e sua inversa *has_outputs* é funcional, ambas são transitivas.

is_no_output_of – define a saída "Não" de um objeto *Decision*, sua inversa é a propriedade *has_output_no* ambas são funcionais e transitivas.

is_yes_output_of – define a saída "Sim" de um objeto *Decision*, sua inversa é a propriedade *has_output_yes* ambas são funcionais e transitivas.

is_output_of – define as saídas de objetos *Process_abstraction* e *Joining*. Sua inversa é *has_output* ambas são transitivas e funcionais.

3.5 A ontologia *ITGov*

Esta ontologia amarra as ontologias *Process* e *CMDB*, sua função é fundamental para que a arquitetura funcione como uma única ontologia coesa.

Nesta ontologia não foi necessário especificar classes, o que nada impede de numa nova versão, novas classes possam fazer parte desta ontologia ampliando sua expressividade.

Basicamente, quatro propriedades são adicionadas a pilha de ontologias de maneira a conectar os itens de configuração ao itens de processo, são elas: *is_supported_by*, *has_agent*, *supports* e *is_agent_of.* A seguir a figura 14 mostra os relacionamento entre as três ontologias, conectadas pelas propriedades de ITGov.

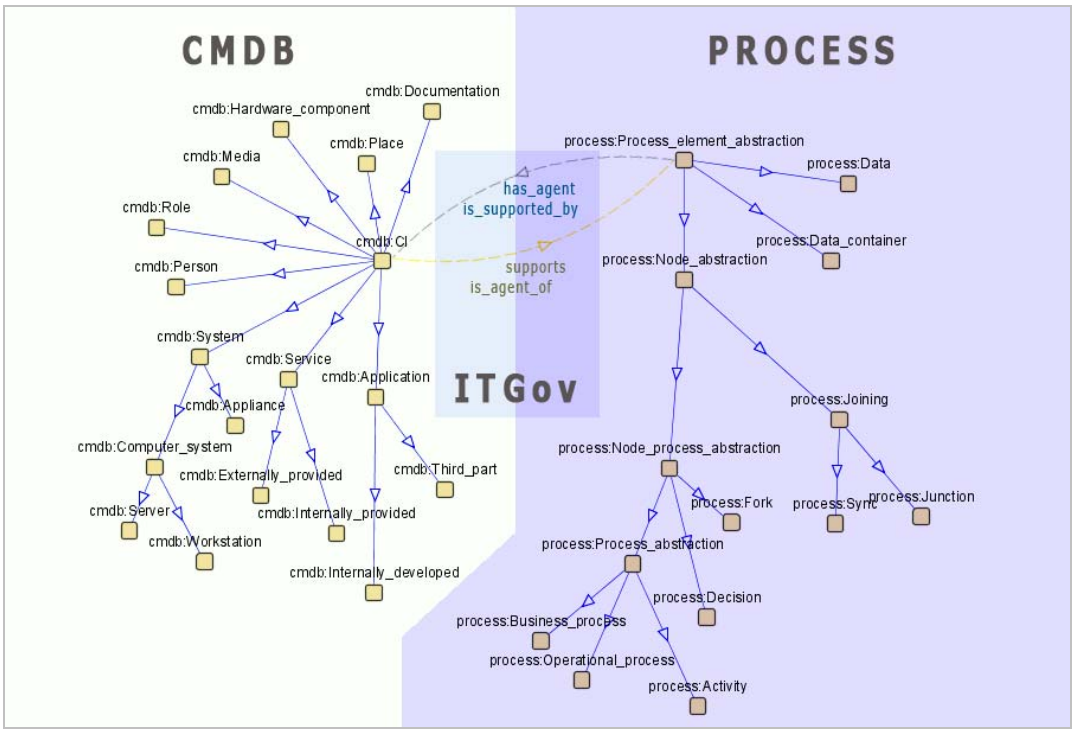

Figura 16 – Relações *is-a* das classes da ontologia de processo de negócio.

Nesta configuração, é possível especificar, por exemplo, que uma atividade é realizada por uma pessoa a qual desempenha algum determinado papel. Nesta mesma atividade é possível associar documentações, equipamentos necessários, etc.

Certas atividades, decisões, e outros elementos de processo podem ser desempenhados por sistemas autômatos, assim é possível determinar que um agente, por exemplo de uma decisão, seja um programa de computador específico que pode conter um modelo capaz da devida tomada de decisão, como por exemplo uma rede neural treinada para tal. Um outro exemplo bem plausível está no uso de uma lista de distribuição de e-mails que é caracterizada como serviço desempenhando o papel de agente de um *Fork* na distribuição de atividades.

3.5.1 Propriedades da ontologia de *ITGov*

supports – propriedade transitiva que relaciona um item de configuração com uma processo de negócio. Esta relação é fundamental para a inferência de quais

processos de negócio são afetados quando um item de configuração falha. Apesar de se poder ligar praticamente todos os itens de configuração aos processos de negócios, isto não é necessário e nem é recomendável, uma vez que impactaria muitas mudanças no caso de uma atualização da infra-estrutura. É recomendável apenas, ligar as aplicações e serviços chaves aos processos de negócios e usar de inferência para a analise de risco. Esta propriedade é uma especialização de *cmdb:has_dependent* e possui uma inversa transitiva chamada *is_supported_by* que por sua vez é uma especialização de *cmdb:is_dependent_of.*

is_agent_of – propriedade transitiva especializada a partir de *supports*. Sua semântica é especificar agentes de derivados de *cmdb:CI* que sejam responsáveis por executar alguma ação em um objeto derivado de *process:Node_abstraction*. A inversa desta propriedade é *has_agent* igualmente transitiva.

4 EXPERIMENTO DE DESEMPENHO I

4.1 Motivação

Com a intenção de verificar a viabilidade do uso de raciocinadores autômatos em ambientes de produção, foram realizados alguns experimentos, onde em um ambiente controlado foi simulado uma infra-estrutura contendo centenas de funcionários, servidores, papéis, serviços, aplicações, etc.

Sobre este ambiente foram executados experimentos com a intenção de verificar o desempenho de raciocinadores efetuando buscas na base de conhecimento criada.

Neste momento foi estudado o *framework* Jena[40] de manipulação de ontologias o qual é possível persistir ontologias em formas de triplas em bancos de dados relacionais e conjuntamente conectar um *reasoner* no *framework* de maneira a usar suas capacidades de inferência.

Para gerar o ambiente para os testes, foi criado um gerador de indivíduos aleatórios para forjar ontologias extencionais contendo milhares de indivíduos interconectados seguindo um padrão lógico de cardinalidade. A triplas desta grande massa de dados foram gravadas em um banco de dados relacional através de um recurso do *framework Jena*.

4.2 Gerador de indivíduos aleatórios

O gerador de indivíduos aleatórios foi concebido em linguagem *Java* 6.0 juntamente com *Jena* 2.5.5 e o *reasoner Peller* 1.5.2. Foi usada uma ontologia, criada especialmente para isso, a qual fornecia meta dados para geração dos indivíduos.

As etapas da criação do *IndGen* (*Individual Generator*) foram as seguintes ilustradas na figura 16:

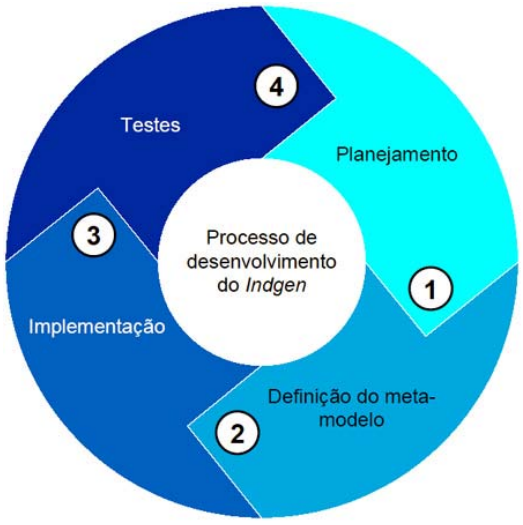

Figura 17 – Processo de desenvolvimento do gerador de indivíduos aleatórios.

Foram executados 3 ciclos iterativos até os testes mostrarem que os resultados gerados atendiam as necessidades do experimento. Os requisitos simplificados do Gerador de indivíduos aleatórios segue abaixo:

R1 - Permitir criar conjuntos de estruturas disjuntas, cujo número fosse parametrizado.

R2 - Permitir a definição de cardinalidade mínima e máxima tanto para *Range* quanto para *Domain* para cada propriedade. No processo de geração as cardinalidades serem geradas aleatoriamente dentro da faixa fornecida.

R3 - Permitir usar dicionários de domínio das classes da ontologia a ser preenchida. O objetivo deste requisito é permitir a geração de massas de dados o mais próximo possível do domínio de mundo real da ontologia.

R4 – Permitir entrar um valor probabilístico para cada elemento das tabelas criadas por R3. O objetivo deste requisito é gerar nuvens de agrupamentos em volta de certos elementos chave dentro de uma ontologia, de maneira a simular comportamentos semelhantes aos encontrados no mundo real.

R5 – Privilegiar sempre a criação de indivíduos interconectados, de maneira a sempre formar um malha conexa. O número de malhas disjuntas é definido por R1.

R6 – Permitir a definição da ordem em que as triplas *RDF* são criadas.

R7 – As ontologias uma vez criadas, devem ser persistidas num banco de dados.

A metodologia adotada no desenvolvimento deste gerador de indivíduos aleatórios assemelha-se a metodologia ágil d e programação - XP[48], devido ao seu caráter de produção de software estável que atenda aos requisitos no prazo mais curto possível. Esta abordagem foi empregada devida as sérias restrições de tempo impostas a este projeto, já o controle de qualidade e testes projeto foi inspirado nas melhores práticas de testes[45, 46, 47].

O processo de geração de indivíduos a partir de uma ontologia dada, é dividido em sete estágios, onde em cada um é criado uma ontologia intermediária contendo as alterações feitas no estágio.

Estagio 1 – Criação da classe *IndGenThing* dentro ontologia X.

Estagio 2 – Transformar todas classes que estendem diretamente *Thing* como especializações de *IndGenThing.*

Estagio 3 – Criação de duas propriedades, *has_metadata* e *indGen_has_next.* A propriedade *has_metadata.*

Estagio 4 – Carga dos indivíduos com os nomes apropriados ao domínio da ontologia. Por exemplo nomes de pessoas, nomes de aplicações, nome de fornecedores de software.

Estagio 5 – Criação de mata-dados para cada indivíduo da ontologia advinda do estágio 4. Nestes meta-dados estão contidas as probabilidades de seleção de cada individuo.

Estagio 6 – Criação de um novo modelo baseado na ontologia original. Também é feita a entrada manual das cardinalidades mínimas e máximas, bem como as probabilidades dos indivíduos carregados no estágio 4.

Estagio 7 – Geração dos indivíduos aleatórios.

O pseudo-código do algoritmo do estágio 7 é descrito no quadro 12:

```
01Inicio;
02  Criação dos modelos;
03    modelMetaSource – Modelo criado a partir do estágio 5 usando um reasoner;
04    modelNew – Modelo criado a partir da ontologia criada no estágio 6;
05  Fim Criação dos modelos;
06  Faça enquanto :numero de iterações;
07    Pega a propriedade inicial nos metadados;
08    Lista todas os metadados da propriedades da ontologia original -> lista de propriedades;
09    Faça enquanto em todos os elementos da lista de propriedades;
10      propriedade da lista -> Prop;
11      Lê a cardinalidade mínima do domínio -> dominioMínimo;
12      Lê a cardinalidade máxima do domínio -> dominioMáximo;
13      Gera um numero aleatório entre o dominioMínimo e dominioMáximo -> nDominio;
14      Lê a cardinalidade mínima do Range -> rangeMínimo;
15      Lê a cardinalidade máxima do Range -> rangeMáximo;
16      Gera um numero aleatório entre o dominioMinimo e dominioMáximo -> nRange;

17      Faça enquanto para todos n;
18        Rotina: Pega um individuo em modelMetaSource -> IndividuoDominio;
19        Cria um novo individuo em modelNew baseado em IndividuoDominio;
20        Faça enquanto nRange;
21          Rotina: Pega um individuo em modelMetaSource -> IndividuoRange;
22          Cria um novo individuo em modelNew baseado em IndividuoRange;
23          Cria a tripla RDF baseado em IndividuoDominio, prop e IndividuoRange;
24        Fim faça enquanto;
25      Fim faça enquanto;

26    Fim faça enquanto;
27  Fim faça enquanto;
28  Grava o modelNew no disco na forma de aquivo .owl
29Fim;

30Sub rotinas;
31  Rotina Pega um individuo em modelMetaSource(classe dada);
32    Lista todos os indivíduos de da classe dada e suas especializações no modelo modelMetaSource -
> ListaFonte;
33    Faça enquanto cada elemento da ListaFonte;
34      Faça até o número de probabilidades do individuo;
35        Adiciona na lista o individuo;
36      Fim faça até;
37    Fim faça enquanto;
```

```
38    Se lista não for vazia;
39      Escolhe aleatoriamente um item da lista -> escolha;
40      Remove o item escolha de modelMetaSource;
41      Retorna a escolha;
42    Senão;
43      Cria um individuo baseado no nome da classe concatenado com um numero seqüencial;
44      Retorna o individuo;
45    Fim se;
46  Fim rotina;
47Fim sub rotinas;
```

Quadro 10 – Pseudo-código do algoritmo do estágio 7 do gerador de indivíduos aleatórios.

4.3 Geração da massa de dados

Uma vez o gerador de indivíduos aleatórios implementado e testado, foi possível iniciar o processo de criação das massas de dados. Para tal, foi optado por usar apenas uma ontologia a qual simplificava a estrutura em camadas. Nesta ontologia a área de processos foi representada apenas por uma classe denominada *Business_process*, no entanto a estrutura completa da ontologia *CMDB* foi mantida.

O objetivo desta simplificação foi a abrandar o impacto do desenvolvimento dos experimentos devida a complexidade acrescida com o uso de quatro *namespaces*.

4.3.1 Definição das cardinalidades e seqüência da criação das triplas.

As seguinte cardinalidades foram especificadas na seqüência da tabela 1. Tais valores foram baseados em entrevistas com dois engenheiros de infra-estrutura com mais de 2 anos de experiência de trabalho em empresas de pequeno e grande porte.

Seq	*Domain*			Propriedade	*Range*		
		Min	Max			Min	Max
1	*System*	50	50	*runs*	*System*	3	3
2	*Person*	150	150	*plays*	*Role*	2	4
3	*Software_master_copy*	50	50	*has_instance*	*Application*	15	15
4	*Business_process*	5	5	*is_supported_by*	*CI*	2	2
5	*Application*	3	3	*has_application_dependent*	*Application*	2	2
6	*Place*	1	1	*locates*	*CI*	50	50
7	*System*	1	1	*is_compound_of*	*Hardware_component*	2	8
8	*Internally_provided*	1	1	*is_provided_by*	*Application*	1	1
9	*Role*	1	1	*owns*	*CI*	1	1
10	*Vendor*	1	1	*supplies*	*Software_master_copy*	1	5
11	*Software_master_copy*	1	1	*previous_version*	*Software_master_copy*	1	1
12	*Role*	1	1	*uses*	*Application*	3	3

Tabela 1 – Definição das cardinalidades e seqüência da criação das triplas.

4.3.2 Carga dos dicionários de nomes

Para a carga dos dicionários foi criado um utilitário que lê todos os arquivos de um diretório específico e os importa como indivíduos para dentro do modelo.

Na tabela 2 é mostrado quais dicionários foram carregados.

Dicionários					
	Qt		Qt		Qt
Business_process	*9*	*Internally_developed*	*169*	*Vendor*	*75*
Documentation	*1*	*Server*	*123*	*Workstation*	*37*
Media	*1*	*Software_master_copy*	*117*	*Third_part*	*1981*
Appliance	*77*	*Externally_provided*	*288*	*Role*	*115*
Hardware_component	*94*	*Person*	*1009*	*Internally_provided*	*68*
DHS	*42*	*Place*	*101*		

Tabela 2 – Dicionários carregados.

O utilitário de carga de dicionários lê cada arquivo dentro diretório, abre este arquivo, lê linha a linha e cria um indivíduo no modelo a partir do nome lido. Obrigatoriamente o nome do arquivo deve ter o mesmo do nome da classe da ontologia.

No processo de importação, foram limpos automaticamente usando expressões regulares os caracteres não permitidos para nomes de URI [28] os quais, incluindo o caractere espaço: : / ? # [ ] @ ! $ & ( ) * + , ; =

4.3.3 A geração

Após todos os preparativos citados nas sessões anteriores, o algoritmo de geração foi acionado. Após 9 horas de processamento, foi pausado o código e observado que o gerador não tinha processado completamente nenhuma das 12 propriedades escolhidas para a geração. Com base nesta morosidade e na possibilidade da inviabilização do experimento, foi notado que ao não usar os dicionários e gerar sequencialmente os nomes dos indivíduos, o tempo de processamento crescia consideravelmente. Então, os arquivos de dicionários foram alterados de maneira a

conter apenas uma linha dentro de cada arquivo e algumas linhas apenas no arquivo *Business_process*. Na tabela 3 é mostrado a nova configuração dos dicionários.

Dicionários					
	Qt		Qt		Qt
Business_process	*9*	*Internally_developed*	*1*	*Vendor*	*1*
Documentation	*1*	*Server*	*1*	*Workstation*	*1*
Media	*1*	*Software_master_copy*	*1*	*Third_part*	*1*
Appliance	*1*	*Externally_provided*	*1*	*Role*	*1*
Hardware_component	*1*	*Person*	*1*	*Internally_provided*	*1*
DHS	*1*	*Place*	*1*		

Tabela 3 – Nova configuração dos dicionários.

Com esta nova configuração o algoritmo de geração de indivíduos aleatórios, após ter usado o único individuo da base *modelMetaSource*, iniciava a criar nomes baseados no nome da classe em questão, seguido de um número serial conforme a instrução 43 do peseudo-código do algorítimo (Quadro 12). Isso evitava o processamento excessivo criado pela seleção aleatória e remoção dos indivíduos que faziam que o *reasoner* precisasse refazer suas estruturas de pesquisa.

Cada ciclo de criação levou aproximadamente 30 minutos, onde era gerada uma base com 450 pessoas, assim simulando uma empresa ou filial com 450 usuários.

Na tabela 4 é mostrado o resultado da primeira geração.

Classe	Qt	Classe	Qt
Business_process	15	*Role*	901
Internally_developed	157	*Externally_provided*	3
Third_part	146	*Appliance*	54
Hardware_component	8	*Server*	50
Person	450	*Workstation*	46
Place	3	*Software_master_copy*	150

Tabela 4 – Quantidade de indivíduos gerados na primeira geração.

Foi tentado aumentar a cardinalidade de maneira a criar uma infra-estrutura de 4500 usuários em uma única iteração, porém após 8 horas de processamento, foi percebido que era mais adequado e factível a criação de blocos de 450 usuários e depois mescla-los de maneira a obter o volume desejado.

Na seqüência, foram criados 10 arquivos contendo indivíduos aleatórios distintos, uma vez que foi criada uma *URI* única para cada um. Os arquivos foram aglutinados em blocos conforme é ilustrado na tabela 5.

Arquivo	1	2	3	4	5	6	7	8	9	10
Individuos	1965	3930	5905	7579	9399	11226	12904	14880	16852	18678
Usuários	450	900	1350	1800	2250	2700	3150	3600	4050	4500
Triplas	8574	17448	26340	34014	41980	50272	51950	60534	69106	77080
Estruturas disjuntas	1	2	3	4	5	6	7	8	9	10

Tabela 5 – Quantidade de indivíduos gerados na primeira geração.

Com os arquivos de massa prontos a próxima etapa foi a carga das triplas no banco de dados. Para isso foi criado mais um utilitário com a função de persistir os arquivos *.owl* em triplas em um banco de dados *PostgresSQL*. Foram criados 10 bancos de dados, cada um refletindo um arquivo.

4.4 Resultados

Uma vez com o bases de dados em triplas das infra-estruturas aleatórias prontas, planejou-se um experimento em que seria submetida uma consulta em cada banco de dados similar à explanada no capitulo 3.2 que retorna os processos de negócios afetados na falha de um determinados item de configuração.

No primeiro experimento o conjunto Jena+Pellet+Postgres retornou uma lista vazia após 10 segundos de consulta.

Foram realizadas outras tentativas, porém todas retornavam uma lista vazia de resultados. A conclusão deste resultado está no fato do raciocinador desistir por *timeout* da busca na base, uma vez que em bases menores as respostas forma obtidas.

5 EXPERIMENTO DE DESEMPENHO II

5.1 Motivação

Após a frustração do resultado alcançado no experimento I, foi planejado um novo experimento de maneira a verificar se o problema do retorno da lista vazia estava no uso do Pellet como *reasoner* ou se haviam outros problemas na preparação do ambiente de testes.

Assim um segundo experimento usando uma consulta mais simples poderia provar que a não resposta estava na complexidade envolvida no processamento de todas as inferências necessárias para o retorno dos processo de negócios afetados por itens de configuração. Assim, uma consulta que apenas solicitava a lista de todas as mídias da base de conhecimento foi criada e aplicada sobre as 10 bases.

De maneira a verificar e isolar o problema, o mesmo experimento foi realizado sem o uso do Pellet associado ao Jena. Na tabela 6 são expostos o resultados alcançados.

Experimento	1	2	3	4	5	6	7	8	9	10
Individuos	1965	3930	5905	7579	9399	11226	12904	14880	16852	18678
Usuários	450	900	1350	1800	2250	2700	3150	3600	4050	4500
Triplas	8574	17448	26340	34014	41980	50272	51950	60534	69106	77080
Estruturas disjuntas	1	2	3	4	5	6	7	8	9	10
Resposta em segs. JENA+PELLET	**7**	**10**	**13**	**29**	**Exception**	–	–	–	–	–
Resp. segundos. JENA	**2**	**2**	**2**	**2**	**3**	**4**	**4**	**3**	**4**	**4**

Tabela 6 – Quantidade de indivíduos gerados na primeira geração.

A partir do experimento 5 o conjunto Jena+Pellet gerou exceções devido ao extrapolarão do limite da área de *heap* do Java.

Com base nos dados da tabela 6 foram plotados alguns gráficos para facilitar a análise dos resultados.

Primeiramente foi criado um gráfico para analisar a linearidade do número de triplas através dos 10 modelos. O gráfico 1 representa esses dados.

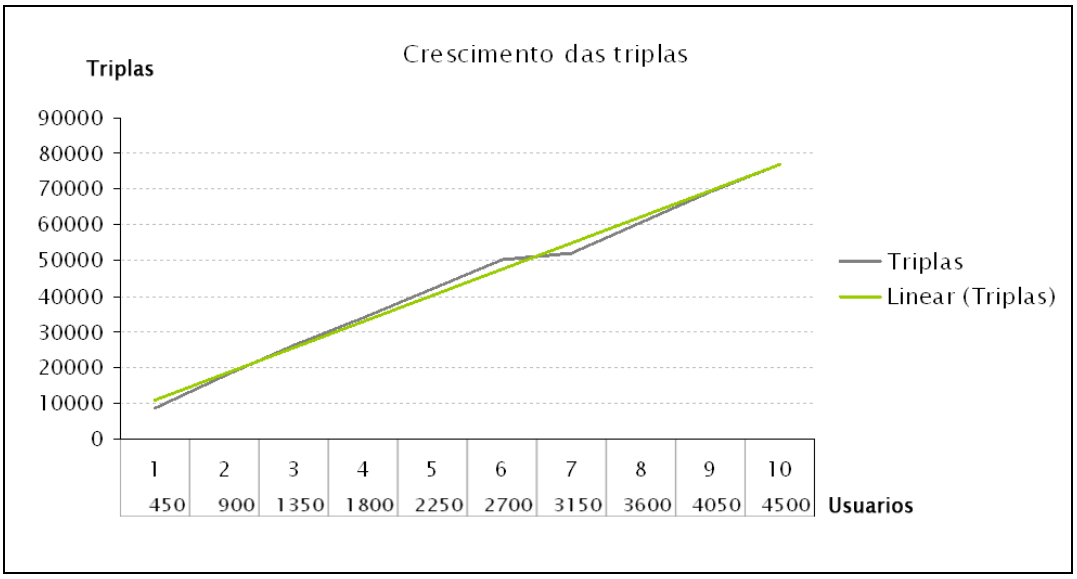

Gráfico 1 – Número de triplas crescendo de forma linear

O gráfico 1 expões a curva de crescimento das triplas confrontada com uma curva de tendência linear. Pelo observado, o crescimento das triplas sucede de forma linear.

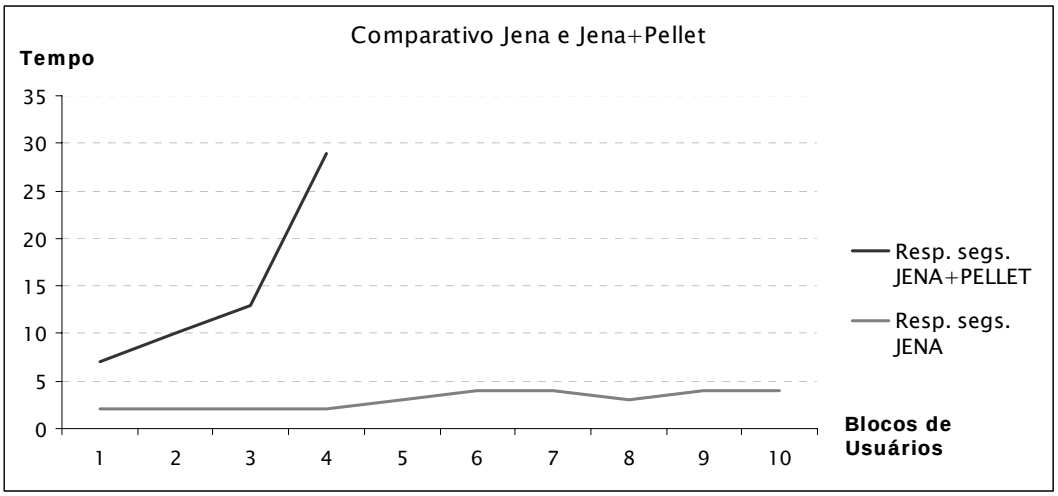

Gráfico 2 – Comparativo Jena e Jena+Pellet

No gráfico 2 é feito o comparativo entre as duas modalidades testadas: Jena com Pellet e somente Jena. É possível perceber a linearidade do comportamento do Jena operando só. Por outro lado, analisando o gráfico do conjunto Jena+Pellet é notável a tendência exponencial da curva.

5.2 Resultados

É possível observar que os dois conjuntos *Jena* e *Jena+Pellet* não realizam suas buscas da mesma forma. Apenas o *Jena,* tem como objetivo fornecer uma interface de acesso a estruturas semânticas, não provendo muito poder de inferência. Desta forma seu desempenho é seguramente superior quando não atrelado a um mecanismo de inferência. Já ligado a um mecanismo de inferência, dentro de suas funcionalidades, está a de procurar todos os resultados possíveis através de inferências em praticamente toda a base de conhecimento. Desta forma ganhando-se em poder de inferência e perdendo-se em desempenho.

5.3 Ferramentas utilizadas

A confecção das ontologias utilizou o *Protegé* versão 3.3.1 [37] e a análise de impacto de falhas de itens de configuração utilizou o *reasoner* Pellet versão 1.5.2 [38], juntamente com o *framework* de manipulação de ontologias *Jena* versão 2.5.5[40].

A implementação das aplicações experimentais usaram as linguagens:

- *Java 6* através da *IDE Eclipse* versão 3.3.2 e o *plugin subclipse*;
- *C#* versão 2.0 usando o *Visual Studio* 2005 como *IDE* e o *TortoiseSVN* para controle de versão.

Todo o controle de versão dos código fonte, ontologias, tabelas e documentação foi feito através de um servidor SVN(Subversion) instalado e configurado especialmente para este fim. Nas estações de trabalho os artefatos tiveram seu controle de versão através do cliente para *Subversion* o *TortoiseSVN*.

Para a tradução de binários *Java* para *assemblies C#* foi feita usando o *IKVM*.

A plotagem dos gráficos, bem como o algoritmo de curva de tendência usados no capitulo 7.1 foram feitos gerados a partir do *Microsoft Excel 2003*.

Para a manipulação de textos na construções dos dicionários, foi usado o *Notepad++* .

O banco de dados usado para a persistência das ontologias foi *PostgreSQL* 8.2.

6 Trabalhos relacionados

Os objetivos específicos deste trabalho envolvem diferentes áreas e no decorrer do desenvolvimento foram criadas várias ferramentas. Assim, este capítulo está subdividido em quatro partes: ontologia da base de dados de gerência de configuração do (*CMDB* do *ITIL*), ontologia de processos, governança de TI e geração de indivíduos para ontologias.

6.1 Ontologia *CMDB* do *ITIL*

Foram pesquisados trabalhos relacionados que implementassem ontologias para o *ITIL* e foi encontrado um trabalho realizado como dissertação de mestrado na Universidade de Brasília[49].

Após uma análise sobre a ontologia e a dissertação, notou-se que o autor não fez nenhum teste de inferência usando sua ontologia. Também foi observado que autor usou nenhum tipo de especialização de suas propriedades, estando todas paralelas. Pelo estudo realizado para a construção das ontologias *CMDB* e *Process*, descobriu-se que esta abordagem plana torna muito árduo o trabalho de raciocínio em uma ontologia construída desta forma.

Outro ponto discutível está na forma como foi definida a classe “pessoa”, onde o autor não a define como sendo um item de configuração, estando assim em desacordo com os conceitos definidos pelo livro “*Service Support*” no capítulo 7.3.7 onde é dito que a informação das pessoas que usam e fazem parte do corpo da empresa devem estar armazenadas no CMDB e ter suas relações com os demais itens de configuração. A

exceção para essa regra esta nos casos de isto implicar em problemas legais, dependendo da legislação do pais onde o sistema estiver operando, o que não se aplica ao Brasil por exemplo.

Um outro aspecto que poderia impactar na adoção em massa da ontologia proposta pelo autor está no fato da mesma ter sido escrita com termos na língua portuguesa, o que limitaria o alcance a somente paises lusofônicos.

Foram pesquisados os termos “cmdb” + “ontology” no portal *IEEEXplore* e no portal da ACM os quais não retornaram nenhum artigo que trouxesse alguma ontologia, seja do ITIL ou de algum subconjunto. Também forma procurados os termos (“itil” + “ontology”) e (“itil” + “ontologies”), todos sem sucesso.

6.2 Ontologia *Process*

Definições de processo podem ser encontradas nos sites da *WFMC - Workflow Management Coalition* e no site da *OMG – Object Management Group*. Porém estas duas organizações não provém uma ontologia que defina processos, apenas especificações de notação e *Web Services.*

6.3 Governança de TI

Existem muitos trabalhos acerca deste tema, podendo ser citado aqui os trabalho de ALVES e RANZI[50] , VETTER[51] e SODRÉ e SOUZA [52].

6.4 Geração de indivíduos para ontologias.

Não foi encontrado nenhum trabalho na *Web* ou artigos que descrevessem um aplicativo para geração de massas de indivíduos usando dicionários.

6.5 Outros trabalhos relacionados

Existem outros trabalhos relacionados relevantes como o trabalho de VIEIRA[52] e MORAES[53] onde suas contribuições foram inspiradoras deste trabalho.

7 CONCLUSÕES

Os resultados obtidos pelo *Jena* funcionando sem a máquina de inferência permitiram observar uma possibilidade a adoção do modelo de triplas persistidas em banco de dados relacionais como uma opção a tradicional modelagem ER. Obviamente, este estudo não tem profundidade suficiente para uma afirmação assertiva sobre este tema, porém tem seu valor agregador no caminho evolutivo desses modelos e trás subsídios para a discussão.

Sobre a arquitetura em camadas é possível afirmar que sua concepção e implementação foram bem sucedidas uma vez que dentro da escala de alcance do raciocinador, os resultados das consultas alcançaram os objetivos propostos por este trabalho e em determinado momento superam os mesmos quando vislumbrado a expansibilidade do modelo, bem como a possibilidade deste poder responder a perguntas não formuladas originalmente para este trabalho.

Pela ótica do uso das ferramentas e demais artefatos que estão sendo desenvolvidas para a materialização da *Web* Semântica, através deste trabalho é possível observar que a pesquisa e desenvolvimento das mesmas vêm gerando bons frutos. Uma vez observada a estabilidade e funcionalidade de ferramentas como o *Protégé* e o *Jena*, claro, sem ofuscar o poder de inferência do *Pellet*, que dentro de um escopo definido, pode atuar de forma estratégica na execução de tarefas que exijam inteligência de máquina.

Como já confirmado historicamente, a formação de indústria para o desenvolvimento das tecnologias advindas de teorias, metodologias e modelos é

fundamental. Assim fazer o uso destas tecnologias consideradas novas, aliadas a sólidos arcabouços, onde há anos de conhecimento armazenado, pode ser uma escolha inteligente a qual poderá fazer que estes modelos se desenvolvam no interior de empresas interessadas. Em contrapartida tais empresas ao aproveitar os benefícios que esta abordagem oferece, estarão alargando e fortalecendo seus passos na corrida junto ao seus concorrentes.

Em *ITGov* é facilmente possível estender os itens de configuração, bem como os itens de processo, afim de criar uma propriedade que armazenasse os valores de custo por hora, minuto, dias de cada elemento, degradação e tempo médio de falhas. Assim, fazendo uso desse arcabouço na avaliação de custos, análise de obsolescência e degradação, promovendo ainda mais seu uso como ferramenta de alinhamento de negócios e tecnologia.

Analisar o *reasoner Owlgres*, que foi lançado em 7 de maio, o qual seu fornecedor sendo o mesmo do Pellet menciona ser uma máquina de inferência de alto desempenho, projetada para operar em grandes bases de conhecimento. Como continuidade deste trabalho, caberia submeter este *reasoner* às mesmas condições experimentadas pelo seu irmão e averiguar os resultados.

O uso da ontologia de processo para a criação de todos os processos do ITIL assim como foi feito com *Incident Handling*. Desta forma ter todos os processos definidos.

Também é possível o uso do arcabouço proposto neste trabalho de ser usado para modelagem de processos de outros arcabouços de qualidade com o COBIT[52], MPS.BR e outros.

A partir dos processos definidos, é possível criar aplicações que sigam o fluxo de informações, assim é possível a criação de uma aplicação de *Workflow* baseado nesta abordagem.

7.1 Trabalhos futuros

Em *ITGov* é facilmente possível estender os itens de configuração, bem como os itens de processo, afim de criar uma propriedade que armazenasse os valores de custo por hora, minuto, dias de cada elemento, degradação e tempo médio de falhas. Assim, fazendo uso desse arcabouço na avaliação de custos, análise de obsolescência e degradação, promovendo ainda mais seu uso como ferramenta de alinhamento de negócios e tecnologia.

Analisar o *reasoner Owlgres*, que foi lançado em 7 de maio, o qual seu fornecedor sendo o mesmo do Pellet, menciona ser uma máquina de inferência de alta performance, projetada para operar em grandes bases de conhecimento. Como continuidade deste trabalho, caberia submeter este *reasoner* às mesmas condições experimentadas pelo seu irmão e averiguar os resultados.

O uso da ontologia de processo para a criação de todos os processos do ITIL assim como foi feito com *Incident Handling.* Desta forma ter todos os processos definidos.

Também é possível o uso do arcabouço proposto neste trabalho de ser usado para modelagem de processos de outros arcabouços de qualidade com o COBIT[52], MPS.BR e outros.

A partir dos processos definidos, é possível criar aplicações que sigam o fluxo de informações, assim é possível a criação de uma aplicação de *Workflow* baseado nesta abordagem.

A ontologia CMDB proposta não define mecanismos de representação de dispositivos que operem de forma redundante como por exemplo servidores gêmeos que operam como backup. Assim, um possível trabalho futuro é definir semanticamente a redundância de itens de configuração de maneira ao raciocinador inferir que na falha de

um item mestre um backup entra em operação e o serviço não é degradado ou afetado, criando-se assim mais um estado para um IC que seria *operando no backup.*

8 REFERÊNCIAS

[1] LIPMAN, F. D.; LIPMAN, L. K. **Corporate Governance Best Practices: Strategies for Public, Private, and Not-for-Profit Organizations**. John Wiley & Sons, 2006.

[2] BROWN, L. D.; CAYLOR, M. L. **Corporate Governance and Firm Performance**. Georgia State University, 2004.

[3] WEILL, P.; ROSS, J. W. **IT Governance: How Top Performers Manage IT Decision Rights for Superior Results**. Harvard Business School Press, 2004.

[4] WEILL, P.; ARAL, S. **How IT Savvy is Your Enterprise? Self Assessment and Benchmarking.** MIT Sloan Center for Information Systems Research, 2006.

[5] Organization for Economic Cooperation and Development, Directorate for Financial, Fiscal and Enterprise Affairs. **OECD Principles of Corporate Governance**, OECD, 1999.

[6] KHOSHAFIAN, S. **Business Process Management - Service Oriented Enterprises**. Auerbach Publications, 2007.

[7] Sarbanes-Oxley Act of 2002, Pub. L. No. 107-204, 116 Stat. 745 (codified as amended in scattered sections of 15 U.S.C.).

[8] TANIAR, D.; RAHAYU, J. W. **Web Semantics and Ontology.** IGI Publishing, 2006.

[9] Office of Government Commerce – OGC. **ITIL Best Practices - Service Delivery v.2.2** – Crown, 2001.

[10] LACY, S.; HARROW, R. **ITIL Best Practices - Service Support**. Versão 2.3 – Crown 2001.

[11] SANTOS, M. H.; PERANTUNES, H. **Web Semântica e Ontologias - uma abordagem sistêmica**. Agosto, 2006.

[12] World Wide Web Consortium – W3C. **Semantic Web Tools**. MIT- ERCIM-Keio, 2008.

[13] ITIL Central. **What is ITIL**. Disponível em http://itsm.fwtk.org/. Acesso em 10 de novembro de 2007.

[14] PILTZECKER, T. **Microsoft Operations Manager 2005 Essentials - How to Cheat at Managing Microsoft Operations Manager 2005.** Syngress Publishing, 2006.

[15] Office of Government Commerce – OGC. **ITIL ICT Infrastructure Management v.2.2** – Crown, 2002.

[16] Office of Government Commerce – OGC. **ITIL Security Management v1.0**. Crown, 2004.

[17] Office of Government Commerce – OGC. **Management of Risk v.2.0**. Crown, 2002.

[18] Office of Government Commerce – OGC. **ITIL Planning to Implement Service Management v.2.2.** Crown, 2002.

[19] Office of Government Commerce – OGC. **ITIL Application Management v.2.1.** Crown, 2002.

[20] Office of Government Commerce – OGC. **ITIL Software Asset Management v.1.0**. Crown, 2003.

[21] Office of Government Commerce - OGC. **Service Management – ITIL**. Versão 3. Crown, 2004.

[22] RUDD, C. **The IT Infrastructure Library An Introductory Overview of ITIL V3**. Versão 1.0 – itSMF, 2004.

[23] WEILL, P.; ROSS, J. W. **IT Governance: How Top Performers Manage IT Decision Rights for Superior Results.** Harvard Business School Publishing, 2004.

[24] BERNERS-LEE , T.; Hendler, James.; Lassila, Ora. **The Semantic Web-A new form of Web content that is meaningful to computers will unleash a revolution of new Possibilities.** Scientific American, 2001.

[25] CARDOSO, J. **Semantic Web Services: Theory, Tools and Applications**. IGI Publishing, 2007.

[26] World Wide Web Consortium – W3C. **W3C Semantic Web Activity.** MIT-ERCIM-Keio, 2001.

[27] The Unicode Consortium. **The Unicode Standard 5.0**. 5ª edição. Addison-Wesley Professional, 2006.

[28] BERNERS-LEE, T. **Uniform Resource Identifiers (URI): Generic Syntax**. IETF - Internet Engineering Task Force, 1998.

[29] World Wide Web Consortium – W3C. **Extensible Markup Language (XML).** MIT- ERCIM- Keio, 2006.

[30] World Wide Web Consortium – W3C. **Namespaces in XML**. MIT- ERCIM-Keio, 2006.

[31] World Wide Web Consortium – W3C. **XML Schema Requirements.** MIT-ERCIM- Keio, 1999

[32] World Wide Web Consortium – W3C. **Resource Description Framework (RDF): Concepts and Abstract Syntax**. MIT- ERCIM- Keio, 2004.

[33] World Wide Web Consortium – W3C. **RDF Vocabulary Description Language 1.0: RDF Schema.** MIT- ERCIM- Keio, 2004.

[34] World Wide Web Consortium – W3C. **OWL Web Ontology Language Semantics and Abstract Syntax.** MIT- ERCIM- Keio, 2004.

[35] GRUBER, T. R. **Toward Principles for the Design of Ontologies Used for Knowledge Sharing**. Stanford Knowledge Systems Laboratory, 1993.

[36] World Wide Web Consortium – W3C. **What is an Annotation?.** MIT- ERCIM-Keio, 2000.

[37] Stanford Center for Biomedical Informatics Research e Stanford University School of Medicine. **Protégé**. Disponível em http://protege.stanford.edu/. Acesso em novembro de 2007.

[38] SIRIN, E.; PARSIA, B.; Grau, Bernardo C.; Kalyanpur, Aditya.; Katz, Yarden. **Pellet: A practical** OWL-**DL reasoner**. Journal of Web Semantics, 2007.

[39] LAWES, A. **ITIL & itSMF – next Steps in the global success stor**y. In: Proc. of the SwissICT and itSMF, Conference on IT Service Management. Zürich, 2003.

[40] HP - Labs Semantic Web Research. **Jena – A Semantic Web Framework for Java.** Hewlett-Packard Development Company, 2008.

[41] World Wide Web Consortium – W3C. **SPARQL Query Language for RDF.** MIT- ERCIM- Keio, 2008.

[42] SIRIN, E.; PARSIA, B. **SPARQL-DL: SPARQL Query for OWL-DL**. Owled - Third International Workshop, Innsbruck, 2007.

[43] DING, L.; FININ, T. **UMBC Semantic Web Reference Card - v2 (A4).** UMBC, 2005.

[44] Postgres – TODO

[45] The Institute of Electrical and Electronics Engineers. IEEE Std 829: **Standard for Software Test Documentation**. New York: IEEE Computer Society, September, 1998.

[46] BEIZER, B. **Black-Box Testing: techniques for funcional testing of software and systems**. John Wiley & Sons, 1995.

[47] BLACK, R. **Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional.** John Wiley & Sons, 2007.

[48] HUNT, J. **Agile Software Construction.** Springer, 2006.

[49] FERREIRA, R. G. **Um Framework de Alinhamento Ontológico entre a TI e o Negócio de uma Organização.** Universidade de Brasília – UnB, 2007.

[50] ALVES, E. M; RANZI, T.A.D. **Governanança de TI: Avaliação de Maturidade do COBIT em uma Empresa global.** UFSC-CTC-INE, 2006.

[51] VETTER, F. **Um Estudo sobre Governança em Tecnologia de Informação com enfoque a Qualidade da Prestação de Serviços ao Cliente.** UFSC-CTC-INE, 2007.

[52] SODRÉ, M. G; SOUZA, S. M. **Uma Análise Comparativa de Metodologias para Governança de Tecnologia da Informação – ITIL e COBIT**, UFSC-CTC-INE, 2007.

[53] AGUIAR, A. V. **Mitos e Verdades Sobre a Web Semântica.** UFSC-CTC-INE, 2006.

[54] MORAES, M. **Reengenharia do ONTOCOVER: Uma Biblioteca Java Para Manipular Ontologias em Aplicações da Web Semântica.** UFSC-CTC-INE, 2007.