DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TESIS DOCTORAL

Improving IT Service Management using an Ontology-Based and Model-Driven Approach

Autora:

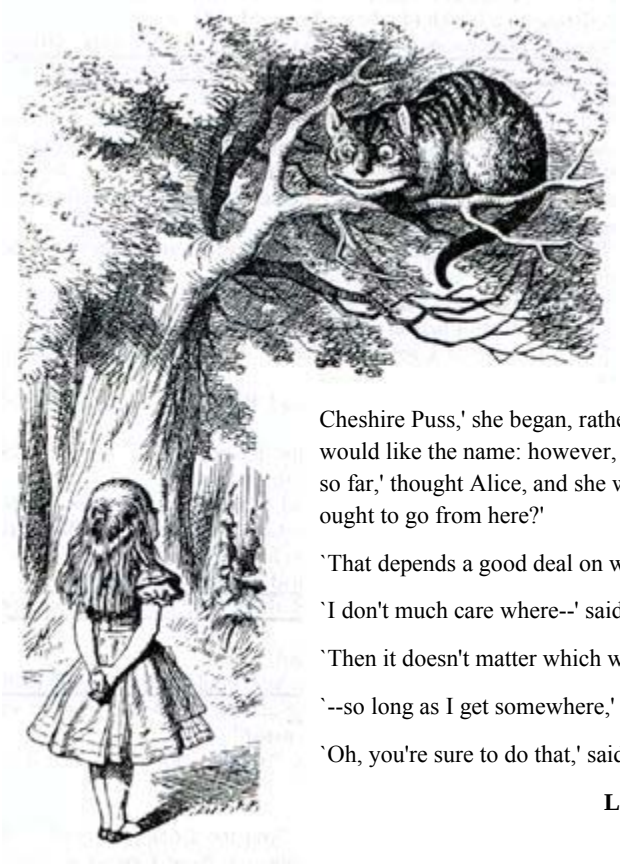
María Cruz Valiente Blázquez

Directores:

Dr. Daniel Rodríguez García
Dra. Cristina Vicente Chicote

Alcalá de Henares, abril de 2011

To my parents, my husband Alfonso and our daughter Natalia



Cheshire Puss,' she began, rather timidly, as she did not at all know whether it would like the name: however, it only grinned a little wider. `Come, it's pleased so far,' thought Alice, and she went on. `Would you tell me, please, which way I ought to go from here?'

`That depends a good deal on where you want to get to,' said the Cat.

`I don't much care where--' said Alice.

`Then it doesn't matter which way you go,' said the Cat.

`--so long as I get somewhere,' Alice added as an explanation.

`Oh, you're sure to do that,' said the Cat, `if you only walk long enough.

Lewis Carroll, *Alice's Adventures in Wonderland*

(Illustration by Sir John Tenniel)

Resumen

La adopción de marcos de trabajo de mejores prácticas que permiten la integración de las Tecnologías de la Información (TI) con el negocio, ayuda a las organizaciones a crear y compartir procesos de gestión de servicios de TI. Sin embargo, las guías y modelos publicados suelen especificarse en lenguaje natural o con representaciones gráficas que carecen de la semántica computacional necesaria para poder automatizar su validación, simulación e incluso su ejecución.

En esta tesis se presenta Onto-ITIL, una propuesta basada en ontologías y en el enfoque de desarrollo de software dirigido por modelos que captura las mejores prácticas ofrecidas por ITIL (del inglés *Information Technology Infrastructure Library*), y destinada a facilitar la prestación de servicios de TI. El objetivo de Onto-ITIL es ayudar a los expertos del dominio a modelar e implementar procesos de gestión de servicios de TI evitando ambigüedades semánticas y contradicciones. La formalización de los procesos de gestión de servicios de TI en términos de ITIL constituye un primer paso para cubrir la brecha que se da entre el negocio y las TI.

Para definir las ontologías se ha utilizado OWL (del inglés *Web Ontology Language*). Adicionalmente, se ha definido un conjunto de reglas basadas en SWRL (del inglés *Semantic Web Rule Language*) que permiten enriquecer la ontología con una serie de restricciones semánticas y de reglas de inferencia de conocimiento. Por último, la definición de un conjunto de consultas basadas en SQWRL (del inglés *Query-Enhanced Web Rule Language*) permite recuperar conocimiento obtenido con OWL e inferido a través de las reglas SWRL.

Además de formalizar los procesos de gestión de servicios de TI en base a las buenas prácticas consideradas por ITIL, Onto-ITIL también permite compartir, reutilizar e intercambiar las especificaciones de dichos procesos a través de mecanismos automatizados que proporcionan ciertos marcos de trabajo de comercio electrónico, como por ejemplo, ebXML.

Mediante la adopción del enfoque MDE (del inglés *Model-driven Engineering*), se ha utilizado un DSL (del inglés *Domain Specific Language*) basado en la ontología Onto-ITIL que sirve para implementar sistemas de información basados en flujos de trabajo

que dan soporte a los *Sistemas de Gestión de Servicios de TI* (SGSTI). Los modelos que se obtienen a partir de este lenguaje de modelado se pueden considerar modelos de alto nivel que han sido enriquecidos con conocimiento ontológico, y que están definidos exclusivamente en términos de lógica de negocio, es decir, que no presentan ningún aspecto arquitectónico o de plataforma de implementación. Con lo cual, de acuerdo con la arquitectura en cuatro capas propuesta por el OMG (del inglés *Object Management Group*), estos modelos se encontrarían a nivel CIM (del inglés *Computation Independent Model*).

En resumen, la propuesta presentada en esta tesis permite: (i) formalizar el conocimiento asociado a los sistemas de gestión de servicios de TI en base a ontologías que recogen las buenas prácticas consideradas por ITIL; (ii) modelar la semántica de las actividades que definen los procesos de gestión de servicios de TI en forma de flujos de trabajo; (iii) generar de manera automática modelos de requisitos de alto nivel para implementar sistemas de información que se necesitan para dar soporte a dichos procesos; y (iv) a partir de los modelos anteriores, obtener modelos de más bajo nivel (llegando incluso al código de las aplicaciones) a través de transformaciones automáticas de modelos.

La investigación llevada a cabo en esta tesis se ha validado mediante de la implementación de un caso de estudio real proporcionado por una compañía española que ofrece servicios de TI.

Abstract

Best practice frameworks, focused on the integration of business and *Information Technology* (IT), help organizations create and share effective *IT Service Management* (ITSM) processes. However, service management guidelines and models are commonly specified using natural language or graphical representations, both lacking the computational semantics needed to enable their automated validation, simulation or execution.

This thesis proposes Onto-ITIL, an ontological and model-driven approach that captures the best practices provided by the *IT Infrastructure Library* (ITIL) framework. Onto-ITIL aims to help domain experts to model and implement ITSM processes avoiding semantic ambiguities, uncertainties and contradictions. Formalizing ITSM processes in terms of ITIL is as a first step towards bridging the current gap between business and IT.

Onto-ITIL has been defined using the *Web Ontology Language* (OWL) and has been enriched with a set of rules defined using the Semantic Web Rule Language (SWRL) to provide semantic constraints and knowledge inference. The definition of a set of queries based on the *Semantic Query-Enhanced Web Rule Language* (SQWRL) enables the retrieval of knowledge from OWL and inferred by the SWRL rules.

Onto-ITIL not only enables the formal specification of ITSM processes, but also to share, reuse, and interchange these specifications by automated means using e-business frameworks such as ebXML.

Adopting a *Model-driven Engineering* (MDE) approach, a *Domain Specific Language* (DSL), based on Onto-ITIL, is used in order to implement workflow-based information systems that underpin *ITSM Systems* (ITSMSs). The resulting high-level models enriched with ontological knowledge are defined just in terms of the business logic, without any architectural or platform-specific consideration. That is, according to the OMG's four-layered architecture, the ontology-based workflow models could be placed at a *Computation Independent Model* (CIM) level.

In summary, the approach presented in this thesis aims: (i) to formalize the knowledge associated to ITSMSs in terms of ontologies that gather ITIL best practices; (ii) to model the semantics of the activities associated to ITSM processes in terms of workflows; (iii) to automatically generate the high-level requirements models of the information systems needed to support these processes; and (iv) from the latter, to obtain lower-level models (and eventually code) by means of automated model transformations. The proposed approach has been validated using a real case study from a Spanish IT service provider.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "Human beings obtain the most of their capacity when they are completely conscious of their circumstances." *Meditaciones del Quijote*
>
> José Ortega y Gasset (1883-1955), *Spanish philosopher*

In this chapter we identify the problems that we want to solve. Then we describe motivations and objectives of this work, and the research method. Finally, we outline the structure of the thesis.

1.1 Problem Statement

Nowadays, *IT Service Management* (ITSM) is considered a fundamental and competitive task in organizations. The interest in ITSM has been motivated not only because of competition amongst companies, but also due to the interest organizations have in integrating *Information Technology* (IT) with the business they belong to as a corporative strategy. The best practices documented in several ITSM frameworks have motivated many companies to move from a product-oriented organization to a service-oriented one. A process-based approach has been considered the most efficient and effective way to achieve this [itSMF, 2008]. An adequate process-based ITSM will allow the companies: (i) to continuously improve their processes; (ii) to achieve a significant improvement in perceived quality by customers; and (iii) to improve their IT strategy.

Information is one of the most important assets for all organizations. Thus, IT services are decisive for good knowledge management, efficient decision making, and planning actions for the company [de Pablos et al., 2008].

IT services are generating a cultural change in organizations since they allow integrating information systems in the business. IT services are required to evolve and rapidly adapt to the different companies' new needs and technologies, even more taking into account the proliferation of shared IT services and *outsourcing*. However, the strategic opportunity that new technologies bring to companies is simultaneously the cause of the difficulties that arise in their management.

A recognized solution to this problem is to implement an *ITSM System* (ITSMS) based on the *ISO/IEC 20000:2005 Information technology – Service management* standard. An ITSMS is a collection of interrelated and coordinated rules, principles and activities, structured in term of processes [Nextel, 2010]. The ISO/IEC 20000 standard [ISO/IEC, 2005a] [ISO/IEC, 2005b] establishes a rule for all of the organizations that offer IT services, not only to external customers but internal customers as well. Through the ISO/IEC 20000 standard, the internal and external suppliers for IT services are challenged to prove that their *Service Management* processes guarantees the quality their customers demand. ISO/IEC 20000 consists of two parts, under the general title '*Information technology — Service Management*':

- *Part 1 – Specification* [ISO/IEC, 2005a]. This part defines the requirements for IT service providers to implement an ITSMS in order to deliver managed IT services of an acceptable quality for their customers.
- *Part 2 – Code of Practice* [ISO/IEC, 2005b]. This part represents an industry consensus on quality standards for ITSM processes. These service management processes deliver the best possible service to meet business needs of customers, within agreed resource levels (i.e. service that is professional, cost-effective and with risks which are understood and managed).

IT service providers tend to find many difficulties to implement the ISO/IEC 20000 standard, especially *Small and Medium-sized Enterprises* (SMEs), which count on limited resources. However, there exist easier-to-adopt best practices, such as those defined in the *Information Technology Infrastructure Library* (ITIL) [ITIL website] or in the *Control Objectives for Information and related Technology* (COBIT) [ISACA, 2007].

Currently, ITIL is the *de facto* standard for ITSM which allows integrating the business with IT by applying a process-oriented method. ITIL offers a detailed description of the most important processes to be carried out by IT service providers, including procedures, responsibilities, and task verification lists. Organizations can (wholly or partially) adopt ITIL, taking from it whatever they find of interest, and adapt it to their specific circumstances and needs. Since ITIL (especially the version 3) is strongly aligned with ISO/IEC 20000 [ISO/IEC, 2005a] [ISO/IEC, 2005b], ITIL can help companies guarantee the recognition of their capacities and can even become a key tool for ISO/IEC 20000 certification. Although the ISO/IEC 20000 standard does not depend on any specific business framework, it is based on the concepts and best practices defined in ITIL, offering a way to explore the guide of best practices.

Meanwhile, COBIT is an IT governance framework and a set of supporting tools that allows IT service providers to bridge the gap between control requirements, technical issues, and business risks [ISACA website]. That is, COBIT provides best practices for the management of IT processes harmonizing practices and standards such as ITIL and the *Project Management Body of Knowledge* (PMBOK) [PMI website]. In this vein, COBIT provides guidance for executive management to govern IT within the enterprise [ISACA, 2009]. A detailed mapping of ITIL V3 with COBIT 4.1 is documented in [ISACA, 2008].

As mentioned earlier, the implementation of an ITSMS is a complex task for any organization. To address it, companies can start using ITIL to manage the IT services included in their *Service Catalog* to support their business processes. The *Service Catalog*, which is part of the *Service Portfolio*, comprises the main IT services provided by a company, individually described. The documents gathering these descriptions must follow a reference framework for enabling a fluent dialogue between the IT service provider and its potential and actual customers. However, these documents are commonly written using natural language or informal notations, frequently leading to ambiguities, contradictions, and misinterpretations. In addition, most IT service providers do not know what should be actually measured for each ITSM process in order to demonstrate its value or to operate in a cycle of continuous improvement [Steinberg, 2006]. Thus, the definition of metrics that can be used to measure and monitor the health and state of each ITSM process in order to demonstrate

the impacts and effects of ITSM practices is of major importance. Following the guiding principles of *"If you do not measure it, you cannot manage it"* [DeMarco, 2009] and *"If you do not measure it, you cannot improve it"*, without metrics, organizations cannot monitor the IT services they are trying to manage, and this should be unacceptable in any business organization [Steinberg, 2006]: any business unit, even IT, cannot operate without learning how to effectively govern itself.

ITSMS is also closely related to information systems, as business information (and its automation) is essential for good service management. Information automation changes the way companies work, affecting its organizational model and business processes, more and more based on collaboration than on competition. According to their possibilities (available resources), organizations will sort the tasks involved in their IT processes and will automate the most crucial ones (i.e., those having a greater impact on their service improvement). There are several computer tools that allow automating the tasks or specific processes of a company (e.g., accounting, inventory management, payrolls, product design, financial simulation, etc.). In general, all these applications tend to work independently, although in some organizations it is necessary to integrate some of them [de Pablos et al., 2008]. Organizations can choose either to buy 'standard' applications available in the marketplace that meet their generic needs, or develop (or customize) applications that respond to the company's specific needs. In this context, there are commercial *Enterprise Application Integration* (EAI) platforms (e.g., TIBCO[1] or Websphere[2]), which enable the integration of business information and the generation of information systems. In the concrete area of ITSM based on ITIL, there exist several commercial tools that can help with the implementation of an ITSMS. The following list gathers some of them:

- EasyVista[3] (Staff&Line)
- FrontRange ITSM Software[4] (FrontRange Solutions)
- OTRS ITSM[5]

[1] http://www.tibco.es/
[2] http://www-01.ibm.com/software/websphere/
[3] http://www.staffandline.es/Front/inicio_3.php
[4] http://www.frontrange.com/software/service-management/itsm/
[5] http://www.otrs.com/en/products/itsm/

- Remedy[6] (BMC)
- Service Manager software[7] (HP)
- Service Desk Manager[8] (CA)
- Tivoli[9] (IBM)

For an effective implementation of ITSM processes, organizations need to use computer tools. However, these tools should not be acquired or implemented hastily. Nowadays, the problem is not the lack of tools, but the lack of an adequate and clear tool selection, adoption and integration policy. The main reason for these is the lack of precise knowledge about the actual ITSM processes carried out by each company. Therefore, the first thing an organization must do is to understand and analyze the maturity of the processes that manage their different services and the relations among them. This is as important or even more, that the choice of the computer tools. Furthermore, organizations need to know which assets are providing value to them, and how.

In Spain, more than 70% of IT service providers do not know the level of maturity of their ITSM processes [OZONA website]. A formal definition of these ITSM processes and their analysis can help organizations [OZONA website]: (i) know how their IT services are; (ii) define a list of deficiencies; and (iii) obtain part of the requirements that help them select or develop the most appropriate computer tools to support them. Finally, it is worth noting that the formal definition (conceptual modeling) of ITSM processes can serve as a base for their later implementation or adoption. This formalization can help in the analysis of ITSM processes, allowing IT service providers to establish the priorities when it comes to implement an ITSMS.

1.2 Thesis Objectives

This thesis proposes an *Ontology Engineering* (OE) and *Model-Driven Engineering* (MDE) approach for representing ITSMSs that formalizes the ITSM domain according

[6] http://www.bmc.com/products/product-listing/53035210-143801-2527.html
[7] https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-85^12473_4000_100__
[8] http://www.ca.com/us/service-desk.aspx
[9] http://www-01.ibm.com/software/tivoli/

to the ITIL V3 service management model. The proposed approach aims to allow IT service providers to implement ITSM processes related to an ITSMS, and help them to understand and manage the associated knowledge to improve the quality of their IT services. Our ontology-based and model-driven approach includes a basis for business decision making defining a set of ITSM *Key Performance Indicators* (KPIs) associated with ITIL processes. Also, through the tasks associated to the ITSM processes, companies can control both manual and automated activities with computer tools.

Thus, the main objective of this thesis is to give a formal definition of ITSM best practices that allow IT service providers to obtain and manage the knowledge associated to their ITSM processes through an ITIL-based semantic model. This will facilitate the management and automatic generation of the software specifications associated to the information systems that support them.

This main objective is composed of the following specific objectives:

O1: To propose and justify representation systems that allow us:

- to formalize the knowledge associated to ITSM best practices based on ITIL,
- to formalize the definition of an ITSM metrics model,
- to formalize the definition of the tasks associated to each ITSM process. The definition of the process tasks makes up the computer semantics of the information system that can give it support, and
- to indicate the computer tools (applications) that have already been implemented or integrated to give support to each ITSM process.

This will lead to the achievement of the following sub-objectives:

  - **O1.1**: Offer higher quality services to customers and users with the agreed costs (both in the new and the modified services).
  - **O1.2**: Obtain a formal ITSMS model that enables organizations to understand and analyze the maturity of each of their ITSM processes. Modeling ITSM processes can offer IT service providers with a perspective on the organization that expands the current business views, towards more collaboration between the interested parties [Ould, 1995]. For example, as

part of our ITSMS semantic model, the ITSM metrics model measures the quality and effectiveness of each ITSM process, providing a basis for its analysis and for business decision making in a well-defined manner. It is worth noting that the ITIL/ITSM terminology used throughout these work is aligned with the one adopted by the *Spanish Association for Standardization and Certification*, AENOR[10].

- **O1.3**: Obtain new knowledge about the ITSM processes offered by the company (added value).
- **O1.4**: Assure that the IT services cover customer and user's needs, reaching the objectives of satisfaction.
- **O1.5**: Improve the communication between the staff that takes part of the IT services and the customers and users of these services.
- **O1.6**: Increase the effectiveness and the efficiency of the internal processes associated to each IT service.

O2: To define a metamodel of the DSL that allows us obtaining models of ITSM process tasks (i.e., workflows).

O3: To process the analysis models obtained from the ITSM process tasks in order to automatically generate the high-level software requirements of the information systems supporting the management of IT services.

This will lead to the achievement of the following sub-objectives:

- **O3.1**: To obtain conceptual models of the information systems that support the ITSM processes associated with an ITSMS.
- **O3.2**: From the former, to select or develop the most appropriate computer tools in order to help organizations improve their competitiveness.

[10] http://www.aenor.es/

1.3 Research Method

This research carried out as part of this thesis has been developed through the following phases:

1. Approach to the Problem

As stated previously, the current difficulties to deal with ITSM give way in this research to posing a modeling approach for ITSMSs. The proposed approach aims to allow IT service provides to adopt ITSM best practices in a formal manner, independently, and with a common knowledge. It also aims to help them defining in a formal, automated and flexible way, the requirements of the information systems needed to automate certain tasks associated to their ITSM processes. For this purpose, the following steps were taken:

- To identify the existing alternatives for knowledge representation.
- To identify the existing alternatives for representing business information (namely, business processes).
- To review the related works in the field of knowledge representation and process modeling for ITSM best practices.
- To integrate the ITSM metrics model presented in [Steinberg, 2006] with the proposed modeling approach for ITSMSs based on ITIL.
- To review the related works in the field of MDE in order to generate high-level requirements models of the information systems that support the ITSM processes associated with an ITSMS.
- To identify the existing alternatives for model transformation.

2. Research Background

A study on the technical state and related tasks is carried out. It is necessary to be able to understand and define the proposal of this thesis.

3. **Analysis and Design of the Approach**

   A formal approach for ITSMSs is proposed following ITIL best practices complementing it with an ITSM metrics model.

4. **Implementation of a Prototype**

   A prototype is implemented that will allow to validate the approach designed in the previous point.

5. **Evaluation of the Objectives**

   Making use of the previous prototype, necessary tests are carried out. These tests will allow checking if and up to what point they abide to the objectives defined in this thesis.

6. **Conclusions**

   Finally, the conclusions reached are detailed after the evaluation of the objectives covered through the prototype, and which future tasks that can be carried out are outlined.

1.4 Thesis Outline

The remaining of this document is structured as follows:

- **Chapter 2** overviews the current state-of-the-art, providing basic concepts, theory and technologies related to ontologies, business process modeling, software modeling, and ITSM. In addition, it also reviews a number of existing ontology-based approaches related to the different domains of interest integrated into the proposed modeling approach for ITSMSs.
- **Chapter 3** describes the foundations of Onto-ITIL: the proposed ontology-based and model-driven approach for ITSMSs based on the ITIL V3 Service Management Model.
- **Chapter 4** details the implementation of the prototype that supports our modeling approach. It also describes the case study used to validate both the proposal and the prototype implemented to support it.

- Finally, **Chapter 5** draws the conclusions of the thesis. The chapter also outlines some areas for future research.
- In addition to the chapter structure described above, this document also includes **two appendices**: one describing the concepts included in the proposed ITSM Ontology, and the other including a Glossary of Terms.

Chapter 2

State of the art

> "Innovation is created by an unprecedented disposition of old things."
>
> Jacques Monod (1910-1976), *French biochemist*

In this chapter we describe the background information and fundamental issues about ontologies, model-driven software development, business processes, and ITSM, all related to our research. The complexity associated to performing quality services is of major importance. The adoption of a process-based ITSM approach has appeared to be a major challenge for many IT organizations that use it to organize themselves around technology. Integrating ontologies with a model-driven software development approach opens a window for the establishment of a systematic method in order to implement ITSMSs in a straightforward and well-defined manner.

2.1 Ontologies

The term 'ontology' arose from the branch of philosophy known as metaphysics, which deals with the nature of what exists (i.e., the real-world). The traditional goal of ontological inquiry is to divide the real-world into concepts (terms) in order to discover those fundamental categories or kinds that define the objects of the real-world. There are vast human-designed and human-engineered systems (e.g., manufacturing plants, businesses, military bases, universities, etc.), in which ontological inquiry plays a key role. In these human-created systems, ontological inquiry is primarily motivated by the need to understand, design, engineer, and manage such systems effectively. Consequently, it is useful to adapt the traditional techniques of ontological inquiry in the natural sciences to these domains as well [KBSI, 1994]. In this context, ontologies are explicit representations of a shared conceptualization [Gruber,

1995] [Uschold & Grüninger, 1996]. The term 'shared' indicates that an ontology captures some consensual knowledge, and the term 'conceptualization' means an abstract, simplified view of a shared domain of discourse (i.e., the real-world) [Gašević et al., 2006]. There may be several conceptualizations of the same domain and therefore several ontologies [Olivé, 2007].

More formally, an ontology defines the vocabulary of a problem domain and a set of constraints (axioms or rules) on how terms can be combined to model specific domains. A ontology is typically structured as a set of concept definitions and relations between them. Ontologies are machine-processable models that provide the semantic context, enabling natural language processing, reasoning capabilities, domain enrichment, domain validation, etc.

Ontology Engineering (OE) is sometimes seen as the next silver bullet in knowledge modeling, aiming at avoiding conceptual ambiguities, advocating reuse and standardization, and serving as building blocks for more complex automated-reasoning systems [Chandrasekaran et al., 1999] [Gruber, 1991]. OE has shown to be useful for [KBSI, 1994]: (i) consensus building; (ii) object-oriented design and programming; (iii) component-based programming; (iv) user interface design; (v) enterprise information modeling; (vi) business process reengineering; and (vii) conceptual schema design. In addition, OE provides several benefits to organizations [KBSI, 1994]:

7. *Enhanced understanding of a domain*. The insights of ontological analysis are useful for: (i) problem identification (diagnosis); (ii) identification of problem causes (causal analysis); (iii) identification of alternative solutions (discovery and design); (iv) consensus and team building; and (v) knowledge sharing and reuse.

8. *Business-IT alignment*. The ontologies that result at the end of an ontology development effort can be used for: (i) information systems development, as ontologies provide a blueprint for developing more intelligent and integrated information systems; (ii) system development, as ontologies can be used as reference models for planning, coordinating, and monitoring complex product/process development activities; (iii) business process reengineering, as

ontologies provide clues to identify focus areas for organizational restructuring and they suggest potential high-impact transition paths for restructuring.

According to the level of generality, the following types of ontology are suggested in [Guarino, 1998]:

- *High-level (upper) ontologies*. This kind of ontologies describes very general concepts like space, time, matter, object, event, action, etc., which are independent of a particular problem or domain.
- *Domain ontologies*. This kind of ontologies describes the vocabulary related to a generic domain (e.g., ITSM or business processes), by specializing the terms introduced in the top-level ontologies.
- *Task ontologies*. This kind of ontologies describes a generic task or activity (e.g., monitoring or measuring), by specializing the terms introduced in the top-level ontologies.
- *Application ontologies*. This kind of ontologies describes concepts depending on both a particular domain and task, which are often specializations of both related ontologies. These concepts often correspond to roles played by domain entities while performing a certain activity.

Since the inception of the Semantic Web, in which ontologies are the principal resource to integrate and deal with online information, a new set of standards have been proposed. The *Web Ontology Language* (OWL) is one of such standards that belong to a family of knowledge representation languages prepared for the Semantic Web (although this language can be adopted in other domains, as we propose in this thesis). OWL has reached the status of *World Wide Web Consortium* (W3C) recommendation. From a technical point of view, OWL extends the *Resource Description Framework* (RDF) and *RDF Schema* (RDF-S), allowing us to integrate a variety of applications using the *Extensible Markup Language* (XML) as interchange syntax. Therefore, due to its RDF basis, OWL ontologies can be associated to any other form of information expressed on the Semantic Web, and it allows the integration of the resulting specifications with a variety of e-business frameworks (e.g., the *electronic business using*

XML (ebXML[11]) [OASIS, 2001]) and business modeling languages (e.g., *Business Process Model and Notation* (BPMN) [OMG, 2010a], both using the XML as interchange syntax in order to match organizations with the same business processes. An e-business framework is a standard for e-business that uses a data format to define data structures and data elements in a business context [Nurmilaakso, 2008]. The main objective of e-business frameworks is to standardize the exchange of electronic business data.

The OWL *Description Logics* (OWL DL) [Baader et al., 2003] is a sublanguage of OWL. OWL DL is a family of logics for concept definitions and it is used to describe domain knowledge. OWL DL enables concept specification by rich and precise logical definitions [Baader et al., 2003]. One of the key capabilities of OWL DL is its ability to define all these classes in terms of necessary and sufficient conditions. New concepts can be defined by specifying property restrictions on existing concepts. Then, an inference engine can execute the ontology and compute the new inferred ontology class hierarchy, remarking inconsistent classes (e.g., a reasoner can test whether one class is subclass of another class or not). It is important to notice that reasoning in OWL DL is based on the *Open World Assumption* (OWA). This means that *"(negative) conclusions drawn from a knowledge base must be based on information explicitly present in the knowledge base"* [Knorr et al., 2011]. That is, it cannot be assumed that a piece of knowledge does not exist until it is explicitly stated in the knowledge base. In the *Closed World Assumption* (CWA), all non-provable expressions are assumed to be false. Because of the absence of a piece of knowledge should not be taken as an indication that the piece of knowledge is false, the decision to rely on the OWA appears to be natural in the World Wide Web domain. However, when an ontology-based reasoning is done in conjunction with data stored in a database, the CWA seems to be the better assumption. In a database, the data are usually considered to be complete in such a way that statements that are not present in the database should be taken as false. In some domains, the combination of open and closed world assumption is required. For example, in a clinical domain, OWA is needed in radiology and laboratory data (e.g., unless a tab test asserts a negative finding, we cannot make arbitrary assumptions about the results of the test). In this case, we can only be certain that some patient does not

[11] http://www.ebxml.org/

have a specific kind of cancer if the corresponding test has a negative result [Knorr et al., 2011]. However, CWA should be used with data about medical treatment to infer that a patient is not on a medication unless otherwise stated. Similar situations occur in other domains that have been explored in [Grimm & Hitzler, 2008].

OWL ontologies are composed of: (i) classes, as sets of individuals, (ii) individuals, as instances of classes (i.e., objects of the domain), and (iii) properties as binary relations between individuals. It is possible to specify property domains, cardinality ranges, and reasoning on ontologies. Also, some reasoners (e.g., Pellet[12]) can be used to infer additional facts about the knowledge that has been explicitly stated in OWL ontologies. Reasoning in OWL can be performed at a class, property, or instance level, and reasoning examples include class membership, equivalence of classes, consistency, classification of the information, obtaining additional properties using transitiveness or equivalence, etc.

A related specification, the *Semantic Web Rule Language* (SWRL) [Horrocks et al., 2004], is based on RuleML[13]. The SWRL extends the OWL, providing logic-based rules and, in consequence, providing more expressiveness. Rules together with stored facts (knowledge base) are executed as inputs by the rule engine, which infers new facts as an output. In addition, if the rule engine infers new knowledge using forward chaining, this knowledge can be used for further inference. A combination of rules and ontologies would clearly yield a combination of the OWA and the CWA. However, rules are usually limited in their applicability to the different objects explicitly appearing in the knowledge base [Knorr et al., 2011].

Finally, the *Semantic Query-Enhanced Web Rule Language* (SQWRL) is a query language that enables to extract information from OWL ontologies [O'Connor & Das, 2009]. SQWRL is based on the SWRL and it uses the SWRL's semantic foundations as its formal support. SQWRL includes a set of operators that allow the definition of negation as failure, disjunction, counting, and aggregation functionality in the construction of retrieval specification.

[12] http://clarkparsia.com/pellet/
[13] http://ruleml.org/

Ontology Development Environments

Graphical ontology editors allow us to build formal ontologies. Graphical ontology development environments integrate an ontology editor with other tools and usually support multiple ontology representation languages. They are aimed at providing support for the whole ontology development process and for the subsequent use of the ontology [Corcho et al., 2002].

The open source Protégé[14] tool is an example of a widespread ontology development environment. The Protégé-OWL editor is an extension of Protégé that provides support to OWL. The Protégé-OWL editor enables users to load and save OWL and RDF(S) ontologies, edit and visualize classes, properties, taxonomies and several restrictions, as well as class instances (i.e., the actual data in the knowledge base). It also includes the *SWRLTab* which is an extension for editing and executing SWRL rules in conjunction with the Jess[15] rule engine.

2.2 The Business Process

The *Compact Oxford English Dictionary* defines *process* as *"a series of actions or steps towards achieving a particular end."* In a business context, a process is the way for an organization *"to organize work and resources (people, equipment, information, and so forth) to accomplish aims"* [Sharp & McDermott, 2001] or the way for an organization *"to achieve its business objectives"* [Ould, 1995]. Different definitions about what is a business process are provided by several authors, although they are variations of the same issues. A *business process* is defined in [Hammer & Champy, 1993] as *"a set of activities that, together, produce a result of value to the customer."* Another definition is found in [Sharp & McDermott, 2001], where a business process is defined as *"a collection of interrelated work tasks, initiated in response to an event, achieving a specific result for the customer and other stakeholders of the process."* The *event* represents a specific request for the output generated by the process. The *customer* of the *process* is the recipient or beneficiary of the output produced by the *business*

[14] http://protege.stanford.edu/
[15] http://www.jessrules.com/

process. The customer does not just refer to a customer purchasing goods or services but it may be a person, and organization, or even a broad marketplace. The customer of the process may be internal to the organization (e.g., the department that receives a newly hired employee). The flow of information and control of a business process may be arranged with a *workflow*. *Develop Product*, *Hire Employee* and *Resolve Incident* are examples of business processes. In [Marshall, 2000], a business process is *"a set of tasks arranged to form workflow structures which define how an organization achieves its purpose."*

Jacobson et al. [1995] provide another definition: *"[...] a business process is the set of internal activities performed to serve a customer. The purpose of each business process is to offer each customer the right product or service (i.e., the right deliverable), with a high degree of performance measured against cost, longevity, service and quality."* Again, the term customer should be taken in a broad sense.

The *Workflow Management Coalition* (WfMC) established in August 1993 is a non-profit international body for the development and promotion of workflow standards (including a workflow reference model). The WfMC defines a business process as *"a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships. [...] a business process may consist of automated activities, capable of workflow management, and/or manual activities, which lie outside the scope of workflow management"* [WfMC, 1999].

In short, we can define a business process as a way to organize work and resources which enables to achieve the aims through a set of activities that will be performed in certain order (i.e., workflow representation). The workflow of an actual process defines *What* (process's purpose), *How* (activities), *Who* (resources) and *When* (activities' order).

On the other hand, Ould [1995] categorizes processes into three groups:

- *Core processes*. Processes that are concerned with addressing external requests from an organization. Core processes directly add value in a way perceived by the customer of the business.

- *Support processes*. Processes that concentrate on satisfying internal customers. Support processes might add value to the customers indirectly by supporting a core business process, or they might add value directly to the business by providing a suitable working environment.
- *Management processes*. Processes that manage both core and support processes, or manage planning at the business level.

The idea of core processes is to enhance customer satisfaction; the idea of support processes is to enhance the organization efficiency; and the idea of management processes is to enhance the organization structure [Mili et al., 2010].

2.2.1 Workflow

A workflow can be defined as *"the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules"* [WfMC, 1999]. Workflow models can help defining the information systems needed to support the business [Eriksson & Penker, 2000].

Eshuis and Wieringa [2001] define a workflow model as a specification of an ordering on activities performed in an organization. Sharp and McDermott [2001] state that a workflow represents the flow of information and control in a business process. The workflow model depicts the three Rs, that is: (i) *Roles*: actors or process performers who participate in the process (that is, the resources); (ii) *Responsibilities*: individual tasks that each resource is responsible for; and (iii) *Routes*: flows of control that connect the tasks together and, therefore, define the path that each individual work item will take through the process.

As every model, a workflow model is a simplified representation of the actual workflow. The following subsection briefly reviews some of the current approaches to business modeling.

2.2.2 Business Process Modeling

Different modeling techniques have been used for years to assess and improve business processes [Recker et al., 2009] [Mili et al., 2010]. Business process models provide a simplified view or description of the business structure and capture the business core functions [Eriksson & Penker, 2000]. A business process model defines how work is to be done. Furthermore, business models provide a suitable communication means for all the stakeholders involved in the business process, helping them to detect and incorporate innovations and improvements. Since information systems are aimed at supporting the business, their development may be greatly improved if an appropriate business modeling support is provided.

Business models do not necessarily include any detail about software systems. However, when a software system is designed to automate (part of) the business process, its requirements can be derived from (part of) the business model. That is, the requirements of the information systems needed to support the business can be derived from (part of) the business model [Kleppe et al., 2003]. Just as Kent [2002] points out, if we have not defined the process within which the artifacts of a particular project are intended to be used, it will be difficult to identify them. Therefore, we believe that success in software system development is difficult to obtain without appropriately supporting the business process (or processes) it will be integrated in.

According to Ould [1995], *Business Process Modeling* (BPM) (also known as *Business Process Management*) is also useful for three basic purposes:

1. *To describe a business process*. A descriptive business process model acts as a work instruction to people in the organization. A descriptive business process model enables us:
    - to define a business process (*"this is how we shall together handle customer complaints"*),
    - to communicate it to others (*"this is how your work contributes to the IT department's goals"*),

- to share it across a group of people (*"so, this is how we do things round here"*), and
- to negotiate around it (*"if you could do this, my life would be made much easier, in return I can..."*).

2. *To analyze a business process*. The properties of a business process model can be explored for further analysis. Such analysis is a common precursor to improving the organization by:
   - changing the ordering of activities,
   - changing responsibilities for activities or decisions,
   - changing scheduling mechanisms,
   - increasing or decreasing the amount of parallel activity,
   - removing or adding buffers or stores between steps in a business process,
   - restructuring functions to align them better with the business process,
   - etc.
3. *To enact a process*. Given a data model, it is possible to store it in a database and automate it to generate forms and reports that the organization can use to add, modify or present data. That is, the data model can be 'executed'. In the same way, a computer system can receive a business process model and enact it that is, run the model, supporting the agents that participate in the business process, handling their agendas, supporting their interactions and, perhaps, playing its own part in the process. These *enactment systems* require a business process model which 'meaning' is sufficiently well defined as to allow them to enact the process without further human intervention to define it.

The next motivations for choosing BPM are given in [Havey, 2005]:

- *Formalize existing process and identify needed improvements*. Adopting BPM forces a business to think about and formalize its understanding of current business processes. This may help organizations identify certain improvements,

such as the removal of certain steps, the automation of others, or the reengineering of (part of) a business process.

- *Facilitate automated, efficient process flow*. Given that a process spans multiple activities, it is better when the time spent between them is significantly reduced. When BPM software drives the process flow, downtime between activities is almost zero, unless the software itself is down. BPM supports process parallelism, so that independent sequences of work can be performed concurrently in isolation of each other, with their results merged and synchronized later in the flow. A process controlled by phone calls or e-mails, for example, is bound to be significantly slower and prone to getting lost or stuck.
- *Increase productivity and decrease head count*. Recent BPM case studies point out that it is possible to get work done faster with fewer people. For example, a financial service department was able to reduce staff while decreasing processing time and increasing customer satisfaction [Plesums, 2002].
- *Allow people to solve hard problems*. Although BPM is often about removing or decreasing human participation in a business process, one of its benefits is its flexibility to use people to help fix problems.
- *Simplify regulations and compliance issues*. BPM helps business build auditable business processes that help organizations comply with several regulatory requirements. For example, in the IT sector, the implementation of the ISO/IEC 20000 standard [ISO/IEC, 2005a] [ISO/IEC, 2005b] has forced IT service providers to build new processes (or to improve existing ones) in order to manage the services they deliver to their customers.

There are several modeling languages that can be used to describe business processes. An up-to-date review of business process modeling languages can be found in [Mili et al., 2010]. According to the authors, business process modeling languages are classified in the next groups:

- *Traditional process modeling languages.* Languages that mostly come from the *Management Information System* (MIS) tradition of *Information Engineering*

(IE) and from work on *Business Process Engineering* (BPE). The most known languages that fall in this category are Petri nets [Silva, 1985] [Jensen, 1996], IDEF[16], *Event-driven Process Chain* (EPC) [Scheer, 2000] and *Resource Event Agent* (REA) [McCarthy, 1982].

- *Object-oriented languages.* Languages that introduce a single abstraction (i.e., the object) which encapsulates both the static and dynamic views that characterize the analysis and design of information systems. The connection between the structure and the behavior of a system is also more natural in an object-oriented solution, where the notion of change of state is central [Pastor & Molina, 2007]. The most known language that falls in this category is the *Unified Modeling Language* (UML) [OMG, 2010c].

- *Dynamic process modeling languages.* Languages that are focused on the dynamic view of business processes. In terms of usage, they cover the full spectrum, from describing business processes for human consumption to enacting/executing business processes. All the dynamic process modeling languages emphasize a serialization format for model interchange, typically XML [Mili et al., 2010]. The most known languages that fall in this category are *Business Process Model and Notation* (BPMN) [OMG, 2010a] and the *Web Services Business Process Execution Language* (WS-BPEL) [Jordan & Evdemon, 2007] [OMG, 2010a].

- *Process integration languages.* Languages that are focused on the interactions between partners within the context of multientity business processes, as for example, within the context of *business-to-business* (B2B) commerce. For electronic interorganizational commerce to take place, the business partners need to have a shared understanding of the business messages and documents that need to be exchanged, the sequence of exchanges, and the expected results from each of the partners [Mili et al., 2010]. These languages typically focus on integration mechanisms in terms of abstract, technology-independent, programming interfaces and data exchange formats. Languages in this category may also capture different semantics levels of the underlying processes [Mili et

[16] http://www.idef.com/

al., 2010]. The best-known languages that fall in this category are the *electronic business using XML* (ebXML) [OASIS, 2001] and RosettaNet[17]. The ebXML framework is an example of an e-business framework that has been standardized by means of the XML format. The vision of ebXML is to reuse predefined business process in such a way that organizations of any size and in any geographical location can meet and do electronic business with each other through the exchange of XML-based messages, where e-mail is used as the primary communication tool for collaboration. To do this effectively, ebXML provides an infrastructure for data communication interoperability, a semantic framework for commercial interoperability, and a mechanism that allows enterprises to find, establish a relationship, and conduct business with each other [OASIS, 2001]. In addition, ebXML provides a shared repository where businesses can discover each other's business offering by means of partner profile information, a process for establishing an agreement to do business, and a shared repository for company profiles, business process specifications, and relevant business messages [OASIS, 2001].

BPMN

Nowadays, OMG's *Business Process Model and Notation* (BPMN) is considered the *de facto* standard notation for business processes modeling, and it is possible to find several workflow management systems described using this notation. BPMN represents many years of effort by the *Business Process Management Initiative* (BPMI[18]) Working Group. The OMG has brought forth expertise and experience with many existing notations (e.g., EPCs, UML activity diagrams, UML EDOC business processes, IDEF, RosettaNet, and ebXML, among others) and has sought to consolidate the best ideas from these divergent notations into a single standard notation in terms of BPMN. Therefore, BPMN represents the amalgamation of best practices within the business modeling community to define the notation and semantics of collaboration, process, and choreography diagrams. The intent of BPMN is to standardize a business process model

[17] http://www.rosettanet.org/
[18] http://www.bpmi.org/

and notation in the face of many different modeling notations and viewpoints. Doing so, BPMN provides a simple means of communicating process information to other business users, process developers, customers, and suppliers [OMG, 2010a].

BPMN provides businesses with the capability of understanding their internal procedures in a graphical notation, and give organizations the ability to communicate these procedures in a standard manner. Furthermore, the graphical notation facilitates the understanding of the performance collaborations and business transactions between the organizations. This ensures that businesses will understand themselves and the participants in their business [OMG/BPMI-BPMN website]. The primary goal of BPMN is to provide a notation readily understandable by all business users, including [OMG, 2010a]: (i) *business analysts*, who create the initial draft of the processes; (ii) *technical developers*, responsible for implementing the information systems aimed at supporting those processes; and (iii) *business people*, who manage and monitor those processes. Thus, BPMN helps bridging the gap between the business process design and their implementation. Another goal of BPMN is to ensure that XML languages, designed for the execution of business processes (e.g., WS-BPEL) can be visualized with a business-oriented notation.

BPMN is provided with an internal model, which enables the generation of WS-BPEL. BPMN defines a *Business Process Diagram* (BPD), which is based on a flowcharting technique, tailored for creating graphical models of business process operations. A *Business Process Model*, then, is a network of graphical objects, which are activities (i.e., tasks) and the control flows that define their execution order.

The lack of a standard metamodel associated to the BPMN graphical notation, has hindered the shared integration of formal BPMN models into different model-driven approaches. For example, formal model-to-model (M2M) transformations require that both the source and the target models are defined in terms of formal metamodels, since they are defined as mappings between the concepts included in both metamodels. To overcome this limitation, the OMG has adopted a specification of BPMN that includes a formal metamodel, which is currently in the finalization phase (at the time writing this thesis, BPMN 2.0 is in the beta phase). Also, the OMG has recently released the first version of the *Business Process Definition Metamodel* (BPDM) [OMG, 2008]. BPDM

defines the abstract syntax (metamodel) of a modeling language, which associated concrete syntax is BPMN. However, the BPDM specification has not been yet extensively proved in real-world projects and, as a consequence, it is liable to change in the near future or even to fall into oblivion in favor of the BPMN metamodel. The last survey by BPTrends regarding the 'State of Business Process Management 2010' [Wolf & Harmon, 2010], reflects that, in addition to general standards such as ISO 9000 [ISO, 2005a] and the *Capability Maturity Model Integration* (CMMI) [2009], organizations are more interested in the adoption of BPMN (51%) and UML (24%) as notations for business process management/modeling. On the other hand, BPDM has very little or no interest to organizations and, as a consequence, BPDM support has been reduced since 2005 (from 10% to 7%). It is worth noting that WS-BPEL (the standard approach for moving from a process description to code) has hardly gained any additional support since 2005, while BPMN, which is sometimes considered a way of preparing to use WS-BPEL, has become very popular. The lack interest in WS-BPEL could be a result of the fact that this standard is incomplete and it cannot handle workflow problems appropriately, while nearly all process modeling vendors have adopted BPMN, which works well for analyzing and designing either business process models or software process automation designs [Wolf & Harmon, 2010]. The BPTrends BPM 2010 Market Survey report [Wolf & Harmon, 2010] summarizes information provided by 264 respondents who participated in BPTrends survey in the fall of 2009. The report analyzes the responses and compares them with the responses from the two previous BPTrends surveys conducted in 2007 and 2005, respectively. In all cases, the respondents represent a broad cross section of industries from around the world.

2.2.3 Ontologies for Business Process Modeling

The research efforts related to the definition of business processes in terms of ontologies have been summarized in Table 2.1. For example, the REA enterprise ontology [Geerts & McCarthy, 1999] [Geerts & McCarthy, 2000] is an evolution of the REA framework [McCarthy, 1982] in a shared data environment. In other words, the ontology is about understanding organizations by identifying operational-level resource categories, events and agents that form a basic ontology. The aim of the REA ontology

is to provide an expressive language that allows users to capture and share enterprise knowledge [Sedbrook & Newmark, 2008]. The REA ontology defines the value chain as a set of business processes through which resources flow. This assumes that customer value is added to the resources within each business process. The value chain is intended to show total value and consists of value activities and margin. Value activities are the physical and technological activities performed by an organization [Dunn et al., 2005]. The value chain level of the REA ontology is constructed based on two concepts: *duality* and *stockflow*. *Duality* is an association between two or more events that coordinate an exchange of resources. Economic events represent either an increment or decrement in the value of economic resources. *Stockflow* is defined as the inflow or outflow of a resource. *Stockflow* relationships exist between events and resources. The REA ontology can provide a basis for service systems ontology, as it represents value transfer appropriately and has the potential of accommodating the characteristics of services. In this vein, Sicilia and Mora [2010] propose a REA enterprise ontology for service systems by means of extensions or refinements of the REA ontology[Sicilia & Mora, 2010].

Table 2.1 Related work about Ontology-based business process modeling

Author and Year	Feature
Abramowicz et al. [2007]	sBPMN ontology
Berztiss [1999]	Ontological approach to business modeling
Belhajjame & Brambilla [2009]	Discovery of business processes by means of abstract business processes
Born et al.[2007]	Semantic annotation in business process modeling based on sBPMN ontology for supporting modeling tools integration
Di Francescomarino et al. [2011]	Semantic annotation in business process modeling based on a proposed BPMN ontology
Geerts and McCarthy [1999] Geerts and McCarthy [2000]	REA enterprise ontology
Green et al. [2005]	Ontological evaluation of ebXML BPSS
Joukhadar and Al-Maghout [2008]	Ontological approach to business modeling
Prieto and Lozano-Tello [2009]	Ontological approach to workflow modeling
Sicilia and Mora [2010]	REA enterprise ontology for service systems
Shangguan et al. [2007]	Ontology-based business process modeling using eTOM and ITIL
Thomas and Fellmann [2009]	Semantic annotation in business process modeling
Walter and Ebert [2009]	Integrated metamodel of BEDSL+OWL

Walter and Ebert [2009] propose the *Business Entities Domain-Specific Language* (BEDSL). BEDSL is a *Domain-Specific Language* (DSL) aimed at business entity

modeling. BEDSL is platform independent, focusing on representing business objects, like entities, their attributes and relationships. Since ontologies support the definition of constraints, rules and semantics, BEDSL is enriched by the integration with the ontology language OWL [Smith et al., 2004].

For semantic annotations in business process modeling, Thomas and Fellmann [2009] and Di Francescomarino et al. [2011] propose extensions of process modeling languages, such as BPMN [OMG, 2010a], using concepts of a formal ontology. The semantic process modeling proposed in [Thomas & Fellmann, 2009] uses the *Suggested Upper Merged Ontology* (SUMO) [Niles & Pease, 2001] for the ontology construction and OWL DL as the ontology language. The semantic annotation consists of connecting the process models and the model elements with other elements in the same ontology. The proposed information model is language independent and it is possible to use it for different modeling languages, although specific mapping and extensions should be defined. In [Di Francescomarino et al., 2011], the authors propose a framework for the collaborative specification of semantically annotated business processes. The proposed framework is based on the notion of a shared workspace aimed at obtaining annotated BPDs specified using BPMN, where each BPD element is considered as an instantiation of an element specified in their BPMN ontology [DKM website].

In the same context, Born et al. [2007] propose an approach for integrating semantics in modeling tools in order to support the graphical modeling of business processes with information derived from domain ontologies. For this purpose, the authors propose the use of an extended BPMN ontology (*Semantic Business Process Modeling Notation* – sBPMN –) [Abramowicz et al., 2007] to augment and annotate business process models. The sBPMN ontology adds meaning to each of the process elements and make then machine-processable. The proposed ontology also allows reasoning on the process description. The sBPMN ontology was created within the SUPER project[19] .

Prieto and Lozano-Tello [2009] propose a workflow model based on ontologies to represent management processes defined in terms of workflows. The authors remark that the application of ontologies in this field can provide several advantages such as exchange of tasks and workflow model reuse.

[19] http://www.ip-super.org/

Another approach is presented in [Shangguan et al., 2007]. In this case, they use ontologies to combine the *Enhanced Telecom Operations Map* (eTOM) and ITIL processes to analyze composite business processes. ITIL is used to improve the soundness and robustness of the eTOM-based business processes. This approach is focused on the application of the eTOM process framework for business process modeling for Telecommunication service providers.

Belhajjame and Brambilla [2009] use ontologies to describe actual business processes in terms of abstract business processes. This approach is aimed at easing business process discovery in order to increase their reuse and, therefore, the overall design productivity. The authors model a business process by means of a workflow model specified using BPMN. They also use the terminology defined by WfMC [WfMC, 1999] and BPMI, and the concepts specified by BPDM [OMG, 2008].

The approaches presented in [Berztiss, 1999] and [Joukhadar & Al-Maghout, 2008] share the idea of adopting an ontological approach for the conceptualization of a business domain. Bertziss [1999] builds a discussion around a flexible generic domain model that can serve all enterprises performing similar functions. According to Berztiss, domain modeling can provide a general framework (i.e., the ontology) that then can be adapted to the specialized needs of individual enterprises. In [Joukhadar & Al-Maghout, 2008], the authors present a cost- and time-effective multilingual solution that improves agility in business application by enabling the domain expert to specify business rules directly in natural language. Different natural languages are supported thanks to the adopting of a novel approach to natural business rules understanding, based on the business models and enriched metadata provided by Elixir MDA Framework[20]. In order to understand a sentence written in natural language, it is necessary to count on a real-world model (i.e., the ontology) that represents the particular context in which the sentence is going to be evaluated.

In another line of research, Green et al. [2005] show the potential utility of the *Bunge-Wand-Weber* (BWW) representation model [Wand & Weber, 2003] to evaluate business process specifications for enterprise interoperability. The BWW representation model is a set of ontological constructs used to describe the real-world that allows users to

[20] http://www.el-ixir.com/en/index.php

represent a conceptual model of a specific information system domain. To validate their approach, they map the BWW representation model constructs to the ebXML BPSS constructs [UN/CEFACT and OASIS, 2001].

2.3 The Software Development Process

Sommerville defines the *Software Development Process* (SDP) or simply *software process* as *"the set of activities and associated results that produce a software product"* [Sommerville, 2010]. The goal of this set of activities is the development or evolution of software, that is, a software process produces software. According to Raistrick et al. [2004], the software development process is *"the means by which we characterize and structure the practice of software production."* On the other hand, Humphrey defines the software process as *"the set of tools, methods, and practices we use to produce a software product"* [Humphrey, 1989]. In this respect, the software process must consider the relationships of all required tasks, the generated artifacts, the technologies, tools and methods used, and the skill training, and motivation of the people involved in the project.

The application of the software process to the development domain (i.e., instantiation or enaction of the software process) is usually called the 'process instance' or a 'software development project' [Graham et al., 1997].

There are different types of software processes, but all of them produce or modify tangible 'things', such as documentation, design artifacts, source code, tests suites, etc. The execution of a software process produces two kinds of *artifacts*: (i) *Internal results*; and (ii) *Deliverables* (i.e., results delivered to the customers).

Software processes are defined in order to improve the way the work is done. If we think about the software development process in an orderly manner, it must be possible to anticipate problems and to devise ways either to prevent or to resolve them. According to Humphrey [1989], some of the major software process issues concern quality, product technology, requirements, instability, and complexity. Therefore, a software process must define the problem in such a way that it is easy to understand and

to design a solution. Thus, as shown in Figure 2.1, the software system is the result of transforming the user's requirements into code.

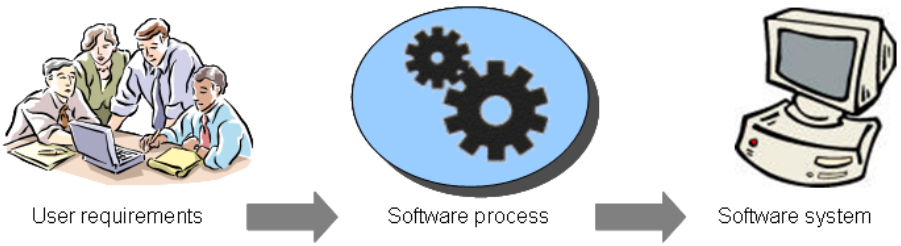


Figure 2.1 The software development process

The activities associated to a software process are mostly carried out by software engineers. According to Sommerville [2010], there are four fundamental process activities that are common to all software processes:

- *Software specification*. The software functionality and the constraints on its operation are defined.
- *Software development*. The software meeting the specification is produced.
- *Software validation*. The software is validated to ensure that it does what the customer wants.
- *Software evolution*. The software must evolve to meet changing customer needs.

These generic activities may be arranged in different ways and may be described at different levels of detail for different types of software systems. For each project, a different development process can be selected, depending on the type of the system to be developed. The use of an inappropriate software process may lead to reduce software quality and usefulness and to increase its development costs.

2.3.1 Software Process Modeling

A *software process model* is *"a simplified description of a software process that presents one view of that process"* [Sommerville, 2010]. That is, a model of a software process (also known as *process definition*) is a simplified representation of an actual SDP. Although a model is a simplification (abstraction) this is one of its main

advantages: a model of a SDP should be easy to understand and follow by all the developers involved in a given project.

Humphrey [1989] defines the software process model as *"one specific embodiment of software process architecture. [...] Software process architecture is a framework within which project-specific software processes are defined. [...] While software process models may be constructed at any appropriate level of abstraction, the process architecture must provide the elements, standards, and structural framework for refinement to any desired level of detail."*

Some examples of software process models that can be extended and adapted to create specific *Software Engineering* (SE) processes in order to enable dynamically certain adjustments to own particular needs and constraints can be found in [Sommerville, 2010] (e.g., the waterfall model, the evolutionary development, the incremental delivery, or the spiral development). These process models are widely used in current SE practice. They are not mutually exclusive and are often used together, especially for large systems development.

2.3.2 Model-Driven Engineering

The emerging *Model-Driven Engineering* (MDE) addresses the inability of third-generation languages to cope with increasing software complexity, allowing designers to describe domain concepts effectively [Schmidt, 2006]. MDE revolves around models (defined in terms of formal metamodels), and model transformations, which provide a powerful mechanism for incremental and automatic software development.

A model in MDE is a "*graph-based structure representing some aspects of a given system and conforming to the definition of another graph called a metamodel"* [Bézivin, 2005]. Therefore, the basic set of MDE principles is based on two concepts and two basic relations. The two concepts are *system* (the OMG's information layer) and *model* (the OMG's model layer) and the relations are *conformance* and *representation*: a *model* is said to *represent* the *system* and a *model* is said to *conform* to its *metamodel*. These principles can be seen in Figure 2.2.

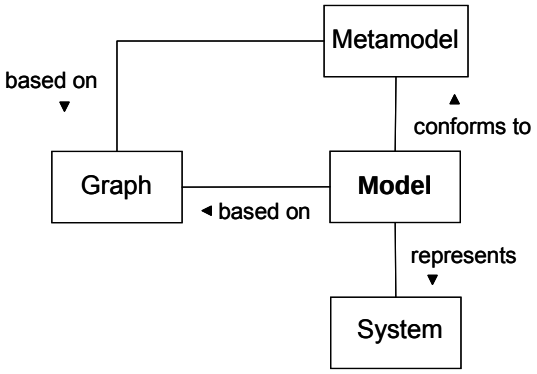


Figure 2.2 Basic MDE principles

2.3.2.1 Software Modeling

Over the last decade, models and software modeling are becoming one of the most important flagships of software development. Developers raise the level of abstraction thanks to the use of models for specifying software solutions (i.e., the final system implementation). Models are part of the software and they do not constitute only documentation. Models are considered equal to code, as their implementation is automated by a sequence of model transformations that can be accomplished in several ways. In this way, models have the exact meaning of the final application code (i.e., the implementation can be generated from them) and, therefore, models can be used for more than just documenting the software development process [France & Rumpe, 2007]. Models allow us to specify the required functionality and architecture of a system [Atkinson & Kühne, 2003].

Although the notion of model is very old, there is a need for a more rigorous definition in the context of this thesis.

A *model* is *"a simplification of the reality"* [Booch et al., 2005]. A model of a system is *"a description or specification of that system and its environment for some certain purpose"* [OMG, 2003]. According to the definition given by Bézivin and Gerbé [2001], a model is *"a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system. The answers provided by the model should be the same as those given by the system itself, on the condition that questions are within the domain defined by the general goal of the*

system." Finally, Seidewitz [2003] defines a model as *"a set of statements about some system under study (SuS)"*. According to Seidewitz [2003], a model is interpreted as *"a mapping of elements of the model to elements of the SuS such that the truth-value of statements in the model can be determined from the SuS, to some level of accuracy. Colloquially, an interpretation of a model gives the model meaning relative to the SuS."* For example, a kind of model in cereal factory management might describe agricultural business, which would be the SuS in this case. Such a model makes statements on the quality and quantity of the grain, the weight of trucks, etc. We can similarly use a UML model to describe the structure of the software system. In this case, if the SuS is an object- oriented software system, then we could use a UML class model to make statements about the classes of the system and how they are related [Seidewitz, 2003]. Then, for this thesis, we can state that a model is an abstraction or simplification of a system that provides information about it within the context of the intended goals (i.e., a model focuses on important aspects and hide irrelevant ones). An example showing different models describing a software system at different abstraction levels is shown in Figure 2.3. Here, in the case of a software system, the source code is considered to be both the real-world that is being modeled in terms of a UML class diagram, and a low-level model that represents the business.

However, for more complex systems, trying to solve the problem with a single model may result in an extremely voluminous and unmanageable specification. That is, with a single model the system as a whole cannot be understood. Therefore it is better to represent the system through a group of interconnected models, where each model offers a different view of the system to be implemented (thus, each view is focused on a specific part of the system). The different views will allow developers to specify the structure and behavior of the system (in other words, its static and dynamic aspects). Moreover, each model also can be expressed in the same language or in different languages, which will provide adequate flexibility.

Thus, in every system, different types of models can coexist, being detailed at different levels of abstraction. These models can be analyzed and transformed into other models, and it is even possible to generate the final application code from them. Furthermore, these models can be useful to direct the development process and to document a large part of the decisions taken during the project, as well as the resulting implementation.

For this purpose, diagrammatic languages are often used. However, textual models, such as the *XML Metadata Interchange* (XMI) specification [OMG, 2007], are also widely used nowadays. Typically, these kinds of models are transformed into code in order to enable compilation and execution

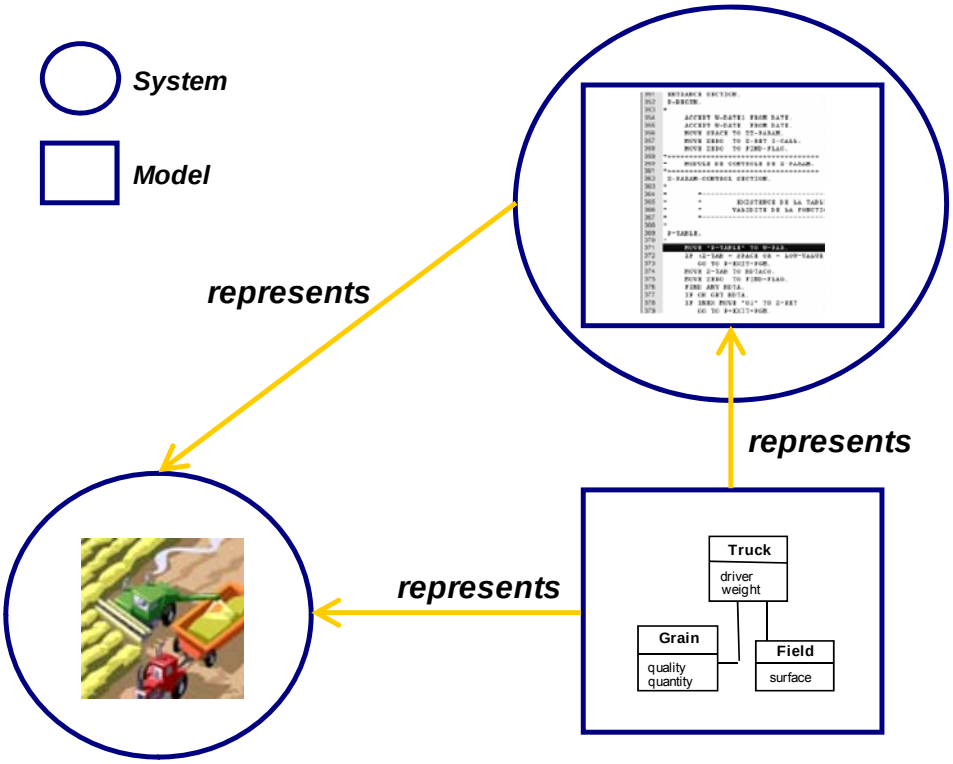


Figure 2.3 Models and systems (adapted from [Bézivin, 2004])

In order to correctly and formally define models it is essential to count on *metamodels*. Metamodels define the abstract syntax of the modeling languages used to define the models. Metamodel define concepts, attributes and relationships that help a model conform more closely to the system that it represents. Metamodels will be further discussed in section 2.3.2.2.

2.3.2.2 Metamodeling Approach for Software Modeling

As mentioned earlier, building metamodels that allow to support and formalize the modeling languages in which the models are based on is essential in MDE. Metamodels enable the definition of a language for expressing a model (i.e., metamodels describe

language constructs for modeling). In general, the metamodel, through the *abstract syntax*, describes the vocabulary concepts, and the relationships and constraints for family models (note that the term *family* is used here to group models that share common syntax). That is, the *abstract syntax* consists of *"a definition of the concepts, the relationships that exist between them and well-formedness rules that state how the concepts may be legally combined"* [Clark et al., 2008].

The OMG's classical framework for metamodeling is based on a four-layer architecture (see Figure 2.4). In the OMG terminology, these four layers are called M0, M1, M2, and M3 [ISO, 2005b].

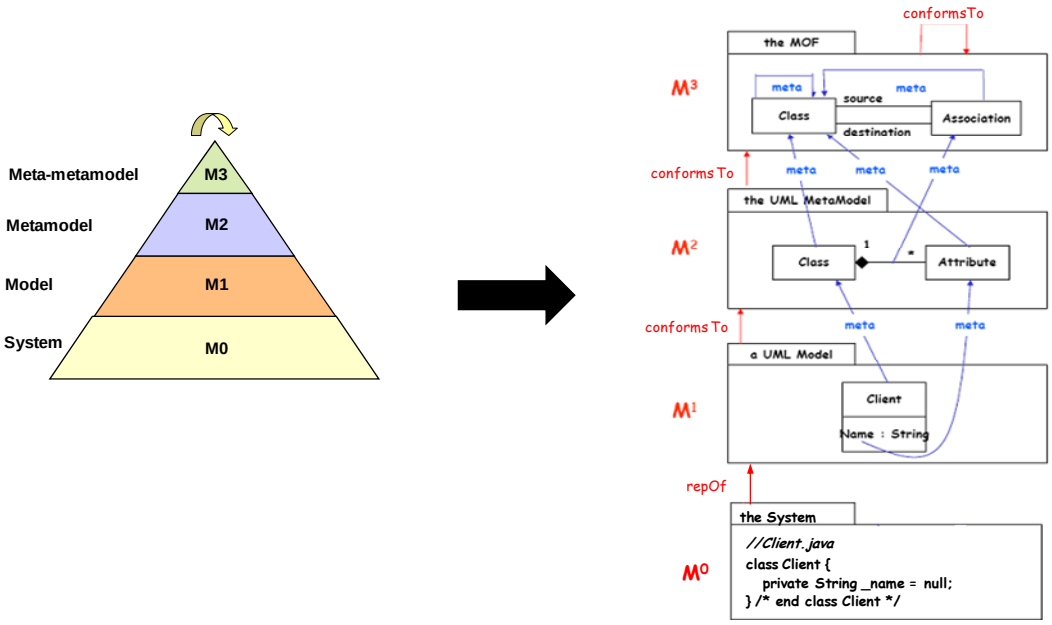


Figure 2.4 OMG's four-layers architecture (adapted from [Bézivin, 2004])

M0: The Information Layer

The information layer is comprised of the data that we want to describe, that is, what is to be modeled [Seidewitz, 2003]. When we are modeling a business, the instances at M0 layer are the items in the real-world business itself (e.g., the IT services, the actual people, the invoices, and the products). When we are modeling software, the instances at M0 layer are the software representations of the real-world items (e.g., the computerized version of the invoices or the orders, the product information, and the

personnel data) [Kleppe et al., 2003]. Thus, this layer holds the actual data, which the software is designed to manipulate [Atkinson & Kühne, 2003].

M1: The Model Layer

The model layer is comprised of the metadata that describes data in the information layer. Metadata is informally aggregated as models (e.g., UML models). This is the layer at which models reside and it holds a 'model' of the data [Atkinson & Kühne, 2003]. The concepts at the M1 layer are all categorizations or classifications of instances at the M0 layer. Likewise, each element at the M0 layer is always an instance of an element at the M1 layer [Kleppe et al., 2003].

M2: The Metamodel Layer

The metamodel layer is comprised of the descriptions that define the structure and semantics of the metadata (i.e., meta-metadata). Meta-metadata is informally aggregated as metamodels, which describe different kinds of data without a concrete syntax or notation (for example, UML). This layer holds a 'model' of the information at M1 (i.e., a model of the models at M1) [Atkinson & Kühne, 2003]. The elements that exist at the M1 layers (classes, attributes, and other model elements) are themselves instances of elements at M2. An element at the M2 layer specifies the elements at the M1 layer. M1 contains the concepts needed to reason about instances at M0, and M2 contains the concepts needed to reason about concepts from layer M1 [Kleppe et al., 2003].

M3: The Meta-Metamodel Layer

The meta-metamodel layer is comprised of the descriptions that define the structure and semantics of the meta-metadata. In other words, it is the abstract language for defining different kinds of metadata. This layer holds a model of the information at M2 and, for historical reasons, it is also referred to as the *Meta Object Facility* (MOF) [OMG, 2006b], the OMG's standard M3 language [Atkinson & Kühne, 2003]. Therefore, every element at M2 is an instance of a M3 element, and every element at M3 categorizes M2 elements. M3 defines the concepts needed to reason about concepts from layer M2 [Kleppe et al., 2003].

There are several definitions for metamodels provided by different sources in the model engineering field. MOF [OMG, 2006b] defines a metamodel as *"a model used to model*

modeling itself." In this way, metamodels provide a platform-independent mechanism to specify [OMG, 2006b]: (i) the shared structure, syntax, and semantics of technology and tool frameworks as metamodels; (ii) a shared programming model for any resultant metadata (for example, using Java, IDL, etc.); and (iii) a shared interchange format (e.g., XML). Mellor et al. [2004] define a metamodel as *"a model of a modeling language."* In [Bézivin, 2005], a metamodel is defined as *"a graph composed of concepts and relationships between these concepts. [...] a metamodel acts as a filter to extract pertinent elements from a system in order to build the corresponding model. Any feature (concept or relationship) not present in the metamodel will be ignored when building the model representing system."* Finally, in the context of the *Model-Driven Development* (MDD), Stahl and Völter [2006] claim that a metamodel defines *"the constructs of a modeling language and their relationships, as well as constraints and the modeling rules, but not the concrete syntax of the language."* In other words, a metamodel defines *"the abstract syntax and the static semantics of a modeling language."*

The *concrete syntax* or notation of a language facilitates the presentation and construction of models in the language to humans. Different concrete syntax forms may have a common abstract syntax. For example, a metamodel can be expressed in different notations (e.g., in a graphical-based notation or in a textual-based notation), or even many different graphical-based notations may use the same metamodel. Thus, the concrete syntax may be defined by a modeling language but is not part of the metamodel. In addition, the distinction between abstract syntax and concrete syntax is very important in our context, because the metamodel (and not the concrete syntax) is the basis for automated, tool-supported processing of models [Stahl & Völter, 2006].

In summary, the abstract syntax of a modeling language deals with the structure of concepts in a language without taking their presentation and meaning into account. It is important to remark that the static semantics is quite different from the semantics that is included in the abstract syntax of a metamodel. The *static semantics* is the definition of concepts in the language providing constraints and rules, which dictate whether or not an expression of the language is well-formed. Conversely, the semantics that is part of the abstract syntax of a model conveys little or even no information about the meaning

of the concepts in the language. For example, in the Eclipse platform[21], there is no mean to add a description stating the meaning of a concept included in the abstract syntax represented by metamodel.

On the other hand, the *metamodel semantics* is embedded in the transformation definition of a metamodel. In other words, the metamodel semantics states how each concept in the metamodel should be interpreted and in what thing the concept is transformed. Then, since the metamodel semantics enables to be clear about what the language represents and means, it is essential to communicate the meaning of models among stakeholders in a software project. Otherwise, assumptions may be made about the language that leads to its incorrect use [Clark et al., 2008]. That is, all transformations from one source model to other target models must keep the same meaning of the metamodel of which the source model conforms to.

Therefore, it must be noted that the metamodel semantics is not part of the abstract syntax of a language although the abstract syntax model is a pre-requisite for defining the metamodel semantics, as the metamodel semantics adds a layer of meaning to the concepts defined in the abstract syntax [Clark et al., 2008]. Then, the metamodel semantics could be considered a meaning in the sense of an interpretation of the model as explained by Seidewitz [2003]: *"Because a metamodel is a model of a modeling language, an interpretation of a metamodel is a mapping of the metamodel elements to the modeling language elements, such that we can determine the truth value of statements in the metamodel for any model expressed in the modeling language. Because a metamodel is a specification, a model in the modeling language is valid only if none of these statements are false."* That is, there may be several interpretations of the same model. For example, a logical class model could be interpreted as the design for multiple platforms or technologies. Therefore, throughout this thesis, we will use the term 'metamodel semantics' to refer to the interpretation of the model that defines the model transformation operation.

Finally, we agree with Bézivin [2004] and Favre [2004] in that a model 'conforms to' its metamodel rather than being an 'instance of' it. Both authors recommend the relationship 'conform to' instead of 'instance of' in the context of relating models to

[21] http://www.eclipse.org/

each other in order to distinguish the conformance relationship between models from the instantiation relationship known from OO technology (i.e., between objects and classes) [Kühne, 2006] [Stahl & Völter, 2006]. The importance of using the relationship 'conform to' rather than 'instance of' is also stressed by Gašević et al. [2007]. In this way, in order to be valid, models have to conform to its metamodel. Models conformant to common metamodels can be used as reusable artifacts for future software projects.

2.3.2.3 Domain-Specific Modeling

Different concerns within the software modeling need to use and integrate different specialized languages in order to be effective in tackling a development project. Unlike general purpose programming languages, *Domain-Specific Languages* (DSLs) are oriented towards a particular domain. Examples of DSLs are the *HyperText Markup Language* (HTML), used as the mark-up language for hypertext on the Web, and *Backus-Naur Form* (BNF), used for describing grammars. By making the notations and concepts of a problem domain available and understandable to all stakeholders in a development project, DSLs allow domain experts to recognize its 'domain language' and allow software systems to be expressed more concisely and directly than in general purpose languages [Stahl & Völter, 2006].

In this respect, *Domain-Specific Modeling* (DSM) is an approach that raises the level of abstraction beyond programming by specifying the software application directly using domain concepts [DSM Forum website]. DSM allows developers to work with graphical models of the problem to solve, and helps to hide the implementation concepts from the models. In DSM, models can be tailored to accurately match the domain's vocabulary. In DSM, metamodels describe the domain concepts and their relationships, as well as the semantics and constraints associated with the concepts.

In the context of MDD, DSLs are composed of a metamodel, including its static semantics, and a corresponding concrete syntax [Stahl & Völter, 2006], specially designed for the MDD solution. Instead of using a general purpose modeling language for software development, a DSL is used that is itself designed to define the specific problem. Eclipse and Microsoft's Domain Specific Language Tools (DSL Tools) [Cook et al., 2007] are environments that enable us to create our own graphical domain

specific modeling language and editor where the complexity to create them is greatly reduced.

Cook [2004] defines DSLs as "*languages that instead of being focused on a particular technological problem such as programming, data interchange or configuration, are designed so that they can more directly represent the problem domain which is being addressed."* A DSL is *"a custom language that targets a small problem domain, which it describes and validates in terms native to the domain"* [Cook et al., 2007]. In other words, DSLs enable us to work within a particular area of interest.

Due to their proximity of the concepts of a particular domain, DSLs make it much easier to discuss the software at the requirements level, and to manage changes in an agile way. DSLs may also be considered as a form of *ontological metamodeling* since they are concerned with describing what concepts exist in a certain domain and what properties they have (similarly, ontologies capture the knowledge of real-world domains, independently from specific application needs) [Atkinson & Kühne, 2003].

2.3.2.4 Model Transformations

In the context of MDE, a development process can be modeled as a set of model transformations that take source models as input and produce target models as output using a set of transformation rules (*transformation definition*) [Sendall & Kozaczynski, 2003]. A transformation (or mapping) implicitly or explicitly defines a relationship between the source and the target models [Stahl & Völter, 2006], where the transformation itself is also a model. This relationship may represent a model translation (relationship between models in the same language) or a language translation (relationship between models that are expressed in different languages) [Kent, 2002]. Model transformations are usually based on a source metamodel and the transformation rules can only be based on the metamodel constructs.

According to Czarnecki and Helsen [2006], model transformations may have different applications:

- Generating lower-level models, and eventually code, from higher-level models.

- Mapping and synchronizing among models at the same level or different levels of abstractions.
- Creating query-based views of a system.
- Model evolution tasks, such as model refactoring.
- Reverse engineering from lower-level models or code into higher-level models.

In order to perform a *model transformation*, we must have a clear understanding of the *abstract syntax* and *semantics* of both models, that is, source and target models. When defining *transformations*, there are three different architectural approaches [Sendall & Kozaczynski, 2003]:

- *Direct model manipulation*. It defines the access to an internal model representation and the ability to manipulate the representation using a set of procedural *Application Programming Interfaces* (APIs). The language used to access and manipulate the APIs is commonly a *General Purpose Language* (GPL) such as Visual Basic or Java. However, UML models and the *UML's action language* (xUML) [Raistrick et al., 2004] could be used as well.
- *Intermediate representation*. It defines the exporting of the model in a standard form (typically XML) that may be transformed by external tools. For example, many UML tools can export and import models to and from XMI (the XML-based standard for interchange of UML models).
- *Transformation language support*. It defines a language that provides a set of constructs for explicitly expressing, composing, and applying transformations. The desirable characteristics for model transformation languages would be: (i) *Preconditions*. They describe the conditions under which the transformation produces a meaningful result; (ii) *Composition*. Since it is usually easier to compose components than to build 'things' from basic parts, combining existing transformations to build new composite ones is a desirable feature; (iii) *Form*. The accessibility and acceptance of a language depends on its form, and the graphical representations of models are preferred to fully textual representations; and (iv) *Usability*. Strongly affected by whether the language is declarative (i.e., makes the language more concise, making implicit a number of issues of the

transformation algorithm) or imperative (i.e., offers a familiar paradigm for composing transformation rules, that is, sequence, selection, and iteration), and involving language's purpose and the preferences and backgrounds of its users, who might balance ease-of-understanding, precision, concision, and ease-of-modification differently.

In MDE, there are two different kinds of model transformations (see Figure 2.5): *Model-to-Model transformation* (M2M) and *Model-to-Text transformation* (M2T).

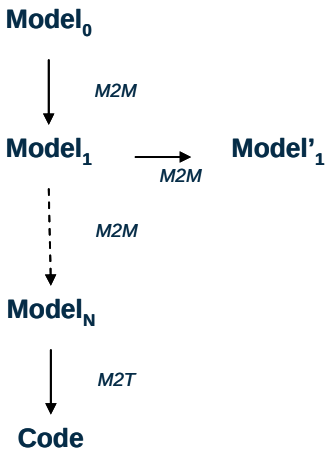


Figure 2.5 Model transformations

A M2M transformation creates another model based on the target metamodel. Here we have different models at different levels of abstraction. In M2M, we can specify transformations horizontally and vertically. On the one hand, *horizontal transformations* describe relationships between different views of a problem domain: the models describe different aspects of the system, but at the same level of abstraction. On the other hand, *vertical transformations* relate models at different levels of abstraction: the models are refined from higher to lower levels of abstraction, and at the lowest level, models consider implementation technology issues [Sendall & Kozaczynski, 2003]. Mapping between a specification and a design, and between design and implementation are examples of vertical transformations. It is important to remark that vertical transformations may also go in the reverse direction (*reverse engineering*), for example, from implementation to design [Clark et al., 2008]. The *Extensible Stylesheet Language*

Transformations (XSLT) [Clark, 1999], Medini QVT[22] and the *Atlas Transformation Language* (ATL[23]) are examples of M2M approaches.

A M2T transformation (also known as *model-to-platform* or *model-to-code transformation*) generates the code (i.e., just strings) that is based on a platform. For this type of transformation we do not need a target model, because usually we are dealing with simple text replacements of a programming language. JET[24], MOFScript[25] and OLIVANOVA[26] are examples of M2T approaches.

On the other hand, model transformations can be also categorized depending on the scope of their effect on a given model [Alanen, 2007]: (i) a *mapping transformation*: this approach translates each element from a source model into zero, one or more elements of a target model. The source and target models may be described in the same of in different modeling languages. In a mapping translation, the source model is not modified; and (ii) an *update transformation*: this approach modifies a model in place; it adds, deletes and updates elements in one model. The source and target models are the same and the effects of the transformation are visible while performing the transformation. There can be two kinds of update transformations: to modify and already existing element or to create a new element of the same type followed by the deletion of the initial element.

XSLT

The *Extensible Stylesheet Language Transformations* (XSLT) is a language for transforming XML documents into other XML documents [Clark, 1999], widely used in the development of data-intensive applications. An XSLT stylesheet is composed of a set of rule templates. Each rule template matches elements in the source model, and produces output elements to the target model.

The benefits of using XSLT have been explored in [Li et al., 2011]: (i) All major *Computer-Aided Software Engineering* (CASE) tools can import and/or export models

[22] http://projects.ikv.de/qvt/
[23] http://www.eclipse.org/gmt/atl/
[24] http://www.eclipse.org/modeling/m2t/?project=jet#jet
[25] http://www.eclipse.org/gmt/mofscript/
[26] http://www.care-t.com/

as XMI files; (ii) XSLT is the most common and powerful language for XML transformation; (iii) XSLT (Xpath) has strong support to complex pattern matching; (iv) XSLT has many industrial strength implementations, including commercial and open source tools; (v) XSLT can also be embedded in Java; and (vi) XSLT can be easily executed and integrated into different system environments and platforms, without additional packages and libraries.

2.3.3 Model-Driven Architecture

The *Model-Driven Architecture* (MDA) approach [OMG, 2003], defined and supported by the OMG, defines a particular MDE process aimed at separating the business logic from the technological platforms. Thus, organizations can use MDA to meet the integration challenges posed by new platforms, while preserving their investments in existing business logic. MDA is a model-driven approach for software system development in which models direct the course of understanding, design, construction, deployment, maintenance and modification of systems. MDA is built on the solid foundation of well-established OMG standards, including: UML [OMG, 2010c], MOF [OMG, 2006b] and XMI [OMG, 2007], among others.

MDA proposes three modeling layers specified as MOF metamodels, namely, ordered from highest to lowest levels of abstraction: *Computation Independent Models* (CIMs), *Platform Independent Models* (PIMs), and *Platform Specific Models* (PSMs). Different M2M transformations among these abstraction layers can be defined either top-down, bottom-up or horizontally. Commonly, each CIM (model gathering high-level system requirements) is transformed into one or more PIMs (platform-independent architectural models). Similarly, each PIM is transformed into one or more PSMs (one for each target platform). PSMs are commonly low level models, enabling the definition of direct M2T transformations for automatically generating the final system implementation (including code, documentation, etc.).

A model of a system in MDA is defined as *"a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language"* [OMG, 2003]. All metamodels must be written in the MOF language

to be MDA compliant (Figure 2.6). That is, in MDA, all modeling languages are defined either using the standard MOF metamodeling language or an extension of an existing UML metamodel defined as a profile. Andrew Watson [2008], OMG's Vice-President and Technical Director, states that *"MDA uses MOF-defined models to create and manipulate precise, detailed, machine-readable descriptions of application structure and behavior that are independent of what programming languages, operating systems or database may be used to implement them."* Therefore, we could conclude that the key MDA standard is MOF and not UML, like some people still believe.

We can use MDA to gain control over and systematically improve the whole lifecycle of IT solutions: from modeling the overall business (facilitating effective communication between business analysts and IT members and capturing specific solutions requirements) to developing, deploying, integrating, and managing many kinds of software artifacts [Guttman & Parodi, 2007].

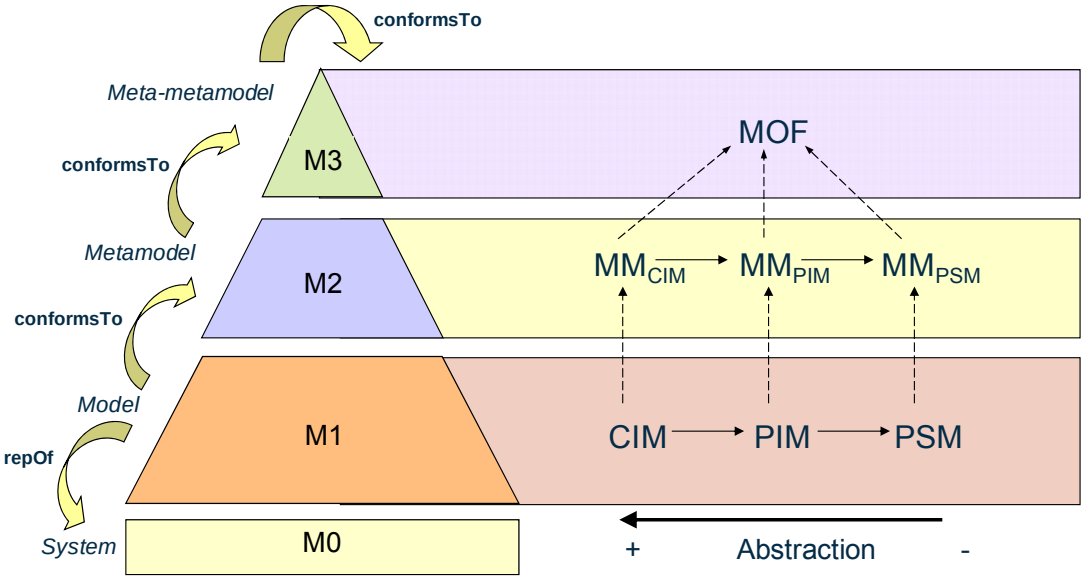


Figure 2.6 MDA and the OMG's four-layers metamodeling pyramid as depicted in [Vicente-Chicote & Alonso, 2007]

2.3.4 Matching Ontologies and Conceptual Models with Metamodels

As stated previously, metamodeling is one of the most important concepts of MDE. In the context of MDE, we must be clear about the structure of a domain (that is, the

ontology) related to the system to build, so that we can formalize this structure or its relevant part in terms of a metamodel for any attempt of automation in the SDP [Stahl & Völter, 2006]. According to MDE, ontologies (that is, the OE part) would cope with the 'repOf' (representation of) relation that exists between *models* (i.e., the M1 layer in the OMG's four-layer architecture) and *systems* (i.e., the real-world, which means the M0 layer in the OMG's four-layer architecture) [Bézivin, 2005]. However, we note a continuous confusion between the terms 'metamodel' and 'ontology'. Several authors have tried to compare ontologies and metamodels, for example Ruiz and Hilera [2006] and Henderson-Sellers [2011]. In this thesis, just like the approach of [Ruiz & Hilera, 2006], we consider that ontologies and metamodels have different purposes: ontologies are descriptive and they belong to the structure of a domain (that is, the real-world), whereas metamodels are prescriptive and they belong to the MDE solution. However, conceptual modeling of information systems is comparable with ontologies, because they share some modeling principles. A conceptual model captures the semantics for a given application domain, and ontologies are supposed to capture semantics about real-world domains, independently from specific application needs.

Similarly to the approach of Bézivin [2009], we also consider a metamodel as a simplified ontology in the sense that it is a set of concepts and relations between these concepts. Therefore, ontologies can act as the basis for defining DSLs in terms of a metamodel in order to generate conceptual models for the implementation of specific information systems. Since a DSL describes domain knowledge it requires detailed knowledge about the domain (that is, the ontology). Just as remarked by Devedžić [2002], if ontologies are not used, different conceptual models of the same domain could be incompatible, even if they use the same DSL for the implementation of the related information systems.

In this vein, Henderson-Sellers [2011] establishes a formal relationship between metamodels and ontologies in order that the adoption and integration of ontological thinking and theory into SE will result in theoretically sound software development methodologies that are also practical for industry usage. Because of ontologies can be understood by both human beings and computers, they can be used to mediate communication within an information system, between people themselves or between

people and software systems. For example, Wand [1996] uses an ontology to model information system concepts. The proposed ontology use concepts from the Bunge's Ontology [Bunge, 1977] [Bunge, 1979]. The author identifies three aspects of information systems: (i) Deep structure (*meaning*): it represents the aspects of the information system that reflect the represented domain; (ii) Surface structure (*interface*): it represents the user interface characteristics of the information system; and (iii) Physical structure (*technology*): it represents the technical means employed in the implementation.

In recent years, there are works that discuss the contributions of ontologies to the model-driven software development approach [Decker et al., 2005] [Goknil & Topaloglu, 2005] [Silva Parreiras & Staab, 2010]. In this vein, for example, the OMG's *Ontology Definition Metamodel* (ODM) [OMG, 2009] is a proposal for ontology modeling that enables capabilities for MDA-based SE. From a SE perspective, ontologies are considered CIMs. According to [OMG, 2009], ODM provides MDA with *"the formal grounding for representation, management, interoperability, and application of business semantics."* ODM defines a metamodel for OWL, and describes, for example, mappings from ODM OWL models to UML models. Therefore, it seems that the application of the MDA approach in conjunction with ontologies may help software engineers developing and managing complex systems. On the one hand, the use of models and metamodels for software development is an established practice in SE, and on the other hand, the use of ontologies as modeling and reasoning frameworks for the management of models has been successfully reported and promoted by researchers over the last decade. Furthermore, as remarked previously, ontologies provide shared domain conceptualizations representing knowledge that enable software engineers to model the problem domain as well as the solution domain.

Most studies related to the integration of ontologies into the model-driven approach use ontologies to define DSLs and domain conceptual modeling, as for example, the works presented by Walter et al. [2009], Durak et al. [2006] and Garrido et al. [2007], among others. In this research area, a DSL could be considered as the joint use of a metamodel and an ontology in which ontologies provide the semantic context (i.e., knowledge modeling) for the models providing reasoning capabilities, model enrichment, model validation, etc.

Another common approach is to use ontologies as a basis for model transformations. As stated previously, model transformations are a fundamental mechanism in the model-driven approach and these transformations rely on semantics that is not part of a metamodel (that is, the *metamodel semantics*). The metamodel semantics necessary to support model transformations at the metamodel level (i.e., metamodel mappings) can be added and expressed in terms of ontologies. In this way, the ontological model definitions may be used, for example, to transform from CIMs to PIMs or from PIMs to PSMs by using query statements, transformation rules and models defined in ontology languages such as OWL. Ontology-based transformations allow the seamless and coherent transition from one development focus to another [Pahl, 2007].

Finally, *ontology-aware MDE* is a research area presented as a new architecture where ontologies and automatic reasoning play a key role in MDA and its generalization MDE [Assmann et al., 2006] [Živković et al., 2008]. The idea of ontology-aware MDE is to benefit from semantic technologies (*the ontology aspect*). Thus, MDE is extended to be considered as ontology-aware. As an extension to models on different levels of the MDE architecture (i.e., models, metamodels and the meta-metamodel), in ontology-aware MDE architecture, an ontology repository serves as a store of the semantics of each level in form of descriptive analysis models. The semantics is formally described in terms of ontologies, and reasoning on ontologies is part of the ontology-aware mechanisms. Model and ontology editors are used for the management of models, metamodels and ontologies.

2.4 IT Service Management

The concept of service is understood differently depending on the domain or application area, involving a certain confusion that has been explored by Jones [2005] and Ferrario and Guarino [2009]. For example, The *Service Oriented Architecture* (SOA) is an approach to structure software systems by grouping functionalities into manageable services with well-defined interfaces that can be invoked remotely, where a service represents how its consumers wish to use it [Jones, 2005]. Within ITSM, and throughout this paper, the term ‘service’ should be understood as an overall IT service, such as software distribution or server support [Black et al., 2007]. Therefore, the term

does not refer to Web services in the SOA context since this approach is outside of the scope of our work. However, it is possible to use SOA and principles to develop flexible, re-usable IT services that are common and can be shared and exploited across many different areas of the business [OGC, 2007a].

The *IT Service Management Forum* (itSMF[27]) is an independent organization dedicated to promoting a professional approach to ITSM. The itSMF defines an IT service as *"a service provided to one or more customers by an IT service provider. IT services are based on the use of information technology and supports the customer's business processes. IT services are made up from a combination of people, processes and technology and should be defined in a Service Level Agreement (SLA)"* [itSMF, 2007a]. A SLA represents a formal agreement between an *IT service provider* and a *customer*. The SLA describes a level of assurance or warranty with regard to the level of service quality for each of the services delivered to the business (customer). In this context, IT services can be considered as commitments just like the approach of Ferrario and Guarino [2009].

According to the ISO/IEC 20000 standard [ISO/IEC, 2005a], an ITSMS must include *"policies and a framework to enable the effective management and implementation of all IT services"*:

- *Management Responsibility:* Through leadership and actions, IT service providers must prove its commitment to developing, implementing and improving its ITSM capability within the context of the organization's business and customers' needs.
- *Documentation:* IT Service providers must provide documents and records to ensure effective planning, operation and control of ITSM.
- *Competence, awareness and training:* All ITSM roles and responsibilities must be defined and maintained together with the competencies required to execute them effectively. Also, staff competencies and training needs must be reviewed and managed to enable staff to perform their role effectively. Finally, IT service providers must ensure that its employees are aware of the relevance and

[27] http://www.itsmfi.org/

importance of their activities and how they contribute to the achievement of the ITSM objectives.

There are several well established good practice frameworks to create an effective ITSMS such as ITIL. Nowadays, ITIL is the best known and most widely accepted guidance and it has become the de facto standard for ITSM, providing *"a detailed description of a number of important IT practices, with comprehensive checklists, tasks, procedures and responsibilities which can be tailored to any IT organization"* [OGC, 2007d].

ITIL version 3, also known as ITIL V3, is an enhanced and consolidated framework that proposes a new approach to ITSM by considering the lifecycle of a service. Provided that ITIL V3 is the most complete and up-to-date version of this ITSM framework, and since the *Office of Government Commerce* (OGC) has announced its plans for the withdrawal of publications and qualifications of ITIL version 2 (complete in the middle of 2011)[28], we selected ITIL V3 for our ontology approach.

2.4.1 The Information Technology Infrastructure Library

The *Information Technology Infrastructure Library* (ITIL) was originally developed by the *Central Computer of Telecommunications Agency* (CCTA, later to become part of the OGC), and started by the late 1980s and early 1990s by documenting an approach to the ITSM needed to support business users. The library originally consisted of approximately forty books providing guidance to all areas of local and central UK government. It ware subsequently adopted and used by many organizations within the private sector as well. In 1991, a user forum, the *Information Technology Information Management Forum* (ITIMF), was created to bring ITIL users together to exchange ideas and learn from each other, and would eventually change its name to the itSMF. A formal standard for ITSM, The British Standard 15000 (BS15000), largely based on ITIL practices, was established and followed by several national standards in different countries. Since then, the ISO/IEC 20000 standard was introduced and gained rapid recognition globally [OGC, 2007d]. ISO/IEC 20000 specifies a set of interrelated

[28] http://www.ogc.gov.uk/itil_ogc_withdrawal_of_itil_version2.asp

management processes and differs only in minor ways from BS15000. In this vein, the ITSM structure can be seen as a pyramid with the international standard ISO/IEC 20000 at the summit (Figure 2.7). Below the summit we can find the layer of ITIL best practices, which helps to ensure and demonstrate that the requirements of the standard are being met. At the lowest level is the layer of the customization of ITIL to meet the particular needs of an organization, which is the broad base of ITIL implementation.

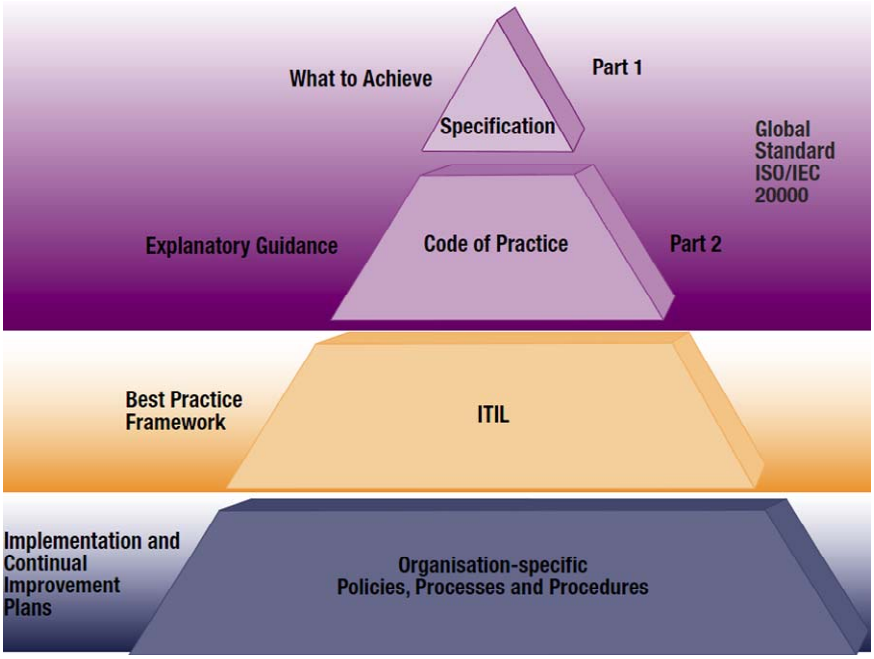


Figure 2.7 IT Service Management pyramid as depicted in [ISACA, 2008]

ITIL V2 began in the mid 1990s, until 2004. This version was a targeted product explicitly bridging the gap between technology and business, and with guidance focused strongly on the processes required to deliver effective services to the business customer [OGC, 2007d].

In 2004, the OGC began the second major refresh initiative of ITIL, that is, ITIL V3, in recognition of the massive advancements in technology and emerging challenges for IT service providers. ITIL V3 was published in 2007 offering best practice guidance applicable to all types of organizations that provide services to a business. ITIL V3 provides a recognized set of standards for bringing improvements to our *IT application support services*. In this way, ITIL can be used to integrate, manage, measure and

therefore improve application support. By using ITIL, organizations may reduce costs and improve service performance in a well-defined manner.

The Table 2.2 shows the key ITIL V2 to V3 concept differences [itSMF, 2006]. First, the term *alignment* has been replaced with the concept of *integration*. Second, *value chain management* in V2 means a business customer being supported by a single internal IT service provider whereas *value service network integration* in V3 means: (i) a business customer being provided service by internal IT service providers; (ii) those provided by a shared service model to multiple business units; (iii) the option of using different external outsourcing options; and (iv) leveraging a software as a service model. Third, *linear service catalog* in V2 means a brochure of IT services where IT publishes the services it provides with their default characteristics and attributes where as *dynamic service portfolio* in V3 means the product of a set of process where service strategy and design conceive of and create services that are built and transitioned into the production environment based on business value. Forth, ITIL V3 core books core books are structured around a *service lifecycle*. This new structure organizes the ITIL V2 processes with additional content and processes [DuMoulin, 2007].

Table 2.2 Key differences in ITIL

ITIL V2	ITIL V3
Business and IT Alignment	Business and IT Integration
Value Chain Management	Value Service Network Integration
Linear Services Catalogues	Dynamic Service Portfolios
Collection of integrated processes	Service Management Lifecycle

The service lifecycle of ITIL V3 contains five elements that are depicted in Figure 2.8. Each element relies on service principles, processes, roles and performance measures. Furthermore, each part of the lifecycle exerts influence on the other and relies on the other for inputs and feedback. Thus, a constant set of checks and balances throughout the service lifecycle ensure that as business demand changes with business need, the services can adapt and respond effectively to them. Furthermore, all services must provide measurable value to business objectives and outcomes, and this principle could be seen as the heart of the service lifecycle.

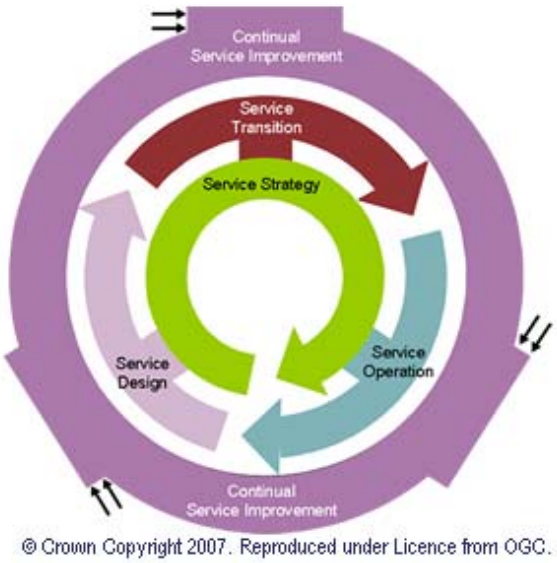


Figure 2.8 The ITIL service lifecycle

To fully benefit from ITIL best practices, business stakeholders should model their own processes from ITIL perspective (i.e., ITIL can be adopted, but it can be adapted according with our own interests) and share this view with all IT stakeholders in order to develop information systems that support these processes. In this way, organizations might reach their objectives that include the next five essential elements: objectives have to be *Specific*, *Measurable*, *Appropriate*, *Realistic* and *Time-bound* (SMART) [itSMF, 2007a].

2.4.2 ITSM Processes

ITIL describes a process as *"a structured set of activities designed to accomplish a specific objective. A process takes one or more defined inputs and turns them into defined outputs. A process may include any of the roles, responsibilities, tools and management controls required to reliably deliver the outputs. A process may define policies, standards, guidelines, activities, and work instructions if they are needed"* [itSMF, 2007a]. Therefore, processes and functions co-exist alongside each other, but we have to be clear of the distinction between the two terms [Ferris, 2008].

With this definition of a process, we can identify the following qualities:

- It should be able to change a group of inputs into a group of outputs.

- It should provide an added value.
- It is made up of a group of coordinated internal activities.
- The activities are carried out by resources: people, individually or in groups (areas, departments, organizational units, etc.), and computer tools. The resources require a lot of knowledge and information, and are closely related with the people, systems, processes and technologies of an organization [itSMF, 2007a].
- The group of activities can graphically be represented in the form of a *workflow*.

The processes should also have the following characteristics:

- **Repetition:** the processes are created to produce a result that can be repeated.
- **Variability:** each time a process is repeated small variations in the activities can be produced, which at the same time generate variations in the results obtained (in the characteristics of the outputs).

This characteristic of variability is the one that can affect the level of satisfaction of the customers and users of the service as to the exit of the process. This is why, it is necessary to establish a system of measurement and of control, ITSMS, which allows knowing this variability and determining the acceptable margins. This way, the result of the process is kept delimited and its success is guaranteed. An ITSMS must be able to generate a series of records that make up the evidence that the processes work, so that these records can be evaluated and contrasted with the objectives that the organization wants.

Finally, the RACI matrix is a model used to help define roles and responsibilities in the activities that are part of an ITSM process. The RACI matrix (see Table 2.3) is a formal way of establishing the role for each stakeholder that participates in a specific process. RACI stands for *Responsible*, *Accountable*, *Consulted* and *Informed* [OGC, 2007d]. ITIL supports the RACI model [OGC, 2007b]. The responsible is attributed to the person who gets a process activity done (i.e., the stakeholder that is responsible for actually doing it). Accountable means 'the buck stops here' (i.e., this is the stakeholder that provides direction and authorizes an activity). The other two roles, consulted (a stakeholder that has needed input about the activity) and informed (a stakeholder that

needs to be kept informed about the activity), ensure that everyone who needs to be is involved and supports the process.

Table 2.3 RACI Matrix

Acronym	Description
R	Responsibility: correct execution of process and activities
A	Accountability: ownership of quality, and end result of process
C	Consulted: involvement through input of knowledge and information
I	Informed: receiving information about process execution and quality

2.4.3 Ontologies for ITSM

Recently, there has been an increasing interest in the use of ontologies to various aspects of ITSM. The importance of using OE to automate and validate service process models is remarked by Verma and Sheth [2007] and Talantikite et al. [2009]. As stressed by Mizoguchi and Ikeda [1996], OE can provide *"a basis of building models of all things in which computing is interested."* A formal description of the functionality of a service process is crucial for service process reuse [Verma et al., 2005], whereas a formal description of the data that the service management processes exchange is a key requirement for interoperability [Nagarajan et al., 2006]. Also, if IT service providers define formally SLAs and quality-of-service attributes, they could different themselves from their competitors [Cardoso et al., 2004] [Oldham et al., 2006].

Table 2.4 Related work about Ontologies in association with ITSM

Author and Year	Feature
Bartsch et al. [2008]	Ontology-based hierarchical service decomposition
Black et al. [2007]	ITSM integration model
Freitas et al. [2008]	UML-based ontology for IT Services
Ghedini and Gostinski [2008]	Ontology-based framework for business-IT alignment
Goeken and Alter [2009]	'Ontological metamodeling' approach to IT governance
Graupner et al. [2009]	Ontological approach to template-based framework to enable making processes, from best practice frameworks, actionable
Paschke and Bichler [2008]	Ontological approach to SLA management
Savvas and Bassiliades [2009]	OWL ontology for administrative procedures and OWL-S service models

Several ontology-based approaches to IT service quality improvement are given in Table 2.4. For example, a proposal of an ontology for ITSM can be found in [Freitas et al., 2008]. This work describes a generic ontology for IT services in terms of UML models, where the *Object Constraint Language* (OCL) [OMG, 2010b] is used for the constraints. Savvas and Bassiliades [2009] propose an ontology in OWL that provides specific knowledge for administrative procedures, which are mapped into OWL-S models.

Bartsch et al. [2008] propose an ontology-based hierarchical service decomposition and identification approach to support service providers in managing their operation service processes. The authors propose three layer process model hierarchy which uses structured knowledge about the respective service process domain to decompose a service process into elemental service process steps and subsequently identify alternative services.

Ghedini and Gostinski [2008] propose a framework using ontologies to provide business-IT alignment. In order to build the ontologies, they use ITIL V2 to obtain concepts related to ITSM using a subset of vocabulary of a business domain ontology related to the biggest public bank of Brazil. The proposed framework helps the concrete realization of governance models in the sense of understanding the effects between business and IT purposes, but their work is not focused on implementations of the ITIL processes.

Graupner et al. [2009] present an approach to bridge the gap between the abstractions available in best practice framework, such as ITIL, and actions that have to be performed by humans or systems. An ontology-based approach is used to represent ITIL processes so that they can be enriched with actionable information.

Goeken and Alter [2009] propose an 'ontological metamodel' of COBIT framework [ISACA, 2007] to IT governance improvement. According to the authors, ontological metamodels deal with the classification of model elements according to their content providing theoretical foundation, and analysis, comparison and integration capabilities.

In the context of SLAs, Paschke and Bichler [2008] propose *ContractLog*, a derivation rule-based language of knowledge representation concepts for SLA management. The *rule-based service level management tool* (RBSLM) has been implemented to help

designers representing SLA rules. Their proposal is a XML-based language that provides high levels of extensibility and support for contractual agreements definitions, although they do not consider ITIL to implement it.

Finally, Black et al. [2007] propose an integration model that tries to cover the entire ITSM space. The model shows how to develop and describe IT solutions, but does not prescribe a specific solution or technology. The model provides a structure that allows users to describe *what* the service is and *how* it is delivered. Ontologies provide expressive depth and potential for inference or tool-assisted realization of facets of the proposed integration model.

Chapter 3

Onto-ITIL: An Ontology-based and Model-driven Approach for ITSMSs

> "–Marco, you should not worry about reaching the target. Just concentrate on the process of reaching it." *La cena secreta*
>
> Javier Sierra (1971-), *Spanish writer*

In this chapter we introduce the approach followed to build Onto-ITIL, an ontology-based and model-driven approach for ITSMSs based on the *ITIL V3 Service Management Model*. Onto-ITIL formalizes the ITSM domain knowledge following best practices provided by the ITIL V3 framework. Onto-ITIL also provides the necessary mechanisms for managing interoperability and consistency checking to serve as a knowledge base for ITIL process implementations. This approach enables IT service providers to add semantics and constraints to the data associated with the different ITIL processes in order to share and reuse information in a homogeneous way.

3.1 Introduction

In an increasingly technology-driven world, organizations must assess the efficiency and quality of their services in order to enhance their competitiveness and performance. Business is what defines the requirements of the information systems needed to automate business activities and, therefore, such systems must be designed to support business processes [Eriksson & Penker, 2000]. However, the integration between business needs and existing technologies is still a challenging issue [Liu & Zhu, 2009]. More frequently than desired, information systems do not meet business requirements and, as a result, many organizations perceive IT as a limitation rather than a benefit for their business growth [Telefónica, 2010]. In other words, generally, business and IT do not share challenges and goals required to achieve a *Sustained Competitive Advantage* (SCA) [Wade & Hulland, 2004]. In order to address this problem, ITSM aims to ease the integration of business and IT in terms of services that

can be managed as another business unit. IT services are recognized as crucial, strategic, organizational assets that must be managed for business success [Black et al., 2007].

Nowadays, the complexity of service management remains a challenge, even when adopting best practices for ITSM. The main reason is that ITSM guidelines and models are commonly specified using natural language or graphical representations, both lacking clearly defined semantics. In fact, natural language specifications can lead to different representations and interpretations [Thomas & Fellmann, 2009], making it difficult to obtain equivalent machine-readable specifications. For example, what should be classified as *incident* in the ITSM domain?; what specific information and tasks are associated with the incident management process?; which of these tasks could be automated using a computer tool?; what metrics (name and description) should be included in the incident management process in order to measure it?; what are the different categories for ITSM metrics?; what are the critical success factors (CSFs) in the incident management process for a specific IT service provider?; and how those metrics are related to each CSF in the incident management process?

To overcome this issues, the proposed approach presented in this thesis relies on: (i) OWL, which provides automated and efficient reasoning facilities; (ii) SWRL, which enables the definition of semantic constraints and knowledge inference rules; (iii) SQWRL for knowledge retrieval; and (iv) MDE for the formalization of a domain or its relevant part in terms of a metamodel for any attempt at automation. The open source Protégé-OWL tool has been selected in this thesis as an ontology editor to create the required ontologies. We use UML class diagrams to present the proposed ontology in a graphical way. In this vein, UML classes represent OWL concepts, UML associations correspond to object properties, UML attributes represent datatype properties, and UML inheritance is used for subclass relationships.

3.2 Onto-ITIL Principles

The ITSM model proposed in this work is based on the structure illustrated in Figure 3.1, which relies on five concepts (*IT service*, *Process*, *Metric*, *Activity* and *Application*) and the four relations defined among them (*managedBy*, *measures, coordinatedBy* and *implements*). *IT services* are *managed by Processes* which are

measured using appropriate *Metrics*. In turn, *Processes coordinate* a set of *Activities*, which can be (fully or partially) *implemented by Applications*. In our context, an *Application* is a piece of software that provides the functionality required by an *IT service*. Each *Application* may support one or more *IT services*.

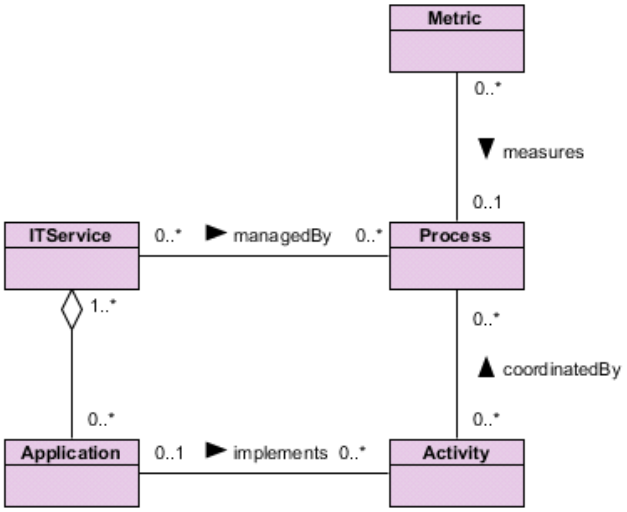


Figure 3.1 Onto-ITIL principles

In order to further detail the most relevant concepts related to the Onto-ITIL principles, some formal definitions are included next.

Definition 1. Let $S = \{S_1, S_2, \ldots, S_n\}$ be the *Service Portfolio*, that is, the complete set of IT services that are managed by an IT service provider. The service portfolio is a key element of ITSM and it is used to manage the entire lifecycle of each service $s_i \in S$. It includes three categories: (i) Service Pipeline P with $P \subset S$ (proposed or in development); (ii) Service Catalog C with $C \subset S$ (live or available for deployment); and (iii) Retired Services R with $R \subset S$. The service portfolio represents the current contractual commitments, the new service development, and the ongoing service improvement plans initiated as part of a *Continual Service Improvement* (CSI) process.

Definition 2. *IT service* is defined as a tuple $s_i = \{\Lambda_i, \Psi_i, \mathrm{P}_i, \mathrm{M}_i, \mathrm{A}_i\}$, where Λ_i represents the lifecycle of s_i; Ψ_i represents the set of people (customers, IT service providers, suppliers, etc.) involved in s_i; P_i represents the set of processes required to

manage s_i; M_i represents the set of metrics that help manage s_i; and A_i represents the set of applications that support s_i. Similarly as in (Ferrario & Guarino, 2009), we consider IT services to be events based on agreements and modeled by a layered set of interrelated activities (events), each one with its own participants and spatiotemporal location. Therefore, IT service providers do not deliver the IT service itself, but its content, that is, *"the actions to be performed in the interest of the customer."*

Definition 3. *Service Lifecycle* $\Lambda_i = \{\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{in}\}$ represents the different stages in which an IT service s_i can be associated.

Definition 4. *Service Stage* $\sigma_{ij} = \{\Pi_i, \Phi_{ji}^{input}, \Phi_{ji}^{output}\}$ represents each of the stages included in the Λ_i lifecycle of an IT service s_i, where $\Pi_i \subset P_i$ is the finite set of processes that support the management of s_i. Since the strength of the ITIL service management model relies on the continual feedback obtained at each service stage, Φ_{ji}^{input} represents the set of input stages that are a feedback for σ_{ij}, and Φ_{ji}^{output} represents the set of output stages that receive feedback from σ_{ij}. This feedback ensures that service optimization is managed from a business perspective.

Definition 5. *Process* $p_k = \{T_i, I_i, O_i, Y_i, \varepsilon_i\}$, with $p_k \in P_i$, represents a structured set of activities (T_i) designed to accomplish a specific objective in the management of an IT service s_i. Each process takes one or more inputs I_i and produces one or more outputs O_i. Each process may have one or more interfaces (Y_i) with other processes, and may include any number of metrics ($\varepsilon_i \subset M_i$) that help to measure its quality and effectiveness.

Definition 6. *Activity* $t_i = \{a_1, a_2, \ldots, a_n\} \in T_i$ represents the set of actions designed to achieve a particular result of a process in the management of an IT service s_i.

Definition 7. *Metric* $m_i = \{r_1, r_2, \ldots, r_n\} \in M_i$ represents a set of measurements designed to manage an IT service s_i. A metric is a scale of measurement r_i defined in terms of a standard, for example, in terms of a well-defined unit. The quantification of an event

through the process of measurement relies on the existence of explicit or implicit metrics, which are the standard to which measurements are referenced.

Definition 8. *Application* $\alpha_i \in A_i$ represents a piece of software that provides the functionality required to manage an IT service s_i. Applications implement activities and each application may support one or more IT services.

Definition 9. *Application Functions* N_i define the mapping between each activity $t_i \in T_i$ and the application $\alpha_i \in A_i$ that supports an IT service s_i.

3.3 The Onto-ITIL Ontology

In this section, we formalize the proposed ITSM model using OWL. This model relies on the *ITIL V3 Service Management Model* and on the Onto-ITIL principles formerly described in Section 3.2. It is worth highlighting that some of the Onto-ITIL concepts have been defined in terms of other existing ontologies that gather interesting domain-independent knowledge [Guarino, 1998]. This allows us to relate ITIL-based service management information to other data in the Semantic Web. Among the existing upper ontologies useful for defining some of the Onto-ITIL concepts, we have selected OpenCyc[29], the public version of the Cyc technology [Lenat, 1995] one of the most complete general knowledge base and reasoning engine available. OpenCyc provides us with the mechanisms to define the core elements of the *ITIL V3 Service Management Model* and assertions on these elements. Model elements in ITIL-based specifications are provided by separate parts of the ontology. This enables a clear separation of the different ITSM concerns and improves the understanding and reusability of Onto-ITIL concepts. From here on, we use the prefixes ‘oc’ and ‘itil’ to refer to the namespaces of OpenCyc and Onto-ITIL respectively. Figure 3.2 shows a general overview of the ITSM model defined by the Onto-ITIL ontology.

[29] http://www.opencyc.org/

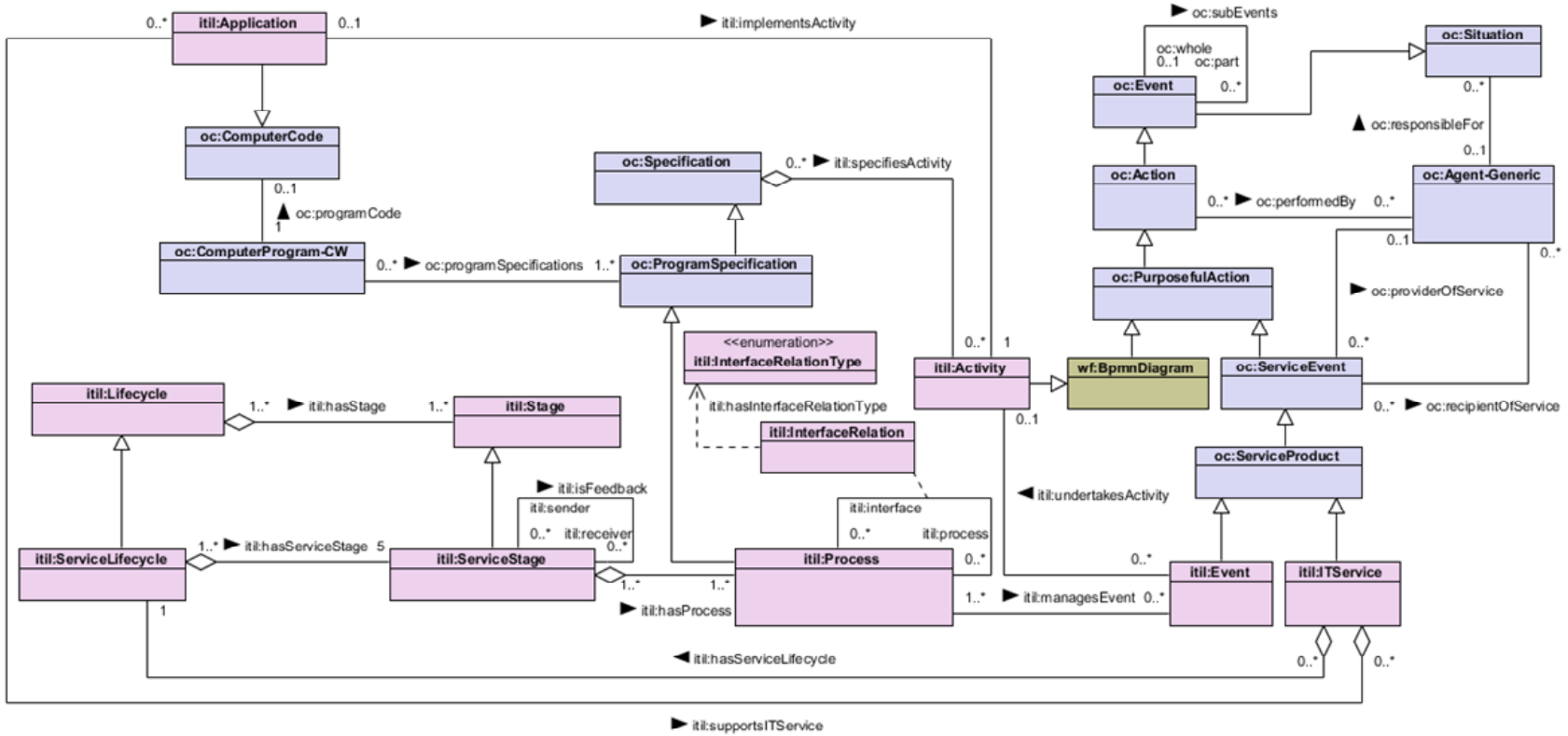


Figure 3.2 UML class diagram representing an overview of the ITSM model defined by the Onto-ITIL Ontology

3.3.1 The Service Lifecycle

An *itil:Lifecycle* represents the various stages (*itil:Stage* class) in the life of any ITSM model element (IT service, incident, problem, etc.). The *itil:Lifecycle* defines the categories for status and status transitions that are permitted using the *itil:hasStage* property. The architecture of the *ITIL V3 Service Management Model* is based on a service lifecycle (*itil:ServiceLifecycle* class, subclassing from *itil:Lifecycle*). The *itil:ServiceDesign*, *itil:ServiceTransition* and *itil:ServiceOperation* stages are progressive phases of the *itil:ServiceLifecycle* class that represent change and transformation. The *itil:ServiceStrategy* stage represents policies and objectives. Finally, the *itil:CSI* stage represents learning and improvement (see Definition 3 in Section 3.2). The stages of a service lifecycle (*itil:ServiceStage* class) are comprised of *itil:Process*(s), modeled using the *itil:hasProcess* property. As stated previously (see Definition 4 in Section 3.2), the strength of the ITSM model relies on the continual feedback obtained at each service stage [OGC, 2007d]. We use the *itil:isFeedback* and *itil:receivesFeedback* properties to express the inputs and outputs provided and required at each stage.

3.3.2 Specifications

An *oc:Specification* is the super class for all concrete specification types that constitute the underlying ITSM model. We use this class to classify the ITIL concepts that are considered specifications, such as *itil:Process* (subclassing from *oc:ProgramSpecification*). In OpenCyc, specifications are defined as "an abstract work that constitutes a description of the properties of a *oc:Situation* or a *oc:SomethingExisting*, and sometimes even entire collections of such things." In our ontology, *oc:Specification*(s) are composed of *itil:Activity*(s) that describe the specification in terms of workflows enriched with ontological knowledge (modeled using the *itil:specifiesActivity* property). The *oc:ProgramSpecification* concept is a subclass of *oc:Specification*.

The *oc:ProgramSpecification* concept represents the specification that "is not a computer program itself (i.e. lines of code), but an abstract characterization of how a program should behave. [...] A notable example of a *oc:ProgramSpecification* is UNIX - which is not (contrary to popular belief) an operating system per se, but a specification to which many different operating systems (instances of *oc:UnixOS*, subclassing from *oc:ComputerProgram-CW*) conform." Since one of our final objectives is to automate the tasks contained in ITIL processes, we consider *itil:Process* a subclass of *oc:ProgramSpecification*.

An *itil:Process* is an structured set of activities designed to accomplish a specific objective (see Definition 5 in Section 3.2). For example, in our pilot project, an *itil:ICTD_IM_Process* element (modeling the concrete incident management process designed by our IT service provider) was created as an instance of the *itil:IncidentManagement* concept. An *itil:Process* may define any number of input/output interfaces from/to other *itil:Process*(s) belonging or not to the same service management lifecycle stage. We define the next concepts in order to model the process interfaces: *itil:InterfaceRelation*, *itil:InterfaceRelationType*, *itil:hasInterfaceRelation*, *itil;hasInterfaceRelationType* and *itil:interfaceValue*.

3.3.3 Applications

The *oc:ComputerProgram-CW* concept is "a deliberately created abstract object composed of propositions that together constitute a list of instructions for computers to execute. [...] The instructions that comprise an instance of *oc:ComputerProgram-CW* can be expressed as abstract computer code (see *oc:ComputerCode*), but no list of instructions expressed in code constitutes an instance of *oc:ComputerProgram-CW*. Rather, the code in which an instance of *oc:ComputerProgram-CW* is expressed constitutes an instance of *oc:AbstractInformationStructure* that can be related to the program it expresses using the predicate *oc:programCode*." Also, the *oc:programSpecifications* property is used to relate the *oc:ComputerProgram-CW* to the *oc:ProgramSpecification* that represents the specification for how the computer program should behave (i.e., the *oc:ProgramSpecification* represents the expected behavior of the related *oc:ComputerProgram*-CW(s)).

The *oc:ComputerCode* concept is "a specialization of *oc:ComputerAIS*. Each instance of *oc:ComputerCode* is an abstract list of instructions expressed in some computer language including executable binary code." The OpenCyc concept *oc:ComputerAIS* is a specialization *of oc:AbstractInformationStructure* where each instance represents the abstract information structure of an abstract work whose instantiation in computer memory is intended to have meaning. In our approach, we consider *itil:Application* a subclass of *oc:ComputerCode*.

An *itil:Application* is a piece of software that provides the functionality required by an *itil:ITService* (see Definition 8 in Section 3.2). According to the Definition 9 in Section 3.2, each *itil:Application* implements an *itil:Activity* (modeled using the *itil:implementsActivity* property), and it may be part of one or more *itil:ITService* (modeled using the *itil:supportsITService* property). For example, in our pilot project, an *itil:HEAT_Help_Desk_Software* element was created as an instance of the *itil:Application* concept that currently implements *itil:ICTD_IM_Activity* and supports the service *itil:Access3G* (among others).

3.3.4 Events

The event concept is *"a dynamic situation in which the state of the world changes."* The *oc:subEvents* property is the most general instance of *oc:SubEventPredicate*. This predicate relates a given *oc:Event* to the *oc:Event*(s) that are its parts. The *oc:Action* concept is the subclass of *oc:Event*.

An *oc:Action* is the super class for all the concrete action types defined in Onto-ITIL. In OpenCyc, actions are defined as "the collection of *oc:Event*(s) that are carried out by some *doer*. Instances of *oc:Action* include any event in which one or more actors effect some change in the (tangible or intangible) state of the world, typically by an expenditure of effort or energy." All *oc:Action*(s) are performed by an *oc:Agent-Generic*, i.e. the actor who is responsible for (modeled using the *oc:performedBy* property). The *oc:PurposefulAction* concept is a subclass of *oc:Action*.

An *oc:PurposefulAction* (subclass of *oc:Action*) is used in our approach to classify the activities involved in an ITIL workflow process (i.e., the set of events, the order in

which they must be performed, and the actors that participate in the process) and to classify service events associated with the *ITIL V3 Service Management Model*. In the Onto-ITIL Ontology, the *wf:BpmnDiagram* and *oc:ServiceEvent* concepts are subclasses of *oc:PurposefulAction*.

An *oc:ServiceEvent* represents the super class for all concrete events. In OpenCyc, service events are defined as "events in which one or more agents (related to the event via the predicate *oc:providerOfService*) do something for one or more other agents (related to the event via the predicate *oc:recipientOfService*)." An *oc:ServiceProduct* is an *itil:ServiceEvent* done for payment. In our approach, *itil:Event* and *itil:ITService* are subclasses of *oc:ServiceProduct*.

An *itil:Event* (see Figure 3.3) is any detectable or discernible occurrence that has significance for the management of the IT infrastructure or the delivery of an IT service and evaluation of the impact a deviation might cause to the services. In our approach, *itil:Event*(s) have a lifecycle and there are three different types of *itil:Event*(s) (modeled using the *itil:EventType* enumeration class): *Informational*, *Warning* and *Exception*. We use the *itil:Event* class to specify all the events that are included in an IT service for proactive and reactive event management (modeled using the *itil:ManagedEventType* enumeration class). According to ITIL, some events could be part of different processes, or even a combination of two or more of them. Therefore an *itil:ITServiceProvider* must decide and indicate what *itil:Process* (or processes) is going to manage a specific *itil:Event* (modeled using the *itil:managesEvent* property). Also, activities undertaken to manage a specific *itil:Event* are included using the *itil:undertakesActivity* property. In our proposal, *itil:Incident*, *itil:ServiceRequest*, *itil:RFC*, *itil:Change* and *itil:Problem* are the subclasses of *itil:Event*. In our pilot project, each *itil:Event* has a type of intervention depending if they are managed by an agent or not (modeled using the *itil:TechnicalManagementType* class); and there are four types of events depending on the business area where the event must be resolved (modeled using the *itil:EventCategoryCode* class): (i) Teaching; (ii) Systems and users; (iii) Development; and (iv) Communications. Also, activities undertaken to manage a specific *itil:Event* are included using the *itil:undertakesActivity* property. For example, an instance of *itil:Incident* (subclass of *itil:Event*), *itil:AppServerFailure*, defines the characteristics of this kind of managed event in the organization, the actions to be performed in order to

resolve it (modeled using the *itil:AppServerFailure_Activity* instance and the *itil:undertakesActivity* property), and since it is an incident, the incident is related to the *itil:ICTD_IM_Process* instance.

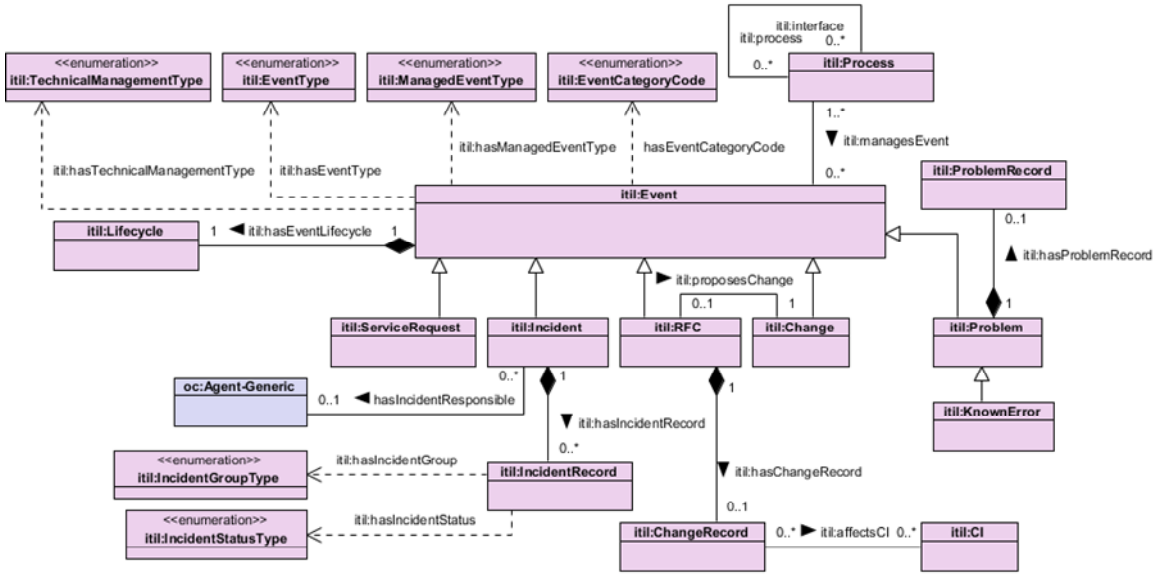


Figure 3.3 UML class diagram representing the Onto-ITIL event knowledge

An *itil:Incident* is an unplanned interruption to an *itil:ITService* or reduction in the quality of an *itil:ITService* that must be managed by the corresponding *itil:ITServiceProvider*. Each *itil:Incident* may be associated with one or more *itil:IncidentRecord*(s). The *itil:IncidentRecord* is the class that contains the details of each occurrence of a specific *itil:Incident* and they are related through the *itil:hasIncidentRecord* property. Each *itil:Incident* may have links to the *itil:Event*(s) concerned (*oc:subEvents* property) (for example, relationship with other *itil:Incident*(s), *itil:Problem*(s), *itil:Change*(s) or *itil:KnownError*(s)), and to the *itil:Activity* undertaken to resolve the *itil:Incident* (modeled using the *itil:undertakesActivity* property). Also, in our pilot project, an *itil:Incident* is allocated to different support groups/persons that could resolve the *itil:Incident* (*oc:performedBy* property). In this project, each *itil:IncidentRecord* includes the responsible (IT service provider side) of the occurrence of the *itil:Incident* (in this case, the person or group recording the incident), the status of a specific *itil:Incident* and the user or group (customer side) that reported the occurrence of the *itil:Incident*.

An *itil:ServiceRequest* is a request from an *itil:User* for information or advice, or for a standard change or for access to an *itil:ITService*. For example to reset a password, or to provide standard *itil:ITService*(s) for a new *itil:User*. To be an *itil:ServiceRequest*, it is normal for some prerequisites to be defined and met (e.g., needs to be proven, repeatable, pre-approved, proceduralized). The *itil:ServiceRequest*(s) do not require an *itil:RFC* to be submitted. In our pilot project, each *itil:ServiceRequest* is allocated to different support groups/persons that could deal with the *itil:ServiceRequest* (modeled using the *oc:performedBy* property).

A *Request for Change* (RFC) is a formal proposal for a change to be made. An *itil:RFC* includes details of the proposed *itil:Change* (related through the *itil:proposesChange* property), and may be recorded on paper or electronically. Authorized *itil:RFC*(s) should be passed to the relevant technical groups for building of the changes. The details of a change are included in *itil:ChangeRecord* using the *itil:hasChangeRecord* property. The *itil:ChangeRecord*(s) reference the *itil:CI*(s) that are affected by the requested change (modeled using the *itil:affectsCI* property).

An *itil:Change* represents the addition, modification or removal of authorized, planned or supported service or service component and its associated documentation. In our pilot project, changes are considered urgent when they must be introduced as soon as possible in order to restore a service after the identification of a problem and to minimize the impact on the business; and changes are considered pre-approved when the change represents a standard change where the intervention of the *Change Advisory Board* (CAB) is not required.

An *itil:Problem* is the cause of one or more incidents. In our pilot project, each *itil:Problem* is allocated to an specific support group/person that could resolve the *itil:Problem* (modeled using the *oc:performedBy* property). The *itil:Problem*(s) are detailed in *itil:ProblemRecord*(s) using the *itil:hasProblemRecord* property. In our ontology, *itil:KnownError* is the subclass of *itil:Problem*.

An *itil:KnownError* is an *itil:Problem* that has a documented root cause and a workaround. The workaround describes how to reduce or eliminate the impact of an *itil:Problem* for which a full resolution is not yet available. For example, by restarting a failed *itil:CI*. An *itil:CI* is an asset, service component or other item that is, or will be,

under the control of *itil:ServiceAsset_and_ConfigurationManagement* process. The details of an *itil:CI* are included in *itil:ConfigurationRecord* using the *itil:hasConfigurationRecord* property.

IT Services

An *itil:ITService* (see Definition 2 in Section 3.2) is an *oc:ServiceProduct* provided to one or more customers by an IT service provider as shown in Figure 3.4. That is, *itil:ITService*(s) represent the means of delivering value to customers by facilitating outcomes, and since they are based on agreements, they have to be defined in a SLA. The *itil:CoreService* and *itil:SupportingService* concepts are the subclasses of *itil:ITService*.

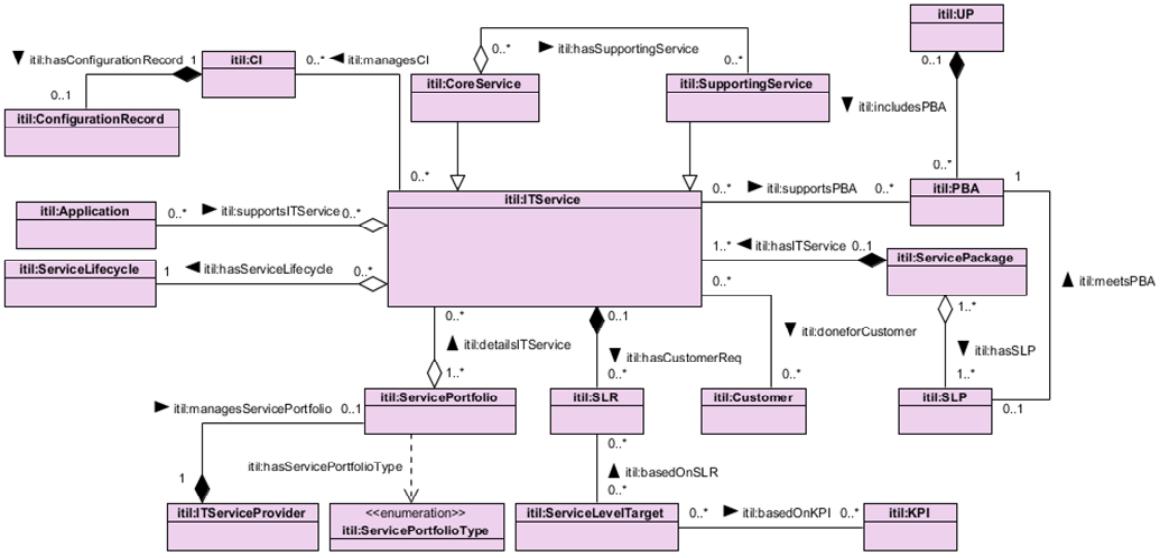


Figure 3.4 UML class diagram representing the Onto-ITIL IT service knowledge

An *itil:CoreService* represents an *itil:ITService* that delivers the basic outcomes desired by the *itil:Customer*. The *itil:CoreService*(s) represent the value that the *itil:Customer* wants and for which they are willing to pay. The *itil:CoreService*(s) anchor the value proposition for the *itil:Customer* and provide the basis for their continued utilization and satisfaction. For example, in our pilot project, *itil:Access3G*, *itil:DNS_Service*, *itil:Staff_email*, *itil:HW_Management* and *itil:Software_Licensing* are examples of instances of *itil:CoreService*. An *itil:SupportingService* is an *itil:ITService* that enables

or enhances an *itil:CoreService*. For example, *itil:Backup* and *itil:Mailing_Lists* instances. These two classes (*itil:CoreService* and *itil:SupportingService*) are related using the *itil:hasSupportingService* property.

Each *itil:ITService* defines a set of *itil:Metric*(s) whose purpose is to measure the quality and effectiveness of that service in order to take timely actions that make sure service are delivered in line with business needs. These are the metrics that really matter in order to demonstrate the value of the service and for the operation in a cycle of continuous improvement. Also, *itil:ITService*(s) are managed according to an *itil:ServiceLifecycle* and they are composed of *itil:Application*(s) and other *itil:CI*(s) necessary to support the provision of the *itil:ITService* to the business.

On the other hand, an *itil:ITService* is based on the use of information technology and supports the customer's business processes (in fact, many business processes rely on IT services). A *pattern of business activity* (PBA) defines dynamics of a business and includes interactions with customers, suppliers, partners and other stakeholders in an *itil:ITService* (modeled using the *itil:supportsPBA* property). An *itil:PBA* represents a workload profile of one or more business activities, where workload is the resources required to deliver an identifiable part of an *itil:ITService*.

A *user profile* (UP) is a pattern of user demand for *itil:ITService*(s). The *itil:UP*(s) are constructed using one or more predefined *itil:PBA*(s) (modeled using the *itil:includesPBA property*). Pattern matching using *itil:PBA* and *itil:UP* ensure a systematic approach to understanding and managing demand from customers.

As customers and suppliers become the direct users of IT services, the expectations and *service level requirements* (SLRs) have become more demanding, requiring a value net approach. An *itil:SLR* is a customer requirement for an aspect of an *itil:ITService*. A set of targets and responsibilities should be documented and agreed within an *itil:SLR* for each proposed new or changed *itil:ITService*. An *itil:SLR* is based on business objectives and it is used to negotiate agreed *itil:ServiceLevelTarget*(s) (modeled using the *itil:usedForNegotiation* property).

An *itil:ServiceLevelTarget* is a commitment that is documented in an *itil:SLA*. The *itil:ServiceLevelTarget*(s) are based on *itil:SLR*(s) (modeled using the *itil:basedOnSLR* property), and they are needed to ensure that the *itil:ServiceDesign* is fit for purpose

(i.e., it meets customer expectations). The *itil:ServiceLevelTarget*(s) should be smart, and are usually based on *itil:KPI*(s) (modeled using the *itil:basedOnKPI* property). For example, in our pilot project, *itil:SLT_IncidentResolution* is an instance of *itil:ServiceLevelTarget* based on *itil:SLR_Incident_and_Problem_Management* (instance of *itil:SLR*) and it is also based on the KPI *itil:Average_Incident_Resolution_Hours* (instance of *itil:KPI*).

Service Portfolios

The *itil:ServicePortfolio* (see Definition 1 in Section 3.2) is the complete set of *itil:ITService*(s) (modeled using the *itil:detailsITService* property) that are managed by an IT service provider. The *itil:ServicePortfolio* is used to manage the entire lifecycle of all *itil:ITService*(s), and includes three categories (modeled using the *itil:ServicePortfolioType* enumeration class): *itil:SERVICE_PIPELINE*, *itil:SERVICE_CATALOG* and *itil:RETIRED_SERVICES*. For example, in our pilot project, *itil:ICTD_ServiceCatalog* is an instance of *itil:ServicePortfolio*, where the *itil:hasServicePortfolioType* property is equal to *itil_SERVICE_CATALOG* and is related to the different instances of *itil:ITservice* using the *itil:detailsITService* property.

Service Packages

An *itil:ServicePackage* is detailed description of an *itil:ITService* that is available to be delivered to *itil:Customer*(s) (modeled using the *itil:hasITService* property). The *itil:ServicePackage*(s) come with one or more *itil:SLP*(s) (modeled using the *itil:hasSLP* property). An *itil:ServicePackage* is considered a core *itil:ServicePackage* (modeled using the *itil:corePackage* datatype property) when it represents a detailed description of an *itil:CoreService* that may be shared by two or more *itil:ServiceLevelPackage*(s).

An *itil:SLP* is a defined level of utility and warranty for a particular *itil:ServicePackage*. Each *itil:SLP* is designed to meet the needs of a particular *itil:PBA* (modeled using the *itil:meetsPBA* property).

3.3.5 Roles

To represent role knowledge (see Figure 3.5), we use the *oc:IntelligentAgent* class (subclassing from *oc:Agent-Generic*). In OpenCyc, *oc:IntelligentAgent* is defined as "an agent that is capable of knowing and acting, and capable of employing its knowledge in its actions. An *oc:IntelligentAgent* typically knows about certain things, and its beliefs concerning those things influences its actions. As with agents generally, an *oc:IntelligentAgent* might either be a single individual, such as a person, or a group consisting of two or more individual agents, such as a business or government organization." The *oc:Organization* concept is a subclass of *oc:IntelligentAgent*.

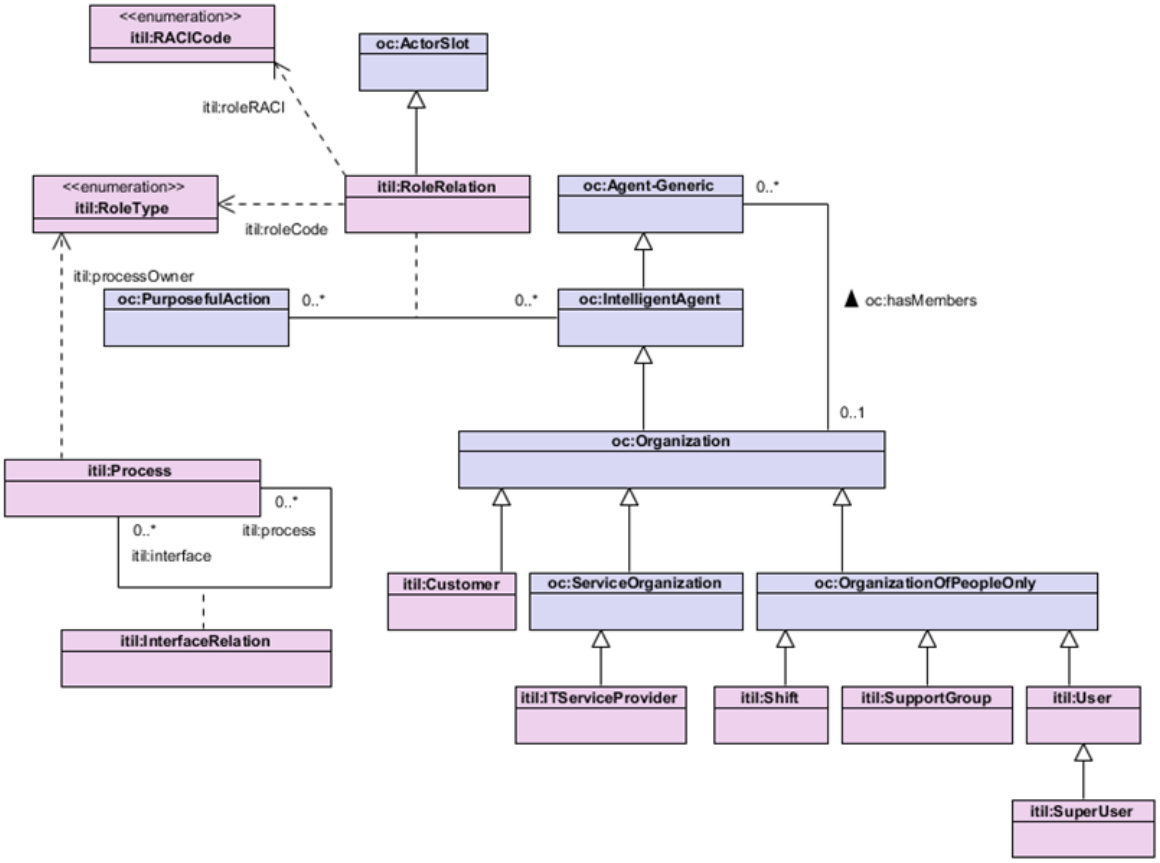


Figure 3.5 UML class diagram representing the Onto-ITIL role knowledge

The *oc:Organization* concept is defined as "the collection of all organizations. Each instance of *oc:Organization* is a group whose group-members are instances of *oc:IntelligentAgent*." We use the *oc:hasMembers* property to relate a particular organization to the agents who are members of that organization. The ontology

concepts of *itil:Customer*, *oc:ServiceOrganization* and *oc:OrganizationOfPeopleOnly* are the subclasses of *oc:Organization*.

An *itil:Customer* is someone who buys goods or services. The *itil:Customer* of an *itil:ITServiceProvider* is the person or group who defines and agrees the *itil:ServiceLevelTarget*(s) in an *itil:SLA*.

An *oc:ServiceOrganization* is "an organization whose main function is to provide some service or services". In our approach, the *itil:ITServiceProvider* concept is the subclass of *oc:ServiceOrganization*. An *itil:ITServiceProvider* is a service that provides *itil:ITService*(s) to internal or external *itil:Customer*(s) (*itil:internalProvider* datatype property).

The *oc:OrganizationOfPeopleOnly* concept is defined as "an organization each of whose members is a person." In our approach, the *itil:Shift*, *itil:SupportGroup* and *itil:User* concepts are examples of subclasses of *oc:OrganizationOfPeopleOnly*.

An *itil:Shift* is a group or team of people who carry out a specific role for a fixed period of time. An *itil:SupportGroup* is a group of people with technical skills. The *itil:SupportGroup*(s) provide the technical support needed by all of the ITSM processes (*itil:Process*). An *itil:User* is a person who uses the IT service on a day-to-day basis. The *itil:User* class is distinct from the *itil:Customer* class, as some *itil:Customer*(s) do not use the IT service directly. An *itil:SuperUser* is an *itil:User* who helps other users, and assists in communication with the *itil:SERVICE_DESK* (instance of *itil:RoleType*) or other parts of the *itil:ITServiceProvider*. The *itil:SuperUser*(s) typically provide support for minor *itil:Incident*(s) and training.

Each *oc:IntelligentAgent* may have several roles (modeled using the *itil:RoleRelation* class). For example, the roles of *itil:INCIDENT_MANAGER* and *itil:PROBLEM_MANAGER* may be carried out by a single agent. The *itil:RoleRelation* class (subclassing from *oc:ActorSlot*) is used to build a RACI chart that is needed to identify/define, on the one hand, the functional roles (modeled using the *itil:RoleType* enumeration class) and, on the other hand, responsibilities of the various roles (modeled using the *itil:RACICode* enumeration class). A *role* represents a set of responsibilities granted to a person or team that takes part in an *oc:PurposefulAction* (modeled using the *itil:RoleType* enumeration class and *itil:roleAction* and *itil:roleCode* properties).

One role may have multiple responsibilities, which are defined according to the RACI matrix in ITIL V3 using the *itil:roleRACI* property and the *itil:RACICode* enumeration class. RACI stands for *Responsible*, *Accountable*, *Consulted* and *Informed*: (i) *Responsible*: the individual who is responsible to perform the actions; (ii) *Accountable*: the individual who is ultimately accountable has the power of veto. Only one accountable can be assigned to an action; (iii) *Consulted*: the individual(s) to be consulted prior to a final decision or action being taken; and (iv) *Informed*: the individual(s) who needs to be informed after a decision or action is taken. The owner of an *itil:Process*, and specific *roles* and *responsibilities* are defined for each *oc:IntelligentAgent* in an *oc:PurposefulAction* using the *itil:hasRoleRelation* property.

3.3.6 The ITSM Metrics Model

The *itil:Process*(s) are measured in terms of *itil:Metric*(s) (see Figure 3.6). In our approach, we include a complete metrics model suggested in [Steinberg, 2006] that can be used with the *ITIL V3 Service Management Model*. In general, an *itil:Metric* (see Definition 7 in Section 3.2) is a scale of *itil:Measurement* defined in terms of a standard, i.e. in terms of a well-defined unit, using the *itil:includesMeasurement* property. Each *itil:Metric* has a type (modeled using the *itil:MetricType* enumeration class) and they must be designed in line with customer (business) requirements for ITSM. The Onto-ITIL concepts of *itil:OperationalMetric*, *itil:KPI*, *itil:Tolerance*, *itil:CSF*, *itil:Dashboard*, *itil:Outcome* and *itil:AnalyticalMetric* are the subclasses of *itil:Metric*, where according to Steinberg [2006], *itil:KPI* and the related *itil:Tolerance*, *itil:CSF*, *itil:Dashboard* and *itil:Outcome* are really the *"metrics that matter."* That is, as mentioned earlier, the metrics that provide a basis for making business decisions in the delivery of the *itil:ITService*.

An *itil:OperationalMetric* is a basic observation of operational events that provides live data from ITSM process (i.e., *itil:Process*) reporting and other infrastructure measurements and observations. For example, in our pilot project, *itil:Percentage_of_incidents_handled_within_agreed_response_time* and *itil:Total_number_of_incidents* are examples of instances of *itil:OperationalMetric* that help determine the efficiency and effectiveness of the *itil:ICTD_IM_Process* instance.

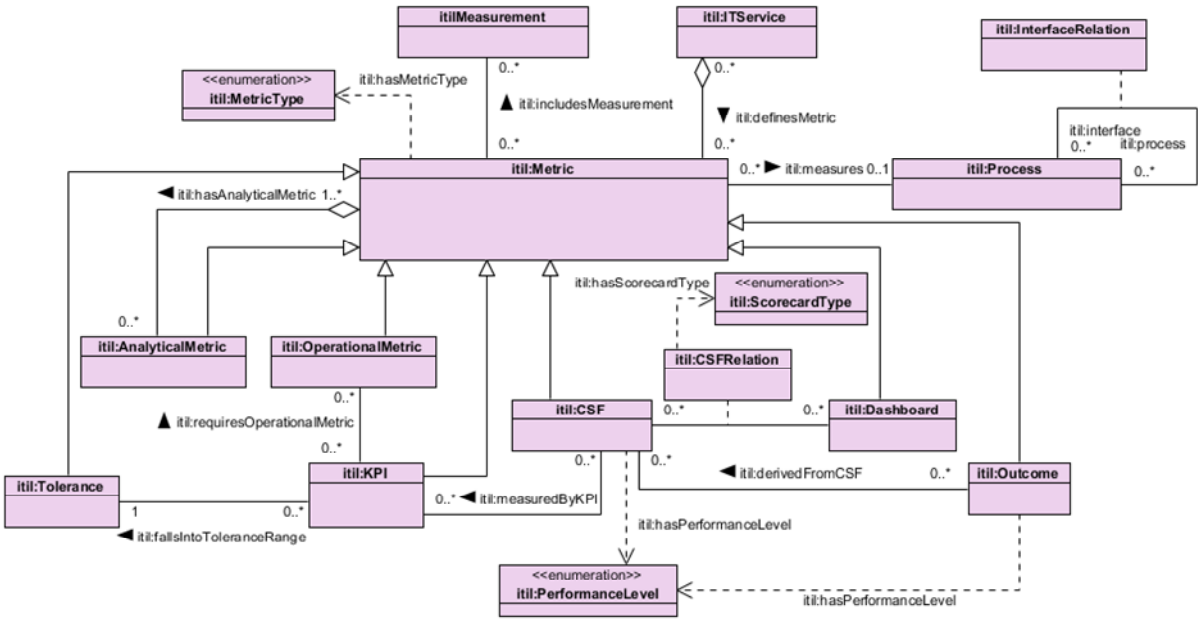


Figure 3.6 UML class diagram representing the Onto-ITIL metrics knowledge

An *itil:Metric* is considered as an *itil:KPI* when it measures the success with the *itil:SLA*(s) defined with an *itil:Customer*. That is, only the *itil:Metric*(s) that provide a basis for making business decisions are defined as *itil:KPI*(s) and they are used to actively manage and report on the *itil:Process*. Each *itil:KPI* is trying to answer a question. While *itil:OperationalMetric*(s) are generally historical in nature, *itil:KPI*(s) are really the *"metrics that matter."* These *itil:KPI*(s) become the data inputs to analyze and identify improvement opportunities. For example, in our pilot project, *itil:Incident_resolution_rate* and *itil:Customer_satisfaction_level* are instances of *itil:KPI* for the *itil:ICTD_IM_Process* instance. In our approach, according to Steinberg [2006], the *itil:KPI*(s) are calculated or derived from one or more *itil:OperationalMetric*(s). For example, in our pilot project, the *itil:KPI* of *itil:Incident_resolution_rate* is the result of dividing *itil:Number_of_incidents_resolved_within_agreed_service_levels* by *itil:Total_number_of_incidents* (instances of *itil:OperationalMetric*). The results of these calculations are then compared to an *itil:Tolerance* range to identify whether those results fall within acceptable levels.

In order to get decisions, we need another type of metric that indicates when to take actions. The *itil:Tolerance* is an indicator that identifies, in advance, the boundary in which the IT service provider expects a KPI to operate and behave. That is, the *itil:Tolerance* class represents the boundaries for acceptable and non-acceptable *itil:KPI* values (i.e., service target and warning level). For example, in our pilot project, if the service target of the *itil:Tolerance* boundary for the *itil:KPI* of *itil:Average_Incident_Resolution_Hours* is 2.0 it means that the service target for this *itil:KPI* would be 2.0 hours. On the other hand, if the warning level of the *itil:Tolerance* boundary for the *itil:KPI* of *itil:Average_Incident_Resolution_Hours* is 3.5, it means that the performance of this *itil:KPI* would be considered acceptable as long as it is not higher than 3.5 hours. If it is higher, management actions may need to take place to raise the performance back to acceptable levels.

A *Critical Success Factor* (CSF) is something that must happen if an *itil:Process* is to succeed. The *itil:KPI*(s) are used to measure the achievement of each *itil:CSF*. For example, in our pilot project, *itil:Quickly_resolve_incidents* is a instance of *itil:CSF* measured by the *itil:KPI*(s) of *itil:Incident_reopen_rate*, *itil:Average_time_to_resolve_severity1_and_severity2_incidents_hours* and *itil:Incident_management_tooling_support_level*. In another example, the *itil:KPI* of *itil:KPI_10_percent_increase_in_customer_satisfaction_rating_for_handling_incidents_over_the_next_6_months* would measure an *itil:CSF* of *itil:Improving_IT_service_quality*, and the *itil:KPI* of *itil:KPI_10_percent_reduction_in_the_costs_of_handling_printer_incidents* would measure an *itil:CSF* of *itil:Reducing_IT_costs*. Also, an *itil:CSF* can be associated with an performance indicator (modeled using the *itil:PerformanceLevel* enumeration class).

In an *itil:CSF*, to receive the performance level of 'High', all the associated *itil:KPI*(s) must have met or exceeded their *itil:Tolerance* acceptable values. When one of the associated *itil:KPI*(s) falls into an *itil:Tolerance* non-acceptable value, the *itil:CSF* performance level might be 'Medium' or 'Low' depending on how the associated *itil:KPI* value fell within the specified *itil:Tolerance* range for it.

An *itil:Dashboard* is a graphical representation of overall IT service performance and availability. The *itil:Dashboard* images may be updated in real-time, and can also be

included in management reports and web pages. Therefore, *itil:Dashboard*(s) can be considered as key *itil:Metric*(s) that are represented on a report or graphical interface that indicates the success, at risk or failure of a business activity. The *itil:Dashboard* results are derived from *itil:CSF* results (modeled using the *itil:CSFRelation* class). The *itil:CSF*(s) can contribute to one or more *itil:Dashboard*(s) and each *itil:Dashboard* may have one or more multiple *itil:CSF*(s). For the purpose of our approach, just like the approach of Steinberg [2006], we use the *Balanced Scorecard* originally developed by Kaplan and Norton [1992]. The *Balanced Scorecard* was originally developed around the concept that financial measures alone are not critical for business success. The *Balanced Scorecard* has been generally recognized as an acceptable approach for senior management levels where the scorecard categories recommended for ITSM are (modeled using the *itil:ScorecardType* enumeration class): Customer, Capabilities, Operational, Financial and Regulatory.

The *itil:Outcome*(s) are key indicators of general business risk areas, that is, they are the kind of things that IT is trying to protect against. These are associated with performance indicators that identify the success, at risk or failure of *itil:KPI*(s) or *itil:CSF*(s). The *itil:CSF*(s) are used to determine *itil:Outcome*(s) (operational risks). Legal exposure, service outages, rework, waste, security breaches, unexpected costs, slow response to business needs and changes, fines and penalties, loss of market share and dissatisfied customers are examples of *itil:Outcome*(s). The *itil:Outcome*(s) can be associated with a performance indicator: High, Medium or Low (modeled using the *itil:hasPerformanceLevel* property) that might reflect the likelihood of risk that the *itil:Outcome* will occur. In Onto-ITIL, the risk level is derived from the mean average of the *itil:CSF* performance levels. Scoring for an *itil:Outcome* runs opposite to how the *itil:CSF*(s) are calculated. If an *itil:CSF* scores 'Low', meaning the likelihood of achieving that *itil:CSF* is low, then the *itil:Outcome* would score 'High'. This means that the risk of the *itil:Outcome* occurring is high because the *itil:CSF* achievement was low.

An *itil:AnalyticalMetric* is used to separate out certain *itil:Metric*(s) that are really more helpful for supporting research into an issue, incident or service problem. The *itil:AnalyticalMetric*(s) are metrics that IT service providers may report on only on a one-time basis or as part of a drill-down (such as for an *itil:Dashboard*). An *itil:AnalyticalMetric* is a subset of subdivision of an *itil:Metric* (modeled

us*hasAnalyticalMetric* property). For example, in our pilot project, the *itil:OperationalMetric* of *itil:Total_number_of_incidents_for_analytical_purposes* has been broken out by the next *itil:AnalyticalMetric*(s): *itil:Department_of_business_unit*, *itil:Physical_Intervention*, *itil:Expert*, *itil:IT_service_delivered* and *itil:Time_of_day*.

3.3.7 Service Level Agreements

For *Service Level Agreement* (SLA) management (see Figure 3.7), we have included the *oc:Contract* concept. In OpenCyc, a contract is defined as "a legal agreement in which two or more *oc:agreeingAgents* promise to do (or not do) something. There are legal consequences to breaking the promises made in a *oc:Contract*." An *oc:Contract* is composed of one or more *oc:ContractDocument* (modeled using the *itil:agreesContractDocument* property).

The *itil:SLA* represents the *itil:Agreement* (subclass of *oc:ContractDocument*) that describes a formal understanding of an agreement between *itil:Customer*(s) and the *itil:ITServiceProvider*. That is, an *itil:SLA* is a written agreement between an *itil:ITServiceProvider* and the *itil:Customer*(s), defining the key service targets and responsibilities of both parties. Each *itil:Agreement* defines a business process that enables the delivery of an *itil:ITService* (modeled using the *itil:definesBusinessProcess* property). An *itil:SLA* describes the *itil:ITService*, *itil:ServiceLevelTarget*(s), and specifies the responsibilities of the *itil:ITServiceProvider* (modeled using the *itil:ITServiceProviderRelation* class and the *itil:hasITServiceProviderRelation* property) and the *itil:Customer* (modeled using the *itil:CustomerRelation* class and the *itil:hasCustomerRelation* property). An *itil:SLA* represents the level of assurance or warranty with regard to the level of service quality delivered by the *itil:ITServiceProvider* to the *itil:Customer*(s) for each of the *itil:ITService*(s) delivered to the business. Also, *itil:SLA*(s) are related to the contracts *Operational Level Agreements* (OLAs) and *Underpinning Contracts* (UCs) which provide support to SLA fulfillment (modeled using *itil:OLA* and *itil:UC* classes, and *itil:supportedByOLA* and *itil:supportedByUC* properties). The *itil:OLA* is an agreement between an *itil:ITServiceProvider* and a third party that assists with the provision of *itil:ITService*(s) to *itil:Customer*(s). However, in this case, the third party is another part of the same

itil:Organization. The *itil:OLA* defines the goods or services to be provided and the responsibilities of both parties. For example there could be an *itil:OLA* between the *itil:ITServiceProvider* and a procurement department to obtain hardware in agreed times. Finally, the *itil:UC* is an *itil:Agreement* between an *itil:ITServiceProvider* and a third party. In this case, the third party (supplier) is another *itil:Organization*. The *itil:UC* defines targets and responsibilities that are required to meet agreed *itil:ServiceLevelTarget*(s) in an *itil:SLA*.

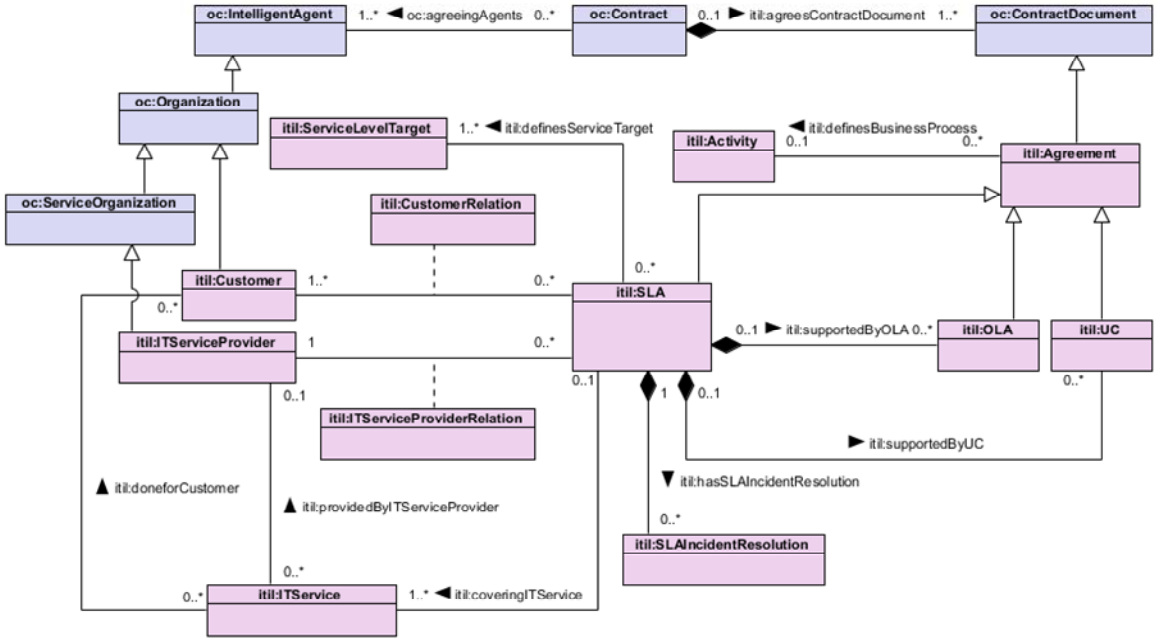


Figure 3.7 UML class diagram representing the Onto-ITIL SLA knowledge

Since suppliers (internal or external) and the management of suppliers and partners are essential to the provision of quality IT services [OGC, 2007a], we can obtain the internal and cross-organizational integration of the supporting services through the management of *itil:OLA*(s) and *itil:UC*(s) using ebXML business process specifications.

The *itil:OLA* and *itil:UC* concepts represent the *Collaboration Protocol Agreements* (CPAs) established between the business parties in the ebXML domain. This means that both parties do electronic business directly according to a specific CPA (i.e., the IT service provider and its supplier follow the business process defined in the CPA). For example, in our pilot project, a new computer tool for incident management was required in order to implement *itil:ICTD_IM_Process*. Therefore, the *itil:ICTD_IM_Activity* business process, instance of *itil:Activity*, that specifies the

corresponding process flow needs to be transformed into a ebXML model and associated with the CPA document (i.e., *itil:ICTD_IM_OLA*, instance of *itil:OLA*, that has been transformed into the ebXML CPA document).

Table 3.1 Mapping between ebXML constructs and Onto-ITIL constructs

ebXML construct	**Onto-ITIL construct**
ebxml:MultipartyCollaboration	wf:Pool, wf:Lane and itil:RoleType
ebxml:BusinessPartnerRole	itil:RoleRelation
ebxml:Performs	oc:performedBy
ebxml:AuthorizedRole	oc:IntelligentAgent, oc:responsibleFor and itil:RoleRelation
ebxml:BinaryCollaboration	itil:Activity
ebxml:BusinessTransactionActivity	itil:Activity and wf:ActivityType="Subprocess" OR "Task"
ebxml:CollaborationActivity	itil:Activity and wf:ActivityType="Subprocess" OR "Task"
ebxml:BusinessTransaction	itil:Activity and wf:ActivityType="Subprocess" OR "Task"
ebxml:RequestingBusinessActivity	itil:Activity and wf:ActivityType= "Task"
ebxml:RespondingBusinessActivity	itil:Activity and wf:ActivityType= "Task"
ebxml:DocumentEnvelope	itil:Agreement
ebxml:BusinessDocument	itil:Agreement
ebxml:Transition	wf:Assocation and wf:SequenceEdge
ebxml:Start	itil:Activity and wf:ActivityType= "EventStartEmpty" OR "EventStartMessage" OR "EventStartMultiple" OR "EventStartRule" OR "EventStartTimer" OR "EventStartLink" OR "EventStartSignal"
ebxml:Sucess	itil:Activity and wf:ActivityType= "EventEndEmpty" OR "EventEndMessage" OR "EventEndCompensation" OR "EventEndTerminate" OR "EventEndLink" OR "EventEndMultiple"
ebxml:Failure	itil:Activity and wf:ActivityType= "EventEndError"
ebxml:Fork	itil:Activity and wf:ActivityType= "GatewayDataBasedExclusive" OR "GatewayEventBasedExclusive" OR "GatewayDataBasedInclusive" OR "GatewayParallel" OR "GatewayComplex"
ebxml:Join	itil:Activity and wf:ActivityType= "GatewayDataBasedExclusive" OR "GatewayEventBasedExclusive" OR "GatewayDataBasedInclusive" OR "GatewayParallel" OR "GatewayComplex"

The resulting mapping between the ebXML business process specification constructs [UN/CEFACT and OASIS, 2001] and Onto-ITIL constructs for supplier management is summarized in Table 3.1 (ebXML abstract classes and optional classes have been

omitted). In order to validate our approach, we implemented a prototype in Java in the Eclipse platform that generates the transformation from an Onto-ITIL model to an ebXML model. Some ebXML constructs are derived from the combination of some constructs in the Onto-ITIL model, as shown in Table 3.1.

3.3.8 The Onto-BPMN Ontology

As mentioned earlier, *oc:Specification*(s) may have associated the process flow or workflow (*itil:Activity*) which defines how a specification achieves its purpose. To complete the semantics of workflows, we have developed the Onto-BPMN Ontology as part of Onto-ITIL Ontology (see *wf:BpmnDiagram* class in Figure 3.2). The Onto-BPMN Ontology is a formalization in OWL of the BPMN constructs [OMG, 2006a], that is shown in Figures 3.8 and 3.9.

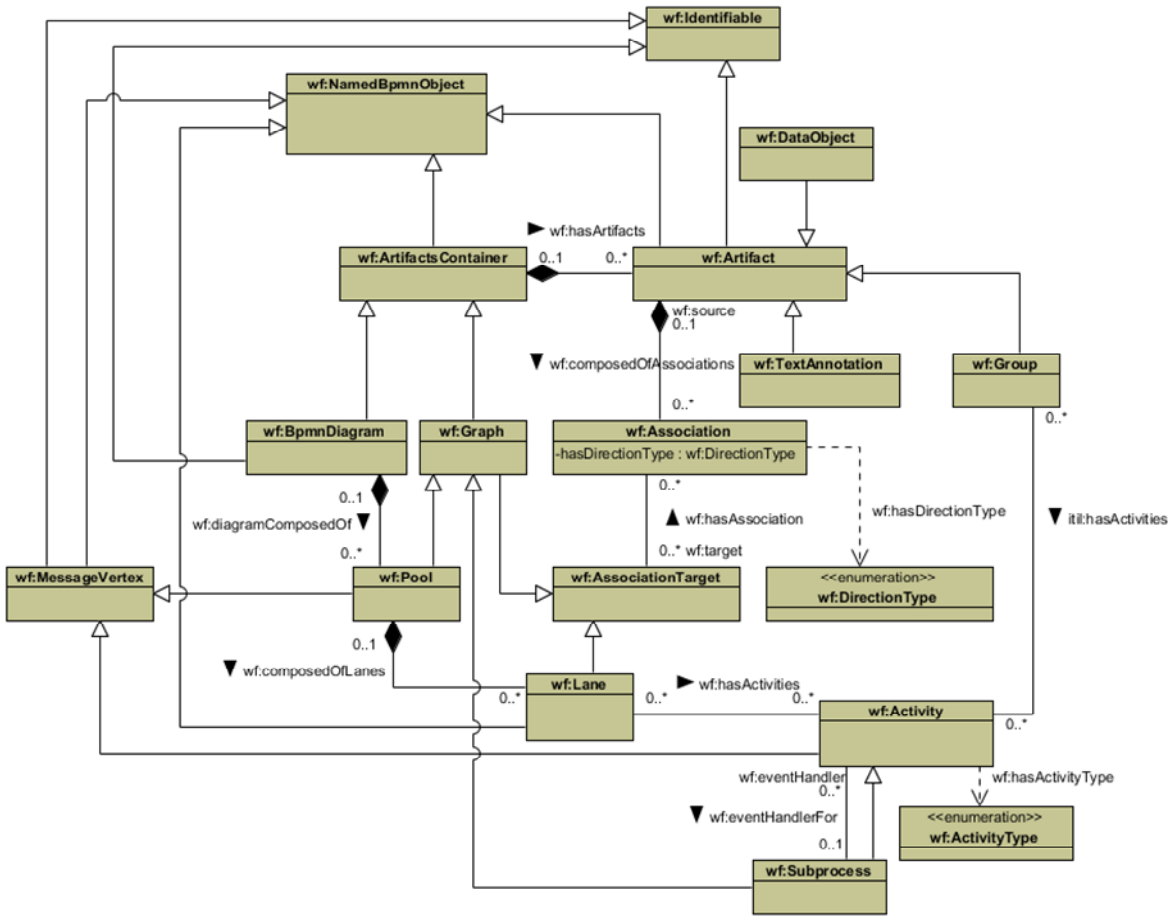


Figure 3.8 UML class diagram representing the Onto-BPMN Ontology

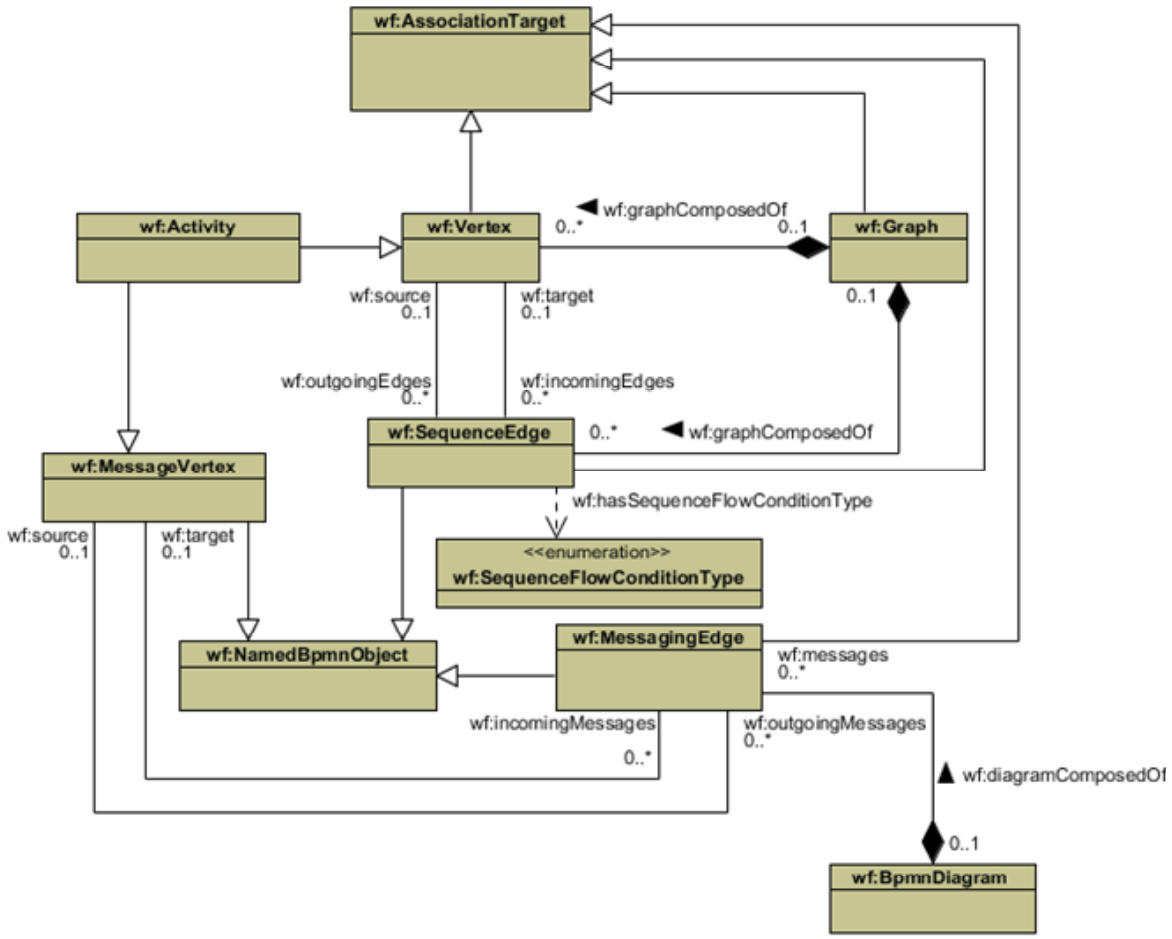


Figure 3.9 UML class diagram representing the Onto-BPMN Ontology (cont.)

In this case, the definition of our ontology was driven by the description of the complete set of BPMN elements contained in the metamodel of the BPMN modeler subproject developed for the *SOA Tools Platform* (STP) project[30], enabling the integration of our workflow specifications into the Eclipse platform. The BPMN metamodel is depicted in Figure 3.10. The BPMN modeler is based on the *Eclipse Modeling Framework Project* (EMF[31]) object model bound to a graphical notation via the *Graphical Modeling Framework* (GMF[32]). This ontology is kept separate for a better management of the workflow knowledge of an ITSM model. In this case, we use the prefix 'wf' to reference the namespace of our Onto-BPMN Ontology.

[30] http://www.eclipse.org/bpmn/
[31] http://www.eclipse.org/modeling/emf/
[32] Graphical Modeling Project (GMP): http://www.eclipse.org/modeling/gmp/

The *wf:BpmnDiagram* concept (subclassing from *wf:ArtifactsContainer*) is therefore used for the *workflow dimension* of our ontology. The *wf:BpmnDiagram* (subclassing from *oc:PurposefulAction* in the Onto-ITIL Ontology) is the workflow representation (i.e., the workflow model) in form of a BPMN diagram which is composed of pools (*wf:Pool*) and messages (*wf:MessagingEdge*). In our approach, we consider *itil:Activity* a subclass of *wf:BpmnDiagram* in order to model the high level requirements of the information system that could automate the activities defined as part of a workflow model associated with an ITSMS.

A complete specification of a BPMN diagram definition in the Onto-BPMN Ontology consists of the next model elements: *Artifacts* (*Data object*, *Group* and *Text annotation*), *Graphs* (*Pool* and *Subprocess*), *Lanes*, *Nodes* (*Activity*) and *Edges* (*Sequence edge* and *Messaging edge*).

A *wf:DataObject* is an *wf:Artifact* that provides provide information about what the what activities require to be performed and/or what they produce. That is, how documents, data, and other objects are used and updated during the business process. A *wf:DataObject* can represent a singular object or a collection of objects.

A *wf:Group* is an *wf:Artifact* that provides a visual mechanism to group elements of a diagram informally.

A *wf:TextAnnotation* is an *wf:Artifact* that provides a mechanism to introduce additional text information for the reader of a BPMN Diagram.

A *wf:Graph* is the workflow model graphical element used to define pools (*wf:Pool*) and subprocesses (*wf:SubProcess*). A *wf:Graph* is composed of vertices (*wf:Vertex*) and edges (*wf:SequenceEdge*).

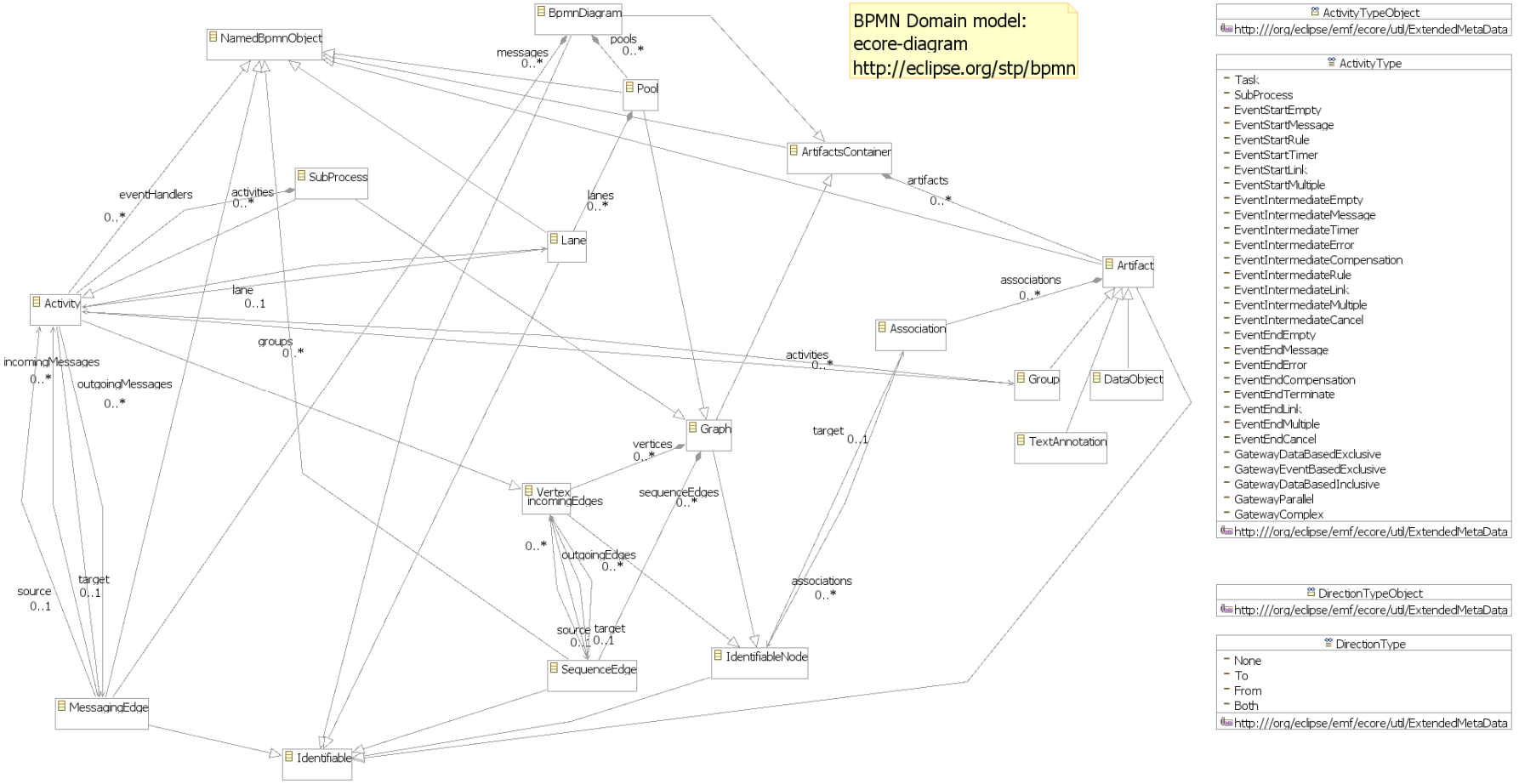


Figure 3.10 UML class diagram representing the BPMN Metamodel as depicted in [Eclipse - BPMN Modeler, 2011]

A *wf:Pool* (subclassing from *wf:Graph* and *wf:MessageVertex*) is the graphical representation of a participant in a collaboration. A participant represents a specific partner entity (e.g., a company) and/or a more general partner role (e.g., a buyer, seller, or manufacturer) that are participants in a collaboration. In Onto-ITIL, a *wf:Pool* is also a subclass of the *oc:Agent-Generic* concept representing the actor that participates in an *itil:Activity*. Furthermore, in our approach, using the *wf:diagramComposedOf* property, an *itil:Activity* is associated with an unique *wf:Pool* (i.e., the IT department responsible of the *itil:Activity*), which is also composed of an unique *wf:SubProcess* that represents the specification of the information system associated with the *itil:Activity* that the *wf:Pool* is responsible for (modeled using the *wf:graphComposedOf* and *oc:responsibleFor* properties).

A *wf:Lane* (subclassing from *wf:AssociationTarget* and *wf:NamedBpmnObject*) is a sub-partition within a *wf:Pool* which extends the entire length of the workflow level, either vertically or horizontally. Just like a *wf:Pool*, in Onto-ITIL, a *wf:Lane* is also a subclass of *oc:Agent-Generic*.

A *wf:Vertex* is a given node in a *wf:Graph*. A *wf:MessageVertex* represents nodes that can send and/or receive messages. A *wf:Activity* (subclassing from *wf:MessageVertex* and *wf:Vertex*) is work that is performed within a business process. A *wf:Activity* can be atomic or non-atomic (compound). The *wf:Activity* represents points in a process flow where work is performed. The *wf:Activity*(s) are the executable elements of a business process. As a vertex, *wf:Activity* may have associations. A *wf:Association* is used to associate information between artifacts (i.e., *wf:Artifact*, which is used to obtain the source of the *wf:Association*) and flow objects (i.e., *wf:AssociationTarget*, which is used to obtain the target of the *wf:Association*). There are different types of *wf:Activity*(s) modeled using the *wf:ActivityType* enumeration class. For example, *wf:Task* is an atomic *wf:Activity* within a flow. A *wf:Subprocess* is a composite *wf:Activity*, i.e., the specification of parameterized behavior as the coordinated sequencing of subordinate units whose individual elements are tasks. A subprocess is also modeled as a class (*wf:SubProcess* class, subclassing from *wf:Graph*) which represents a behavior whose internal details have been modeled using activities, gateways, events, and sequence flows. As every graph, a *wf:SubProcess* will have associated the artifacts that are contained in the graph.

A *wf:SequenceEdge* is used to connect nodes (*wf:Vertex*) in a *wf:Graph*. In *wf:SequenceEdge*, the *wf:objectName* datatype property represents the guard of the edge (i.e., the specification evaluated at runtime to determine if the edge can be traversed). A *wf:MessagingEdge* (subclassing from *wf:AssociationTarget* and *wf:NamedBpmnObject*) is used to connect messages nodes (*wf:MessageVertex*).

Following the approach defined by Ferrario and Guarino [2009], we present an *itil:Activity* (subclassing from *wf:BpmnDiagram*) as the service process that implements the service, i.e., the actions that ultimately lead to service production performed by the IT service provider (see Definition 6 in Section 3.2). These activities are carried out and coordinated by the specifications as part of a business process, during which documents or information are passed from one participant to another, according to a set of procedural rules. For example, in our pilot project, an instance of the *itil:Activity*, *itil:ICTD_IM_Activity*, specifies the workflow that defines the tasks to carry out when an incident is reported and it is related to the corresponding process instance, *itil:ICTD_IM_Process* (modeled using the *itil:specifiesActivity* property).

Chapter 4

Evaluation

> "Sennores e amigos, lo que dicho avemos Palabra es oscura, exponerla queremos: Tolgamos la corteza, al meollo entremos. Prendamos lo de dentro, lo de fuera dessemos." *Milagros de Nuestra Señora*
>
> Gonzalo de Berceo (1197-1264), *Spanish poet*

In this chapter we describe the prototype that we have implemented in order validate Onto-ITIL. As a proof of concept, we started a pilot project with a Spanish IT service provider (the *Information and Communication Technology Department* – ICTD – of a Spanish company) interested in improving the quality of the services they were delivering to their customers in order to obtain an optimal level of customer satisfaction and to become more competitive and efficient.

4.1 Implementation of the Prototype

This section describes the prototype developed in order to validate the proposed approach. The objectives included: (i) to improve customer satisfaction; (ii) to improve the quality of their services; (iii) to make use of a framework for: ITI process, activity and procedure definitions, metric identification and better technology access to service delivery; (iv) to be responsible for ITSM projects for high availability and reliability; and (v) to become a proactive organization. Being a SME company, they had limited time and resources to implement a comprehensive ITSMS. Therefore, the company decided to start adopting ITIL and to implement the incident management process, adapting it according to its business requirements using our approach. Figure 4.1 summarizes the process we followed to implement this prototype, consisting in four phases, briefly described in the following subsections.

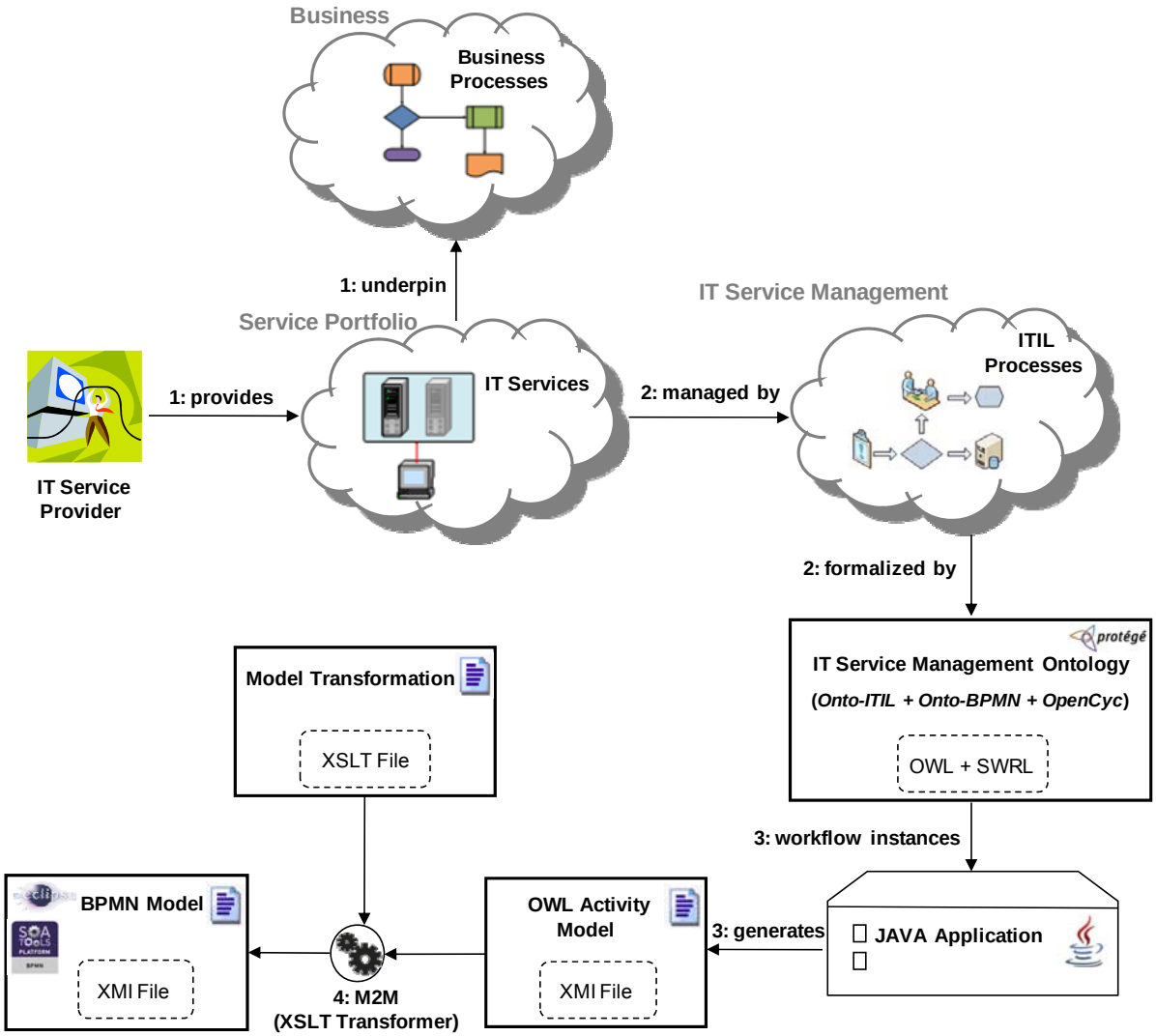


Figure 4.1 Architecture of Onto-ITIL

4.1.1 Stage 1: Service Portfolio

We start with the fact that the IT services are contained within a service portfolio belonging to an IT service provider. These IT services underpin the business processes of different organizations.

4.1.2 Stage 2: ITIL-compliant and Ontology-based IT Service Management

In order to assess the efficiency and quality of the IT services included in the service portfolio, a complete ITSM model is carried out according to Onto-ITIL. We use the Onto-ITIL Ontology to ease the integration of business information and IT for building ITSMSs in terms of ITIL processes. It provides mechanisms for semantic analysis (based on the underlying constraints), new knowledge inference, and SLA management, among others.

4.1.3 Stage 3: Business Process Modeling

In order to provide support to the implementation of the ITIL processes, we use the Onto-BPMN Ontology (included as part of the Onto-ITIL Ontology) for defining the workflows associated to each ITIL process.

4.1.4 Stage 4: Workflow Model Transformation

To manage the knowledge related to the ITIL process that is being automated through computer tools (*itil:Application*), those activities (*itil:Activity*) defined in Onto-ITIL Ontology can be included in the Eclipse platform for its total (or partially) automation by means of an information system. To accomplish this, a Java application is implemented which, (i) shows all of the instances of *itil:Activity* defined in the ontology; (ii) allows the user to establish which of these activities will be automated and implemented in the *itil:Application* as part of the ITSMS; and (iii) executes an XSLT script to transform the selected activities into a BPMN model, which conforms to the BPMN metamodel (obtained from the Eclipse BPMN modeler subproject developed for the STP project). The resulting BPMN model describes, at a very high-level of abstraction, the business processes to be implemented as part of the ITSMS.

4.2 Case study: Implementation of an Incident Management System

As previously mentioned, our approach is illustrated using a real case study of a Spanish IT service provider that wanted to implement the *Incident Management* process from the *Service Operation* stage, as a first step to improve the quality of their services. We selected this process to validate our work because the *Incident Management* process is highly visible to the business and, therefore, it is often one of the first processes to be implemented in ITSM projects [OGC, 2007d]. Also, this process is a relatively simple one with a reasonable number of classes and properties associated.

Starting with our pilot project (see Stage 1 in Subsection 4.1.1), an instance of the *itil:ITServiceProvider*, *itil:ICTD_provider*, provides several IT services (instances of *itil:CoreService* class), which are contained within *itil:ICTD_ServiceCatalog*: *itil:Access3G*, *itil:Backup*, *itil:MailingLists*, *itil:DataNetwork*, *itil:Microcomputing*, *itil:SWManagement*, *itil:SWLicensing*, *itil:Staff_email*...

The next subsections (4.2.1 and 4.2.2) describe the documentation associated to our *Incident Management* process model (see Stage 2 in Subsection 4.1.2).

4.2.1 The ITIL Incident Management Process

The ITIL V3 book on Service Operation [OGC, 2007c] describes best-practice advice and guidance on all aspects of managing the day-to-day operation of an organization's IT services. It covers issues relating to the people, processes, infrastructure technology and relationships necessary to ensure the high quality, cost-effective provision of IT service necessary to meet business needs. This fourth book in the ITSM lifecycle is concerned with business as usual activities.

Since *Incident Management* is the process responsible for managing the lifecycle of all incidents, it includes incident logging, incident escalation, trend and root cause analysis and resolution of incidents [ISACA, 2007]. ITIL defines an incident as *"an unplanned interruption to an IT service or reduction in the quality of an IT service. Failure of a configuration item (CI) that has not yet impacted service is also an incident, for*

example failure of one disk from a mirror set" [OGC, 2007c]. In this respect, we must not be confused by the term problem. Some people use either 'incident' or 'problem' but they are not the same. Incident and problem are not equivalent terms. In ITIL terminology a problem is defined as *"a cause of one or more incidents."* The cause of a problem is not usually known at the time a problem record is created, and the *Problem Management* process is responsible for further investigation [OGC, 2007c].

This process can include failures, questions or queries reported by the customers, by technical staff, or automatically detected and reported by even monitoring tools. The primary goal of *Incident Management* is to restore normal service operation to customers as quickly as possible (i.e., make sure that IT services are quickly available as required) and minimize the adverse impact on business operations, thus ensuring that the best possible levels of service quality and availability are maintained. In this way, incident resolution priorities with business imperatives must be aligned. Normal service operation is defined in ITIL as service operation within SLA limits.

There are several mechanisms in which business can benefit from the Incident Management process [OGC, 2007d] [ISACA, 2007]:

- The ability to detect and resolve incidents which results in lower downtime to the business, which in turn means higher availability of the service.
- The ability to increase productivity through quick resolution of customer queries, questions and incidents.
- The ability to address root causes, such as poor user training, through effective reporting.
- The ability to align IT activity to real-time business priorities. This is because Incident Management includes the capability to identify business priorities and dynamically allocate resources as necessary.
- The ability to identify potential improvements to services. This happens as a result of understanding what constitutes an incident and also from being in contact with the activities of business operational staff.
- The IT help desk can, during its handling of incidents, identify additional service or training requirements found in IT or the business.

The *Incident model* is a way of pre-defining the steps that should be taken to handle a process for dealing with a particular type of incident in a well-defined manner. Support tools can then be used to manage this process. In this way, we can ensure that all incidents are handled in a pre-defined path and within pre-defined timescales. Therefore, the incident model should include [OGC, 2007d]:

- Steps that should be taken to handle the incident.
- Chronological order these steps should be taken in, with any dependences or co-processing defined.
- Responsibilities, that is who should do what.
- Timescales and thresholds for completion of the actions.
- Escalation procedures, that is, who should be contacted and when.
- Any necessary evidence-preservation activities (particularly relevant for security- and capacity-related incidents).

Also, *Incident models* are input to the incident-handling support tools in use and the tools have to automate the handling, management and escalation of the process.

On the other hand, according to COBIT [ISACA, 2007], the RACI matrix related to the *Incident Management* process that maps activities to roles and defines how roles contribute to an activity is given in Table 4.1.

Table 4.1 RACI matrix for the Incident Management process

Activity	**Function**								
	CEO	**CIO**	**BPO**	**HO**	**CHA**	**HD**	**HA**	**CARS**	**IM**
Create classification (severity and impact) and escalation procedures (functional and hierarchical)		C	C	C	C	C	C	C	A/R
Detect and record incidents/service requests									A/R
Classify, investigate and diagnose queries		I		C	C	C		I	A/R
Resolve, recover and close incidents			I	R	R	R		C	A/R
Inform users (for example, status updates)		I	I						A/R
Produce management reporting	I	I	I	I			I	I	A/R

CEO: Chief Executive Officer; CIO: Chief Information Officer; BPO: Business Process Owner; HO: Head Operations; CHA: Chief Architect; HD: Head Development; HA: Head IT Administration; CARS: Compliance, Audit, Risk and Security; IM: Incident Manager

Taking all these aspects related to the *Incident Management* process into consideration, we defined the instance of the *itil:IncidentManagement* class, *itil:ICTD_IM_Process*, using the Protégé 3.4.4 ontology editor tool, that is shown in Figure 4.2.

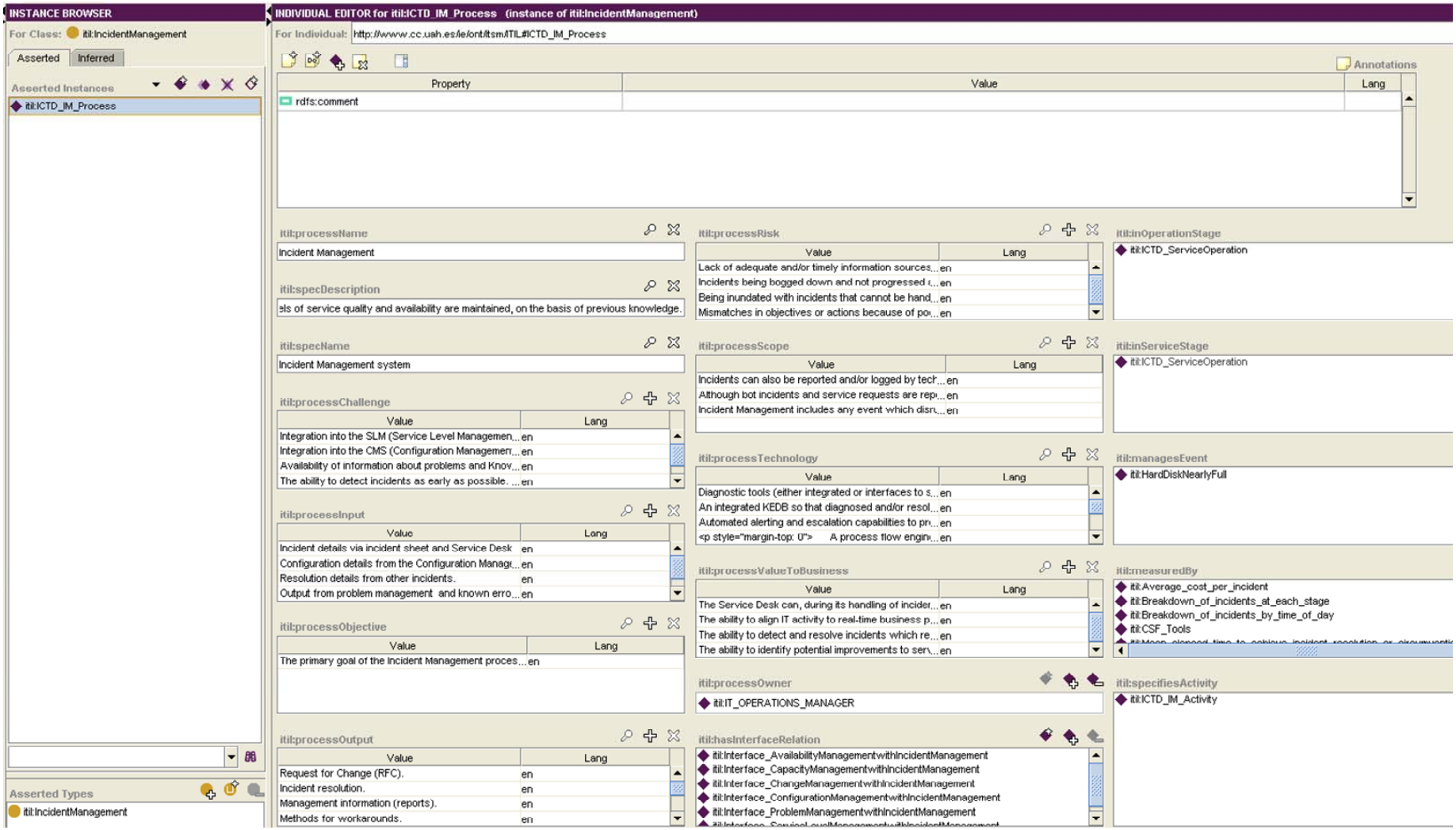


Figure 4.2 The itil:ICTD_IM_Process instance

4.2.2 The Incident Management Metrics Model

The objective of the metrics illustrated in Table 4.2, which have been included in our pilot project (ICTD) from [Steinberg, 2006], is to determine the efficiency and effectiveness of our *Incident Management* process, and its operation.

Table 4.2 Operational metrics for the Incident Management process

ID	Metric
A	Total number of incidents
B	Average time to resolve severity 1 and severity 2 incidents (hours)
C	Number of incidents resolved within agreed service levels
D	Number of high severity / major incidents
E	Number of incidents with customer impact
F	Number of incidents reopened
G	Total available labor hours to work on incidents (non-Service Desk)
H	Total labor hours spent resolving incidents (non-Service Desk)
I	Incident Management Tooling Support Level
J	Incident Management Process Maturity

Table 4.3 lists the suggested KPIs and how they are calculated from the previous operational metrics [Steinberg, 2006].

Table 4.3 KPIs for the Incident Management process

ID	KPI	Calculation
1	Number of incident occurrences	A
2	Number of high severity / major incidents	D
3	Incident resolution rate	C/A
4	Customer incident impact rate	E/A
5	Incident reopen rate	F/A
6	Average time to resolve severity 1 and severity 2 incidents (hours)	B
7	Incident labor utilization rate	H/G
8	Incident management tooling support level	I
9	Incident management process maturity	J

These KPIs are critical to manage and monitor *Incident Management* activities. Table 4.4 lists each KPI and the question is trying to answer [Steinberg, 2006].

Table 4.4 KPI objectives

KPI	Question being answered
Number of incident occurrences	How many incidents did we experience within our infrastructure?
Number of high severity / major incidents	How many major incidents did we experience?
Incident resolution rate	How successful are we at resolving incidents per business requirements?
Customer incident impact rate	How well are we keeping incidents from impacting customers?
Incident reopen rate	How successful are we at permanently resolving incidents?
Average time to resolve severity 1 and severity 2 incidents (hours)	How quickly are we resolving incidents?
Incident labor utilization rate	How much available labor capacity was spent handling incidents?
Incident management tooling support level	How well does our current tool set support Incident Management activities?
Incident management process maturity	How well do we execute our Incident Management practices?

The *Critical Success Factors* (CSFs) associated with the *Incident Management* process are listed in Table 4.5. This information provides CIOs, CEOs and BPOs with indicators from which they can make accurate and timely business decisions and with confidence that IT is managing itself well [Steinberg, 2006].

Table 4.5 CSFs for the Incident Management process

CSF	KPI
Quickly resolve incidents	5,6,8
Maintain IT service quality	1,2,3,4,8,9
Improve IT and business productivity	7,8
Maintain user satisfaction	4,8,9

Table 4.6 summarized the evaluation of the metrics model related to the *Incident Management* process after applying the proposed approach in the ICTD (data were collected six months after the implementation).

Table 4.6 Evalutation of the KPIs for the Incident Management process in the pilot project

ID	KPI	Before adopting ITIL	Using our approach
1	Number of incident occurrences	220	103
2	Number of high severity / major incidents	76	65
3	Incident resolution rate	81.82%	97,31%
4	Customer incident impact rate	81.82%	97,31%
5	Incident reopen rate	12.27%	8,23%
6	Average time to resolve priority 10 and priority 9 incidents (hours)	Unknown	45 minutes
7	Incident labor utilization rate	Unknown	35%
8	Incident management tooling support level	Low	Medium
9	Incident management process maturity	Unknown	Managed

As shown in Table 4.6, the results show clear improvements in the *Incident Management* process. Now, the ICTD can measure some aspects not taken into account before the ITIL implementation. However, physical intervention (non-Service Desk) spent a lot of time resolving customer incidents and thus, it will need further investigation by the *ICTD*'s *Incident Manager*.

4.2.3 The Incident Management Activity

In order to manage the computer tools that the organization required for the *Incident Management* process, we defined the workflow that describes the *Incident Management* process adapted to our pilot project (see Stage 3 in Subsection 4.1.3). Figure 4.3 shows the workflow representing the business process for *Incident Management*.

This business process (*itil:ICTD_IM_Activity*) was defined in terms of our Onto-BPMN Ontology (part of the Onto-ITIL Ontology) using the Protégé 3.4.4 ontology editor (see Figure 4.4.) In this case, as we explained earlier, we only have one pool instance (*itil:ICTD_Pool_IncidentManagement)* associated with the subprocess instance (*itil:ICTD_IncidentManagementSystem*) that contains all the elements of the workflow (see Figures 4.5 and 4.6).

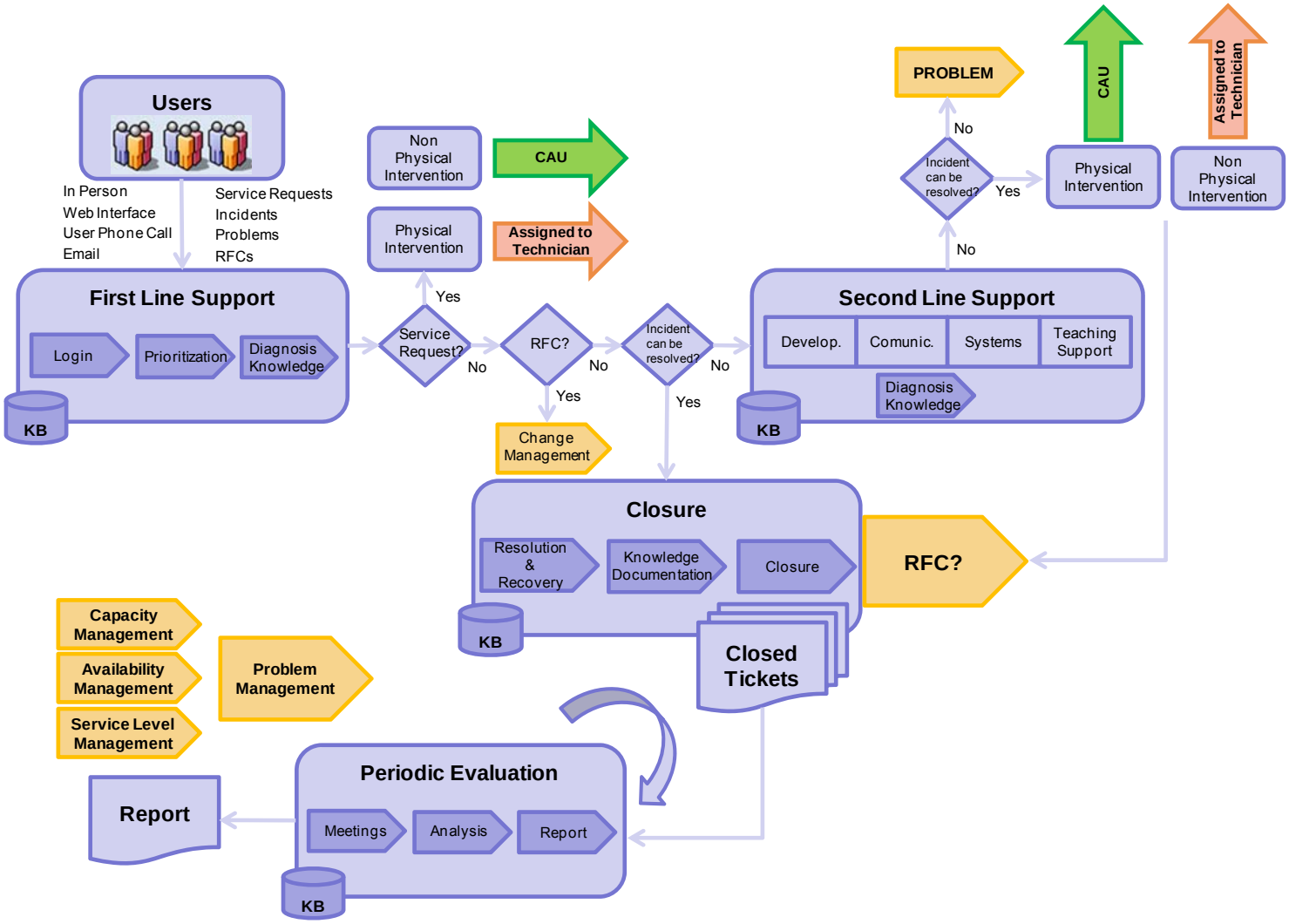


Figure 4.3 ICTD Incident Management Business Process

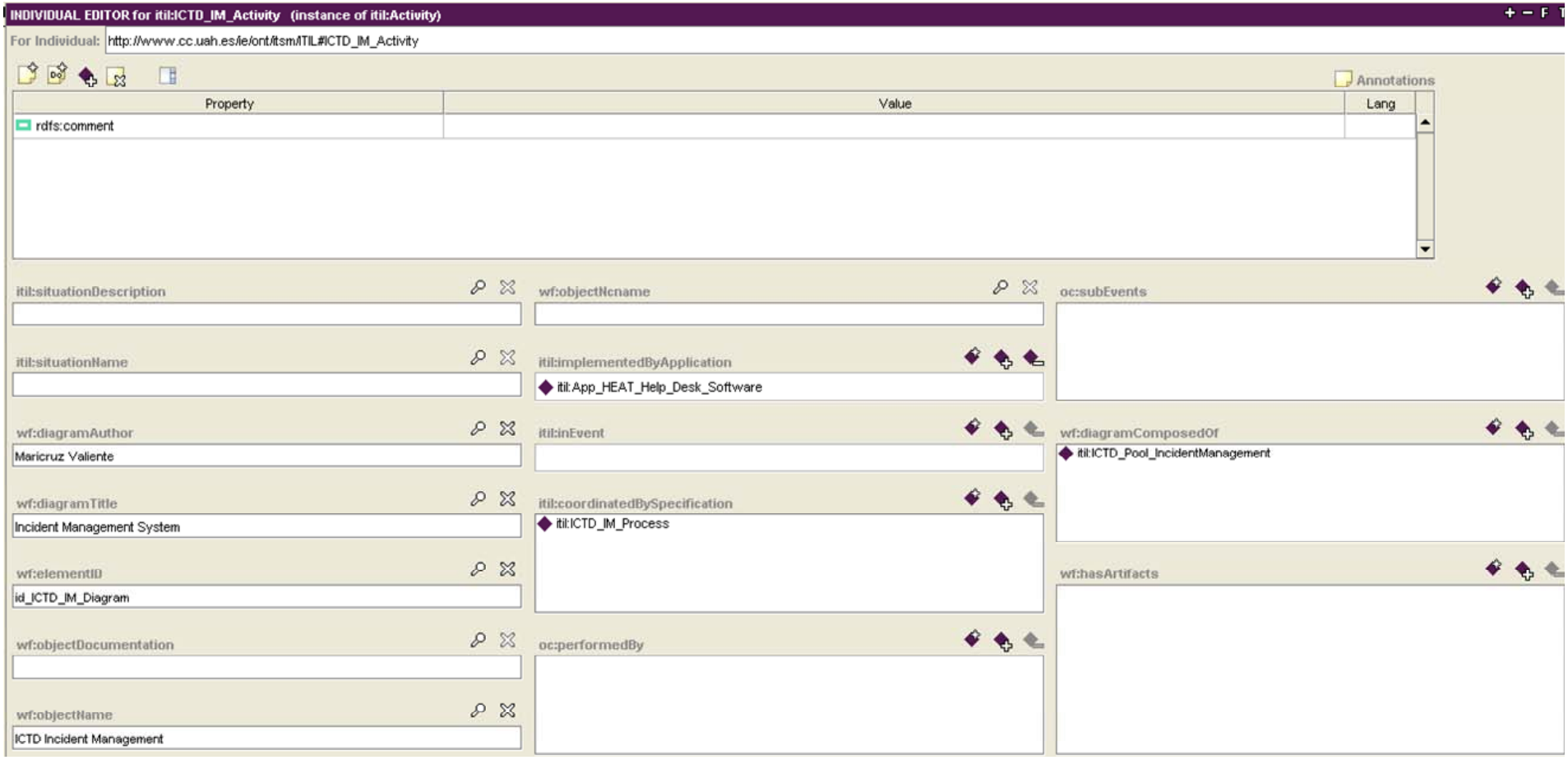


Figure 4.4 The itil:ICTD_IM_Activity instance

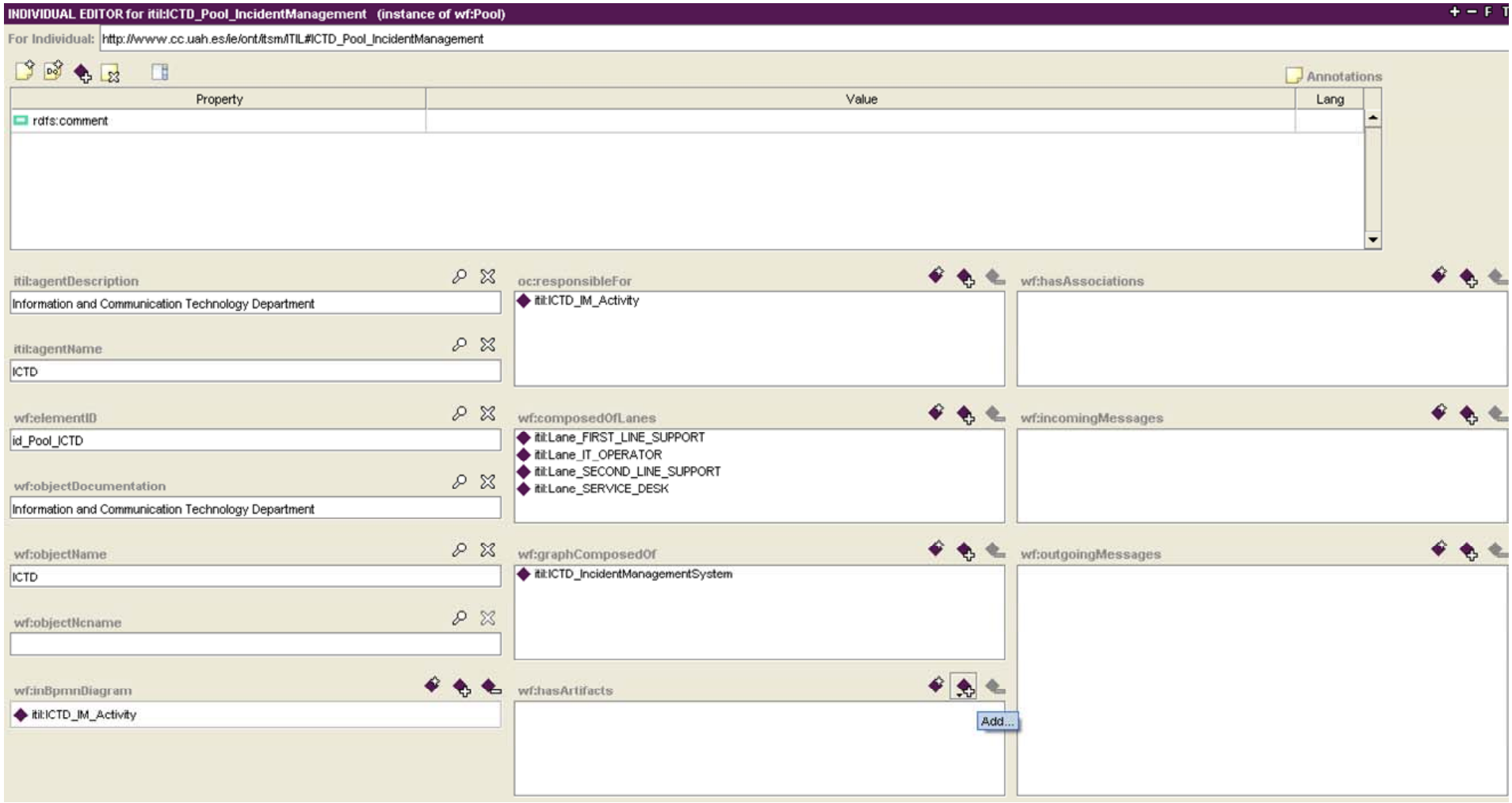


Figure 4.5 The itil:ICTD_Pool_IncidentManagement instance

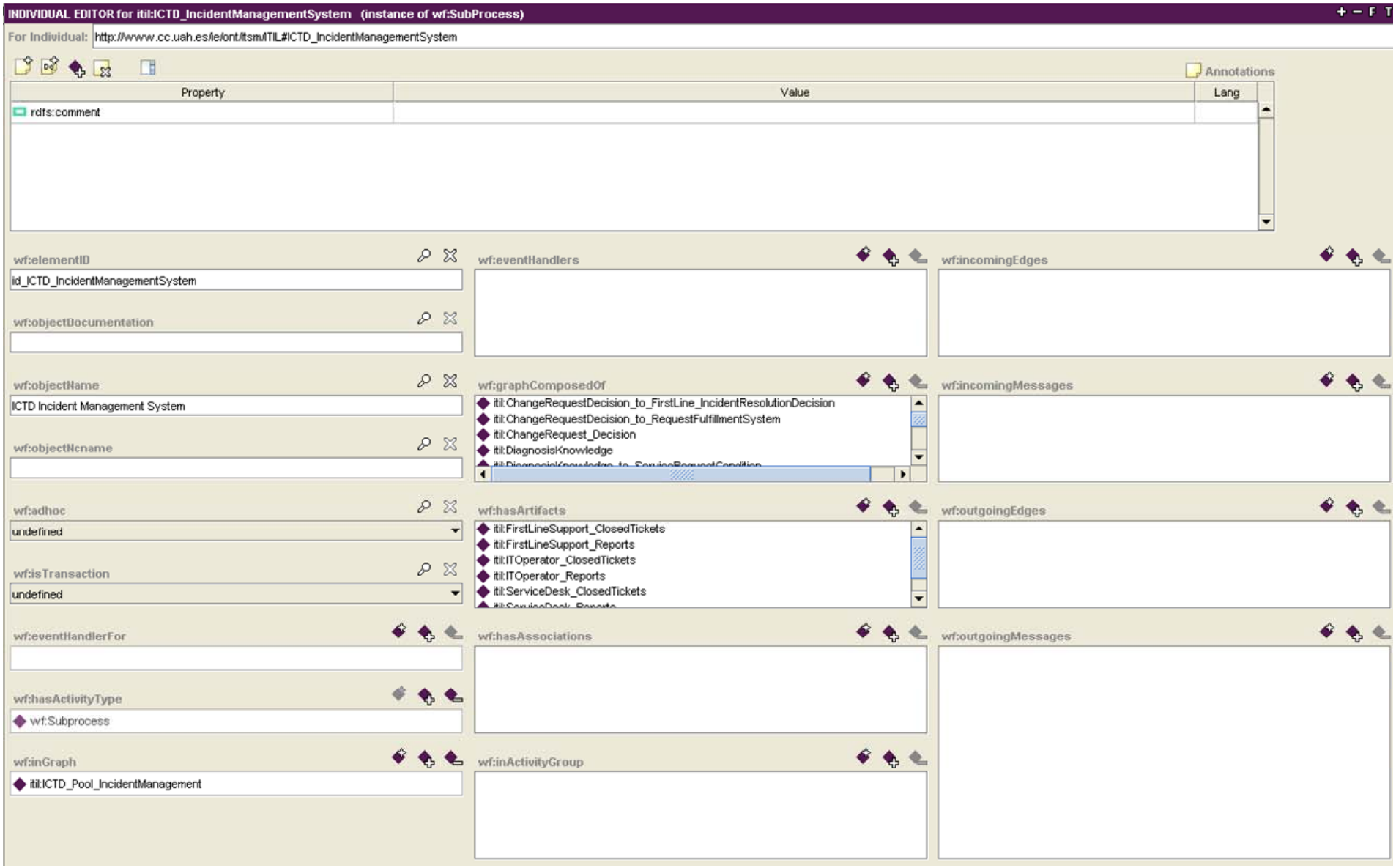


Figure 4.6 The itil:ICTD_IncidentManagementSystem instance

4.2.4 XSL Transformation

Once we had defined the workflow related to the *Incident Management* process in terms of our Onto-BPMN Ontology, we used this knowledge to obtain the conceptual model of the ITSMS needed to support it (see Stage 4 in Subsection 4.1.4). For this purpose, we used a java file (OWL2BPMN_client.java) which: (i) presents to the user all the *itil:Activity* instances in the ontology, allowing him to select those to be automated (see Figure 4.7); (ii) creates an XMI-serialized Onto-ITIL model for each selected activity (*OWL2BPMNTransformer_XSLT.java*) (Figure 4.8 shows an excerpt of the *ICTD_IM_Activity.onto_itil* file generated from the *itil:ICTD_IM_Activity* instance); and (iii) generates a XMI-serialized BPMN models for the resulting OWL models by using an XSLT script (*OWL2BPMNTransformer.xslt*) (Figures 4.9 and 4.10 shows an excerpt of the *ICTD_IM_Activity.bpmn* file resulting from the model transformation). The resulting BPMN model can also be opened and edited (in diagram form) using the Eclipse BPMN Diagram Editor (see Figure 4.11).

Table 4.7 lists the mappings among Onto-ITIL Activity constructs and BPMN constructs. For example, in an Onto-ITIL Activity model, the element *Activity* associated with the element *graphComposedOf* is transformed into the element vertices *xmi:type="bpmn:Activity"* in a BPMN diagram.

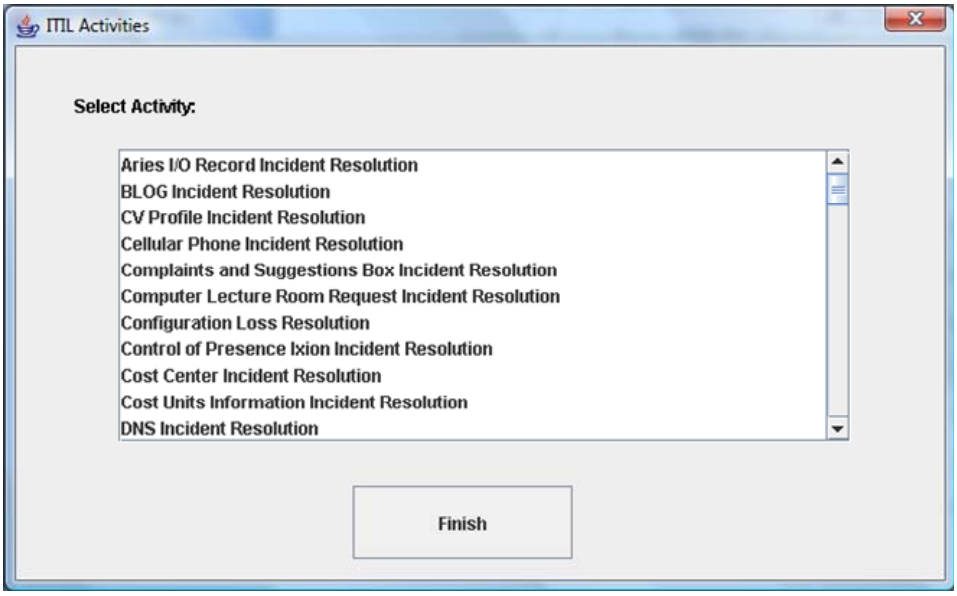


Figure 4.7 ITIL Activities Selection

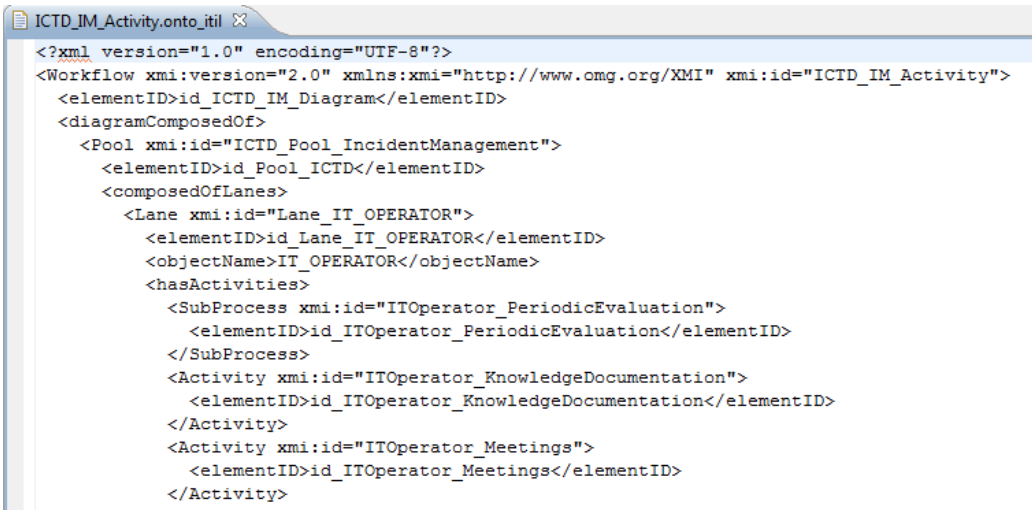


Figure 4.8 Excerpt of the ICTD_IM_Activity.onto_itil file (Eclipse Text Editor)

Table 4.7 Mapping of Onto-ITIL Activity and BPMN constructs

Onto-ITIL Activity Model	Type	BPMN Model	Type
Activity (associated with the element <graphComposedOf>)	element	vertices (with the attribute xmi:type="bpmn:Activity")	element
Activity (associated with the element <hasActivities>)	element	activities (associated with the element <lanes>)	attribute
Association	element	associations (with the attribute xmi:type="bpmn:Association")	element
DataObject	element	artifacts (with the attribute xmi:type="bpmn:DataObject")	element
elementID	element	iD	attribute
hasActivityType	element	activityType	attribute
Lane (associated with the element <composedOfLanes>	element	lanes (associated with the element <pools> and with attribute xmi:type="bpmn:Lane")	element
Lane (associated with the element <inActivityGroup>	element	lanes (associated with the element <vertices>)	attribute
objectName	element	xmi:name	attribute
Pool	element	pools (with the attribute xmi:type="bpmn:Pool")	element
SequenceEdge (associated with the element <graphComposedOf>)	element	sequenceEdges (with the attribute xmi:type="bpmn:SequenceEdge")	element
SequenceEdge (associated with the element <incomingEdges>)	element	incomingEdges (associated with the element <vertices>)	attribute
SequenceEdge (associated with the element <outgoingEdges>)	element	outgoingEdges (associated with the element <vertices>)	attribute
SubProcess	element	vertices (with the attribute xmi:type="bpmn:SubProcess")	
TextAnnotation	element	artifacts (with the attribute xmi:type="bpmn:TextAnnotation")	element
xmi:id	attribute	xmi:id	attribute
workflow	element	bpmn:BpmnDiagram	element

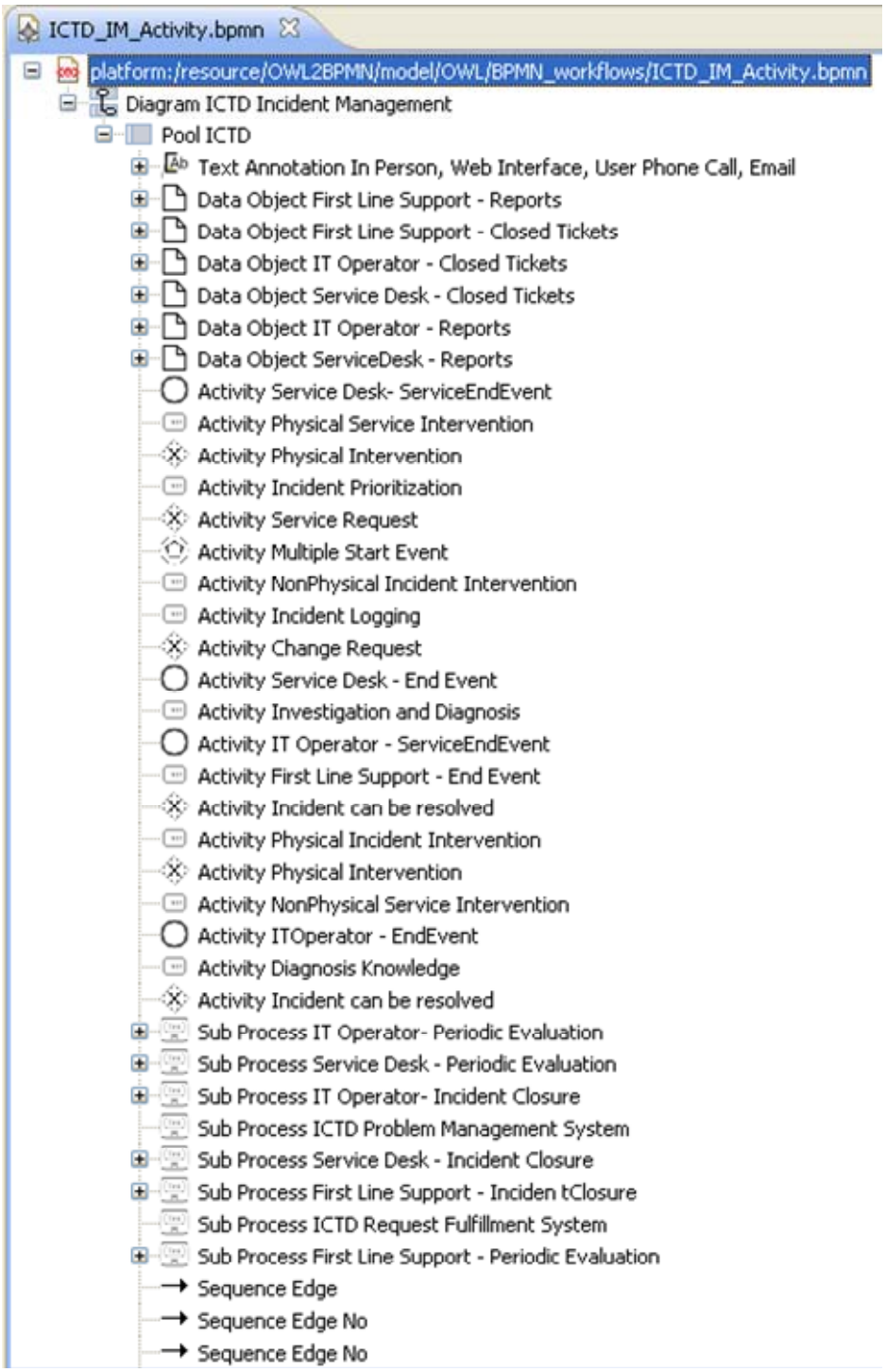


Figure 4.9 Excerpt of the ICTD_IM_Activity.bpmn file (Eclipse Refl. Ecore Model Editor)

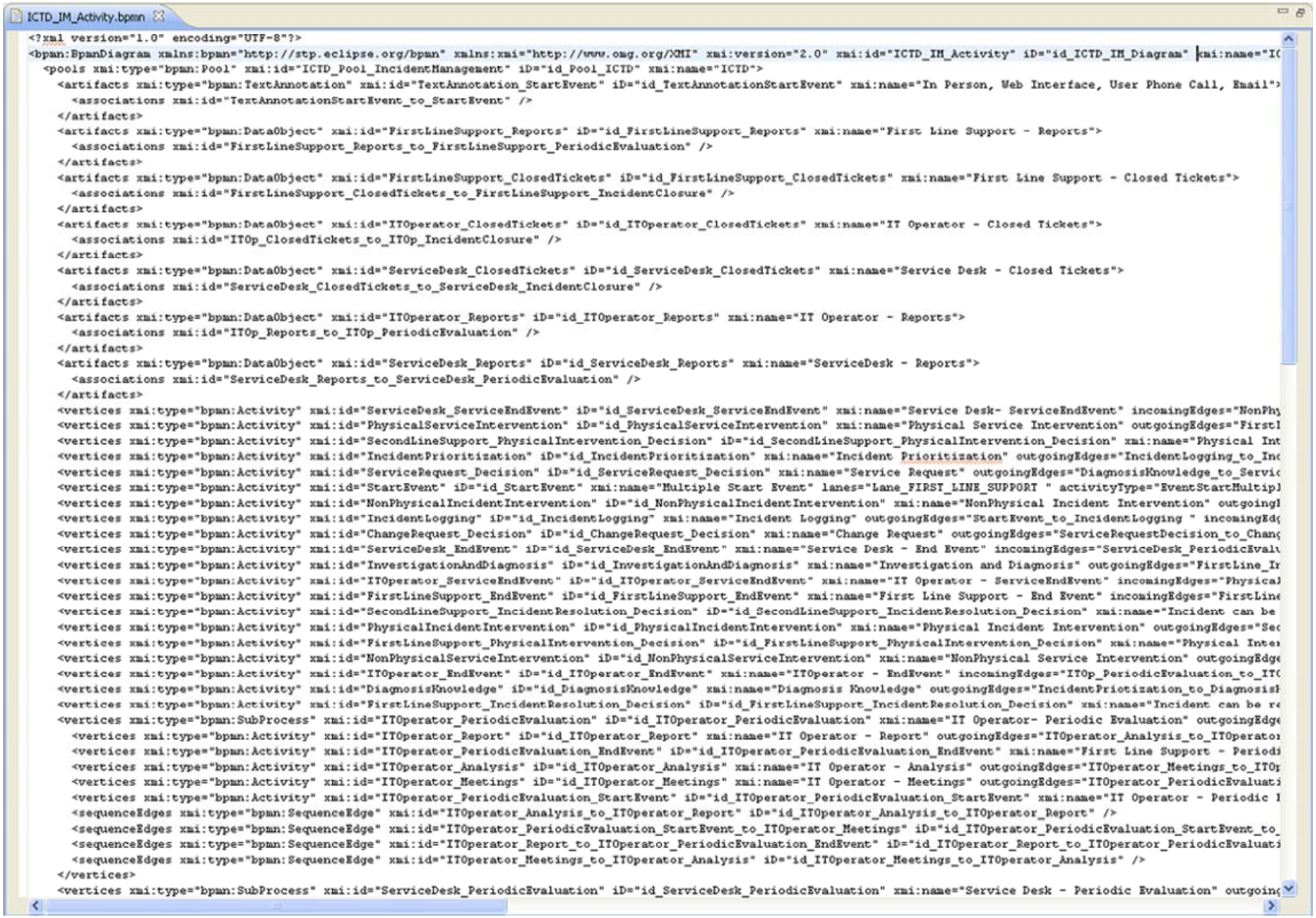


Figure 4.10 Excerpt of the ICTD_IM_Activity.bpmn file (Eclipse Text Editor)

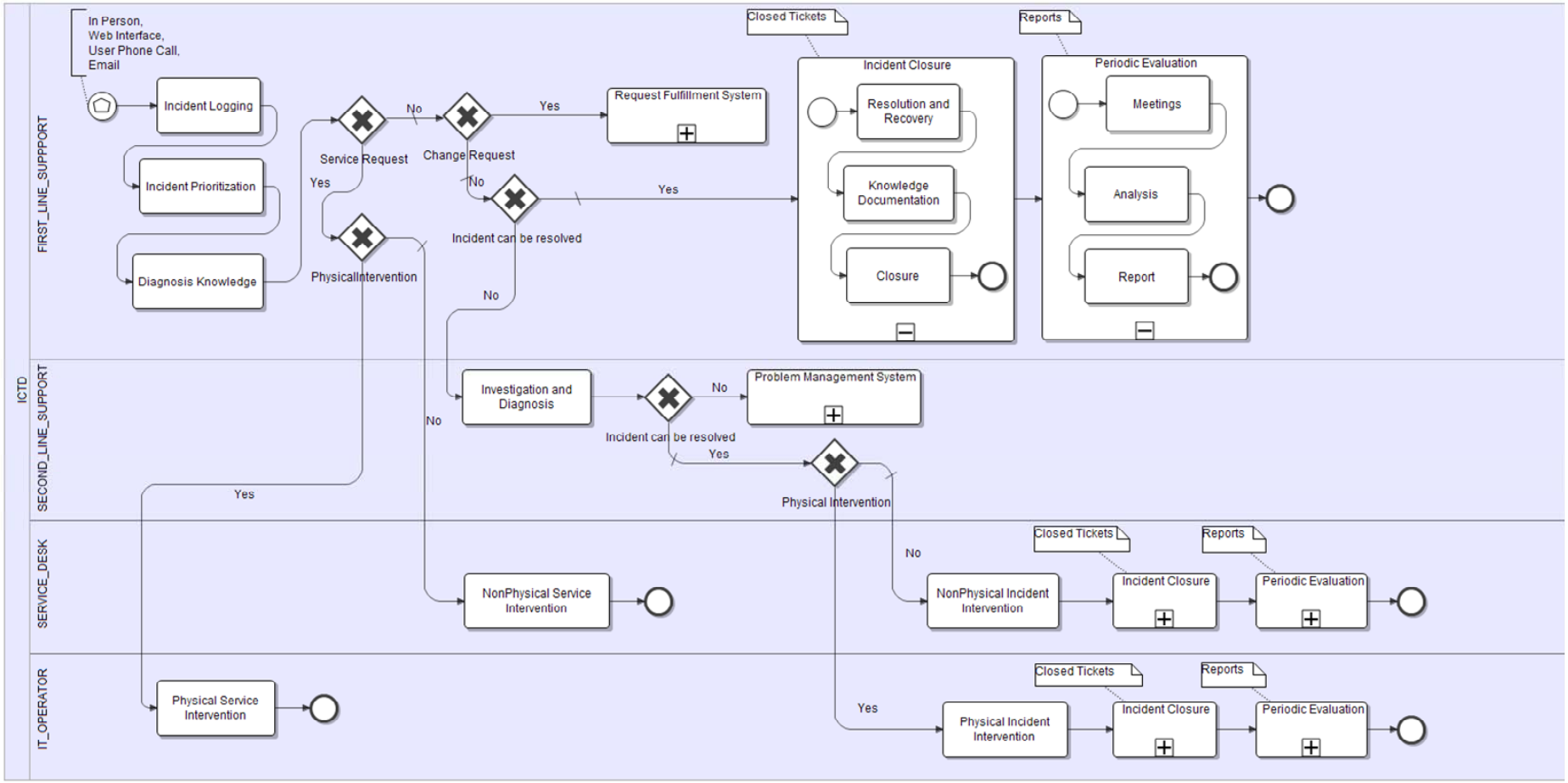


Figure 4.11 BPMN diagram related to the IM Activity (Eclipse Bpmn Diagram Editor)

4.2.5 Ontology Queries, Rule-based Constraints and Knowledge Inference

Finally, for the ICTD pilot project, we have defined a set of SWRL rules for model consistency checking, model validation, and business rule analysis. These rules can be executed on Onto-ITIL Ontology using Protégé and the Jess rule engine, allowing us to both verify constraints and inconsistencies in the incident model, and to incorporate new inferred knowledge into the ontology. Also, queries to the ontology and its knowledge base are performed using SQWRL. These extensions to the ontology demonstrate the feasibility and benefits of Onto-ITIL, as the combined use of the ontology with queries and rules provide us with all the relevant aspects of the ITIL specification as well as dynamic capabilities capable of improving the management of their IT services. The following subsections further describe the three types of rules we have defined in SWRL and SQWRL for (i) *model consistency*, (ii) *SLA breaches* and (iii) *proactive actions*.

Model Consistency Rules

Model consistency rules are applied to all the instances included in Onto-ITIL models. We now provide examples of model consistency rules. The following rule states that, although each service process is part of a unique stage, in order to improve their reusability in the *ITIL Service Lifecycle*, it is possible to have the same process related to different stages, but with the same type:

```
itil:IncidentManagement(?p) ∧ itil:ServiceStage(?s1) ∧ itil:ServiceStage(?s2) ∧
differentFrom(?s1,?s2) ∧ itil:inServiceStage(?p,?s1) ∧ itil:inServiceStage(?p,?s2)
→
itil:ServiceOperation(?s1) ∧ itil:hasProcess(?s1,?p) ∧
itil:ServiceOperation(?s2) ∧ itil:hasProcess(?s2,?p)
```

This rule states that if an incident management process (*p*) takes part in different service stages (s1, *s2*), then *s1* and *s2* must represent service operation stages (i.e., instances of *itil:ServiceOperation*), and both *s1* and *s2* must have *p* as an associated process.

Similarly, the next rule defined states that if a KPI is related to a specific process then, given that a KPI is a metric that enables business decisions in the delivery of a service it must be a metric belonging to the IT service associated with the process:

```
itil:ITService(?serv) ∧ itil:ServiceLifecycle(?l) ∧
itil:hasServiceLifecycle(?serv,?l) ∧ itil:ServiceOperation(?st) ∧
itil:inServiceLifecycle(?st,?l) ∧ itil:OperationProcess(?p) ∧
itil:hasOperationProcess(?st,?p) ∧ itil:KPI(?m) ∧ itil:measures(?m,?p)
→
itil:definesMetric(?serv,?m)
```

In this case, if an IT service (*serv*) has a service lifecycle (*l*), and a service operation stage (*st*) is part of *l*, and an operation process is one of the processes included in *st*, and *m* is a KPI that measures *p*, then *serv* must have *m* also as a defined metric.

The next rule shows how it is possible to force the computation of a specific metric in order to document it and test its results following the metrics model proposed in [Steinberg, 2006]:

```
itil:OperationalMetric(itil:Total_number_of_incidents) ∧
itil:OperationalMetric(itil:Number_of_incidents_resolved_within_agreed_service_levels) ∧
itil:metricValue(itil:Total_number_of_incidents,?v1) ∧
itil:metricValue(itil:Number_of_incidents_resolved_within_agreed_service_levels,?v2) ∧
itil:KPI(itil:Incident_resolution_rate) ∧
swrlb:divide(?result, ?v2, ?v1)
→
itil:metricValue(itil:Incident_resolution_rate,?result)
```

where, the KPI *k* associated with the incident resolution rate, is defined as the ratio (*result*) between the number of incidents resolved within the agreed service levels and total number of incidents.

As a final example, the following SQWRL query extracts the list of incidents associated with each customer group managed by a specific IT service provider as part of its *Incident Management* process. The results of this query can help IT service providers to decide whether or not the incidents have been properly assigned and managed:

```
itil:Incident(?i) ∧ itil:IncidentManagement(?p) ∧ itil:managesEvent(?p,?i) ∧
itil:situationName(?i,?name) ∧ itil:hasIncidentRecord(?i,?r) ∧
itil:incidentPriority(?r,?pr) ∧ itil:hasIncidentGroup(?r,?gr) ∧
itil:incidentPriority(?r,?pr)
→
sqwrl:select(?name,?gr,"Number of incidents") ∧ sqwrl:count(?r) ∧
sqwrl:columnNames("Name","Priority","Description","Count")
```

SLA Breaches

SLA breaches are rules that check whether the agreed level of assurance or warranty regarding the level of service quality achieved by IT service providers for each of the services delivered to their customers is met. For example, in our pilot project, the priority of an *incident* is used to obtain the maximum resolution time agreed. Therefore, we define the following SWRL rule to assign an agreed resolution time (hours) to a specific customer:

```
itil:CoreService(itil:Access3G) ∧
itil:ServiceLifecycle(itil:ICTD_ServiceLifecycle) ∧
itil:hasServiceLifecycle(itil:Access3G, itil:ICTD_ServiceLifecycle) ∧
itil:SLA(itil:SLA_CUSTOMER_1) ∧
itil:coveringITService(itil:SLA_CUSTOMER_1, itil:Access3G) ∧
itil:SLAIncidentResolution(itil:CUSTOMER_1_SLAIncidentResolution_10)
```

→

```
itil:hasSLAIncidentResolution(itil:SLA_CUSTOMER_1,
                                    itil:CUSTOMER_1_SLAIncidentResolution_10) ∧

itil:slaIncidentPriority(itil:CUSTOMER_1_SLAIncidentResolution_10, 10) ∧

itil:slaIncidentResolutionTime(itil:CUSTOMER_1_SLAIncidentResolution_10, 12)
```

In this case, the SLA *itil:SLA_CUSTOMER_1* for a specific customer (*itil:SLA_CUSTOMER_1*) in the service *itil:Access3G* states that for an incident of priority 10, the maximum resolution time is 12 hours. In addition, the priority of a specific incident is calculated according to the following rules:

```
itil:ITService(?serv) ∧ itil:serviceImportanceCode(?serv,?code) ∧

itil:ServiceLifecycle(?l) ∧ itil:hasServiceLifecycle(?serv,?l) ∧

itil:ServiceOperation(?st) ∧ itil:inServiceLifecycle(?st,?l) ∧

itil:IncidentManagement(?p) ∧ itil:hasOperationProcess(?st,?p) ∧

itil:Incident(?i) ∧ itil:managesEvent(?p,?i) ∧ itil:IncidentRecord(?r) ∧

itil:hasIncidentRecord(?i,?r)

→

itil:incidentUrgency(?r,?code)                                          (1)
```

```
itil:ITService(?serv) ∧ itil:serviceUsers(?serv,?usr) ∧

itil:ServiceLifecycle(?l) ∧ itil:hasServiceLifecycle(?serv,?l) ∧

itil:ServiceOperation(?st) ∧ itil:inServiceLifecycle(?st,?l) ∧

itil:IncidentManagement(?p) ∧ itil:hasOperationProcess(?st,?p) ∧

itil:Incident(?i) ∧ itil:managesEvent(?p,?i) ∧ itil:IncidentRecord(?r) ∧

itil:hasIncidentRecord(?i,?r)

→

itil:incidentImpact(?r,?usr)                                            (2)
```

```
itil:Incident(?i) ∧ itil:IncidentRecord(?r) ∧

itil:hasIncidentRecord(?i,?r) ∧ itil:incidentUrgency(?r,?u) ∧

itil:IncidentGroupType(itil:GOVERNANCE) ∧

itil:hasIncidentGroup(?r,itil:GOVERNANCE)

→

itil:incidentLevel(?r,?u)                                               (3)
```

```
itil:Incident(?i) ∧ itil:IncidentRecord(?r) ∧
itil:hasIncidentRecord(?i,?r) ∧ itil:incidentUrgency(?r,?u) ∧
itil:IncidentGroupType(itil:STAFF) ∧ itil:hasIncidentGroup(?r,itil:STAFF) ∧
swrlb:equal(?u, 3)
→
itil:incidentLevel(?r, 2)                                                    (4)

itil:Incident(?i) ∧ itil:IncidentRecord(?r) ∧ itil:hasIncidentRecord(?i,?r) ∧
itil:incidentLevel(?r,?l) ∧  itil:incidentImpact(?r,?imp) ∧
swrlb:equal(?l, 5) ∧  swrlb:greaterThan(?imp, 10000)
→
itil:incidentPriority(?r, 10)                                                (5)

itil:Incident(?i) ∧ itil:IncidentRecord(?r) ∧ itil:hasIncidentRecord(?i,?r) ∧
itil:incidentLevel(?r,?l) ∧ itil:incidentImpact(?r,?imp) ∧
swrlb:equal(?l, 0) ∧ swrlb:greaterThan(?imp, 10000)
→
itil:incidentPriority(?r, 5)                                                 (6)
```

This is an example of rule chaining, where rule **(1)** calculates the incident urgency from the level of importance (*code*) of the affected IT service (*serv*). Then, rule **(2)** calculates the incident impact from the number of users (*usr*) of the affected service (*serv*). Rules **(3)** and **(4)** calculate the level of an incident (*i*) from the incident urgency (*u*) and from the type of group (*g*) that reported the incident. For example, if the incident has been reported by the 'GOVERNANCE' group, then the incident level must be equal to the incident urgency, but if the incidence has been reported by the 'STAFF' group, then the incident level could be less than the incident urgency. Finally, rules **(5)** and **(6)** make use of the incident impact (*imp*) and the incident level (*l*), respectively, to assign the incident priority. The organization states that the impact, urgency and priority codes range from 0 to 10, being 10 the highest priority.

Proactive Actions

Proactive actions are rules aimed to help organizations define how to act in order to prevent possible service failures that may occur in the future. The following example by Jerphanion and Kristelijn, describes a situation requiring a proactive action: *"An IT employee observes that the central hard disks are nearly full. He knows that this will lead to service failure in the near future, which will generate incidents. To prevent these incidents from occurring and to make sure that the service will remain available, the IT employee takes actions"* [Jerphanion & Kristelin, 2008]. According to ITIL, proactive actions are defined as part of one (or a combination of) different processes. Previous proactive action when a nearly full hard disk is detected can be expressed in SWRL rule as follows:

```
itil:Event(itil:HardDiskNearlyFull) ∧ itil:IncidentManagement(?p)
```

$\rightarrow$

```
itil:hasEventType(itil:HardDiskNearlyFull, itil:WARNING) ∧
itil:hasManagedEventType(itil:HardDiskNearlyFull, itil:PROACTIVE_PASSIVE) ∧
itil:managesEvent(?p, itil:HardDiskNearlyFull)
```

In our pilot project, the event of hard disk nearly full is managed by the *itil:ICTD_IM_Process* instance, and it is considered as a warning event whose type of monitoring and control is proactive and passive.

Chapter 5

Conclusions and Future Research

"A journey of a thousand miles begins with a single step"

Confucius *(*551 BC-478 BC)*, Chinese philosopher*

In this thesis, the lack of formal semantics of current ITSM best practices is addressed adopting an ontological approach (that is, *Ontology Engineering* – OE –). We aimed at translating perceptions of the real-world expressed in natural language and graphical representations to an ontology, which is a formal representation of the ITSM domain. The aim of the proposed ontology, Onto-ITIL, was to support: (i) business and IT integration in terms of ITIL implementations; (ii) ITSM knowledge representation; (iii) ITSM formal taxonomy development; (iv) ITSM metrics model; (v) reasoning capabilities; (vi) SLA management; and (vii) the sharing, reuse and interchange of the ITSM knowledge by using different e-business frameworks in the context of B2B commerce.

The proposed ontology captures best practices described in the ITIL framework for both representing services so that organizations can understand their ITSM processes (e.g., maturity level) and for business decision making (based on an ITSM metrics model) that can be executed thanks to semantics capabilities.

The standardization of terminology is another problem in ITSM/ITIL. The diversity of backgrounds causes IT professionals to use similar terminology in many different ways with many different connotations. Because of such differences, the information that one IT professional intends to communicate may, in fact, become garbled. Therefore, in the course of ITSM projects it is necessary to standardize the relevant vocabulary. In this vein, the Onto-ITIL Ontology provides a common terminology (which aligns with that adopted in AENOR), avoiding semantic ambiguities, uncertainties, and contradictions.

The ITSM metrics model, included as part of Onto-ITIL, enables IT service providers to know and understand the KPIs that should be used to measure IT services. These indicators will allow IT service providers to make business decisions. The Onto-ITIL metrics model can be used to test the impact of those decisions on KPIs and CSFs (i.e., how KPIs change according to new scenarios). Also, the Onto-ITIL metrics model can act as a basis for identifying and prioritizing IT service improvements, such as acquisition of new resources and computer tools to support the ITSM processes.

To represent workflow knowledge we have developed the Onto-BPMN Ontology, included as part of the Onto-ITIL Ontology. The Onto-BPMN Ontology is a formalization in OWL of the BPMN constructs.

In addition, we have defined a model-driven approach that helps bridging the current gap between OE and SE with regard to the development of information systems related to ITSMSs in order to maintain and improve IT service quality in line with business requirements. We must keep in mind that, during the analysis phase (i.e., conceptual modeling) of any software system, the emphasis must be placed on the data or in the information (i.e., in the system domain) rather than in the operations (i.e., in the behavior). In this vein, ontologies allow us: (i) to represent models of the real-world in terms of conceptual models used by computers; (ii) to represent abstract domain key concepts appropriately; and (iii) to transform these concepts correctly. Through the definition of the Onto-ITIL Ontology, we introduce the usage of semantic information during conceptual modeling of ITSMSs. This allows us to formalize and coherently and consistently describe all of the knowledge related both to ITSM best practices and to service management, including the workflows related to service implementation. Thus, each ontology-based workflow model represents a perspective of an information system that supports a specific ITSMS. Using our approach, we create models that conceptually represent the workflow-based information systems we need to build for ITSM. For this purpose, we match the Onto-BPMN Ontology with the information system conceptual modeling in terms of the BPMN metamodel, enabling the integration of our workflow specifications into the Eclipse platform.

A real case study regarding the implementation of an *Incident Management* process, carried out by a Spanish IT service provider, has been used to illustrate the feasibility and the benefits of the proposed approach.

The work presented in this thesis can be extended in several directions:

- Further development of the ontologies and rules for other particular ITSM processes and additional case studies for evaluation purposes.
- The definition of workflows using the proposed ontology is a complex task. Thus, workflow modeling should be enhanced in the Onto-BPMN Ontology. For example, Eclipse-based BPMN models could be also transformed into instances inside the Onto-ITIL Ontology by combining them with the workflow part of the ontology (Onto-BPMN Ontology). Therefore, the resulting instances will be in accordance with the ITIL framework and they could be enriched with semantics and constraints in the form of SWRL rules using Protégé and Jess (or any other Semantic Web programming frameworks as HP Jena[33]). The rules could be executed in order to verify constraints and inconsistencies in the instances, and to incorporate new knowledge into the workflow to better software development for ITSM.
- Eclipse-based BPMN models provide support for executable service process models with computational semantics. Thus, the resulting specifications could be transformed into WS-BPEL[34] by means of a M2M transformations (e.g., using ATL [Doux et al., 2009]) in order to allow their execution.
- The implementation of a M2T transformation (e.g., using the MOFScript Eclipse plug-in) that enables the generation of HTML documents from the BPMN models. This will allow the users to generate an easy to navigate documentation associated to each business process described in the BPMN model. The BPMN models (depicted using the BPMN Modeler Eclipse plug-in), together with the HTML documentation generated from them, could be used to complement the *Software Requirements Document* (SRD).

[33] http://jena.sourceforge.net/
[34] http://www.eclipse.org/bpel/

References

[Abramowicz et al., 2007]

Abramowicz, W., Filipowska, A., Kaczmarek, M. & Kaczmarek, T. (2007). Semantically enhanced business process modelling notation. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007) in conjunction with the 3rd European Semantic Web Conference (ESWC 2007)*, Innsbruck, Austria.

[Alanen, 2007]

Alanen, M. (2007). *A Metamodeling Framework for Software Engineering*. Turku Centre for Computer Science, TUCS Dissertations, No. 89.

[Assmann et al., 2006]

Assmann, U., Zschaler, S. & Wagner, G. (2006). Ontologies, Metamodels, and the Model-Driven Paradigm. Chapter in *Ontologies for Software Engineering and Technology*, *Coral Calero, Francisco Ruiz and Mario Piattini (Eds.)*. Springer-Verlag, 249-274.

[Atkinson & Kühne, 2003]

Atkinson, C. & Khüne, T. (2003). Model Driven Development: A Metamodeling Foundation. In Journal of *IEEE Software*, 20(5): 36-41.

[Baader et al., 2003]

Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D. & Patel-Schneider, P.F. (2003). *Description Logic Handbook*. Cambridge University Press.

[Bartsch et al., 2008]

Bartsch, C., Shwartz, L, Ward, C., Grabarnik, G. & Buco, M.J. (2008). Decomposition of IT service processes and alternative service identification using ontologies. *IEEE Network Operations and Management Symposium (NOMS)*, 714-717.

[Belhajjame & Brambilla, 2009]

Belhajjame, K. & Brambilla, M. (2009). Ontology-based Description and Discovery of Business Processes. In *Proceedings of the 10th Workshop on Business Process Modeling, Development, and Support (BPMDS) at CAiSE 2009*, Amsterdam. Springer LNBIP, 29: 85-98.

[Berztiss, 1999]

Berztiss, A.T. (1999). Domain analysis for business software systems. In Journal of *Information Systems*, 24(7): 555-568.

[Bézivin, 2004]

Bézivin, J. (2004). In Search of a Basic Principle for Model Driven Engineering. *CEPIS, UPGRADE, The European Journal for the Informatics Professional*, 2: 21-24.

[Bézivin, 2005]

Bézivin, J. (2005). Model Driven Engineering: An Emerging Technical Space. In *Proceedings of the International Summer School on Generative and Transformational Techniques in Software Engineering (GTTSE)*. Braga, Portugal. Vol. 4143 of LNCS, 36-64. Springer-Verlag Berlin Heidelberg.

[Bézivin, 2009]

Bézivin, J. (2009). Advances in Model Driven Engineering: Achievements and challenges. *Invited talk at JISBD'09: 14th Conference on Software Engineering and Databases*. Retrieved April, 2011, from http://www.emn.fr/z-info/atlanmod/index.php/Advances_in_Model_Driven_Engineering

[Bézivin & Gerbé, 2001]

Bézivin, J. & Gerbé, O. (2001). Towards a Precise Definition of the OMG/MDA Framework. In *Proceedings of 16th IEEE International Conference on Automated Software Engineering (ASE'01)*.

[Bialecki, 2001]

Bialecki, A. (2001). E-Commerce Integration Meta-Framework (ECIMF) Project Information Center at GetOpt.org. Retrieved April, 2011, from http://www.getopt.org/ecimf/contrib/onto/REA/

[Black et al., 2007]

Black, J., Draper, C., Lococo, T., Matar, F. & Ward, C. (2007). An integration model for organizing IT service management. *IBM Systems Journal* 46(3): 405-422.

[Booch et al., 2005]

Booch, G., Rumbaugh, J. & Jacobson, I. (2005). *Unified Modeling Language User Guide. 2nd Edition*. Addison-Wesley.

[Born et al., 2007]

Born, M., Dörr, F. & Weber, I. (2007). User-friendly Semantic Annotation in Business Process Modeling. In *Proceedings of the International Workshop on Human-Friendly Service Description, Discovery and Matchmaking (Hf-SDDM 2007) at the 8th International Conference on Web Information Systems Engineering (WISE 2007)*, Nancy, France, 260-271.

[Bunge, 1977]

Bunge, M. (1977). *Treatise on Basic Philosophy, Vol 3: Ontology I: The Furniture of the World*. Reidel, Dordrecht, Boston.

[Bunge, 1979]

Bunge, M. (1979). *Treatise on Basic Philosophy, Vol 4: Ontology II: A World of Systems*. Reidel, Dordrecht, Boston.

[Cardoso et al., 2004]

Cardoso J., Sheth, A., Miller, J., Arnold, J. & Kochut, K. (2004). Quality of Service for Workflows and Web Service Processes. *J. Web Semantics*, 1(3): 281–308.

[Chandrasekaran et al., 1999]

Chandrasekaran, B., Josephson J. & Benjamins, R. (1999). What are ontologies, and why do we need them. In Journal of *IEEE Intelligent Systems*, 14(1): 20-26.

[Clark, 1999]

Clark. J. (1999). XSL Transformations (XSLT) Version 1.0, *W3C Recommendation*. Retrieved April, 2011, from http://www.w3.org/TR/xslt

[Clark et al., 2008]

Clark, T., Sammut, P. & Willans, J. (2008). *Applied Metamodelling. A Foundation for Language Driven Development. Second Edition*. Ceteva, 2008. Retrieved April, 2011, from http://eprints.mdx.ac.uk/6060/1/Clark-Applied_Metamodelling_%28Second_Edition%29%5B1%5D.pdf

[CMMI, 2009]

CMMI. (2009). CMMI for Services, Version 1.2. *Software Engineering Institute*. Retrieved April, 2011, from http://www.sei.cmu.edu/cmmi/

[Cook, 2004]

Cook, S. (2004). Domain-Specific Modeling and Model Driven Architecture. *MDA Journal*, 1-10.

[Cook et al., 2007]

Steve Cook, S., Jones, G., Kent, S. & Cameron, A. (2007). *Domain-Specific Development with Visual Studio DSL Tools*. Addison-Wesley.

[Corcho et al., 2002]

Corcho, O., Fernández-López, M. & Gómez-Pérez, A. (2002). Methodologies, tools and languages for building ontologies. Where is the meeting point? In Journal of *Data and Knowledge Engineering*, 46(1): 41-64.

[Czarnecki & Helsen, 2006]

Czarnecki, K. & Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3): 621-645.

[DAML Services, 2003]

DAML Services. OWL-S 1.0. Retrieved April, 2011, from http://www.daml.org/services/owl-s/1.0/

[de Pablos et al., 2008]

de Pablos, C., López-Hermoso, J.J., Martín-Romo, S., Medina, S., Montero, A., Nájera, J.J. (2008). *Dirección y Gestión de los sistemas de información en la empresa: una visión integradora (2nd ed.)*. Madrid: ESIC Editorial.

[Decker et al., 2005]

Decker, S., Sintek, M., Billig, A., Henze, N., Dolog, P., Nejdl, W., Harth, A., Leicher, A., Busse, S., Ambite, J.L., Weathers, M., Neumann, G. & Zdun, U. (2005). TRIPLE - an RDF Rule Language with Context and Use Cases. W3C Workshop on *Rule Languages for Interoperability*, 27-28, Washington, D.C., USA. Retrieved April, 2011, from http://www.w3.org/2004/12/rules-ws/paper/98/

[DeMarco, 2009]

DeMarco, T. (2009). Software Engineering: An Idea Whose Time Has Come and Gone? In Journal of *IEEE Software*, 26(4): 96, 95.

[Devedžić, 2002]

Devedžić, V. (2002). Understanding Ontological Engineering. In Journal of *Communications of the ACM*, 45(4): 136-144.

[Di Francescomarino et al., 2011]

Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L. & Tonella, P. (2011). A framework for the collaborative specification of semantically annotated business processes. In Journal of *Software Maintenance and Evolution: Research and Practice*. Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.525.

[DKM website]

Data & Knowledge Management (DKM) website. BPMN Ontology. https://dkm.fbk.eu/index.php/BPMN_Ontology. Last visited: April 2011.

[Doux et al., 2009]

Doux, G., Jouault, F. & Bézivin, J. (2009). Transforming BPMN process models to BPEL process definitions with ATL. In *GraBaTs 2009, 5th International Workshop on Graph-Based Tools*.

[DSM Forum website]

DSM Forum website. http://www.dsmforum.org/. Last visited: April 2011.

[DuMoulin, 2007]

DuMoulin, T. (2007). ITIL V3: The Past & The Future. The Evolution Of Service Management Philosophy. *Pink Elephant*. Retrieved April, 2011, from

http://blogs.pinkelephant.com/images/uploads/pinklink/ITIL_v3_The_Past__The_Future.pdf

[Dunn et al., 2005]

Dunn, C.L., Owen, J. & Hollander, A.S. (2005). The rea enterprise ontology: value system and value chain modeling. Chapter in *Enterprise Information Systems: A Pattern-Based Approach, 3rd Edition*. McGraw-Hill/Irwin.

[Durak et al., 2006]

Durak, U., Oguztuzun, H. & Ider, S.K. (2006). An Ontology for Trajectory Simulation. *Conference on Winter Simulation, Monterey*, 1160-1167, California. SESSION: Modeling methodology b: ontology driven simulation. ISBN: 1-4244-0501-7.

[Eclipse - BPMN Modeler, 2011]

Eclipse – STP BPMN Modeler. (2011). BPMN object model. Retrieved April, 2011, from http://www.eclipse.org/bpmn/model/index.php

[Eriksson & Penker, 2000]

Eriksson, H.E. & Penker, M. (2000). *Business Modeling with UML. Business Patterns at Work*. John Wiley & Sons.

[Eshuis & Wieringa, 2001]

Eshuis, R. & Wieringa, R. (2001). An Execution Algorithm for UML Activity Graphs. In *Proceedings of the International Conference on the Unified Modeling Language (UML)*, Toronto, Canada. Springer Verlag.

[Favre, 2004]

Favre, J.M. (2004). Towards a Basic Theory to Model Model Driven Engineering. In *Workshop on Software Model Engineering, WISME 2004, joint event with UML2004*, Lisboa, Portugal.

[Ferrario & Guarino, 2009]

Ferrario, R. & Guarino, N. (2009). Towards an Ontological Foundation for Services Science. In *Domingue, J., Fensel, D., Traverso, P. (Eds.)*, *FIS 2008*. Vol. 5468 of LNCS, 152–169. Springer-Verlag Berlin Heidelberg.

[Ferris, 2008]

Ferris, K. (2008). Out of one silo and into another. From the Book: *IT Service Management Global Best Practices – Volume 1.* Van Haren Publishing.

[France & Rumpe, 2007]

France, R. & Bernhard Rumpe, B. (2007). Model-driven Development of Complex Software: A Research Roadmap. In *Future of Software Engineering (FOSE'07)*, 37-54.

[Freitas et al., 2008]

Freitas, J., Correia, A., Brito e Abreu, F. (2008). An Ontology for IT Services. In *Proceedings of the 13th Conference on Software Engineering and Databases*, Retrieved April, 2011, from http://ctp.di.fct.unl.pt/QUASAR/Resources/Papers/2008/freitas2008JISBD.pdf

[Garrido et al., 2007]

Garrido, J.L., Noguera, M., González, M., Hurtado, M.V., Rodríguez, M.L. (2007). Definition and use of Computation Independent Models in an MDA-based groupware development process. In Journal of *Science of Computer Programming*, 66(1): 25-43.

[Gašević et al., 2006]

Gašević, D., Djurić, D. & Deved, M. (2007). On metamodeling in megamodels. In *Engels, G. et al. (Eds.), MoDELS 2007*. Vol. 4735 of LNCS, 91-105. Springer-Verlag Berlin Heidelberg.

[Gašević et al., 2007]

Gašević, D., Kaviani, N. & Devedžić, V. (2006). Model Driven Architecture and Ontology Development. Springer-Verlag Berlin Heidelberg.

[Geerts & McCarthy, 1999]

Geerts, G.L. & McCarthy, W.E. (1999). An Accounting Object Infrastructure For Knowledge-Based Enterprise Models. *IEEE Intelligent Systems*, 14(4): 89-94, July/Aug. 1999. DOI10.1109/5254.784089.

[Geerts & McCarthy, 2000]

Geerts, G.L. & McCarthy, W.E. (2000). The Ontological Foundation of REA Enterprise Information Systems. Paper presented at the *American Accounting Association Conference* Philadelphia, PA. Retrieved April, 2011, from https://www.msu.edu/user/mccarth4/Alabama.doc

[Ghedini & Gostinski, 2008]

Ghedini, C. & Gostinski, R. (2008). A Methodological Framework for Business-IT Alignment. In *Proceedings of the Third IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM)*, 1-10.

[Goeken & Alter, 2009]

Goeken, M. & Alter, S. (2009) Towards Conceptual Metamodeling of IT Governance Frameworks. Approach – Use – Benefits. In Proceedings of the 42nd Hawaii International Conference on System Sciences.

[Goknil & Topaloglu, 2005]

Goknil, A., Topaloglu, Y. (2005). Ontological Perspective in Metamodeling for Model Transformations. *Metainformatics, ACM International Conference Proceeding Series*, 214(7), Esbjerg, Denmark. ISBN: 978-1-59593-719-3.

[Graham et al., 1997]

Graham, I., Henderson-Sellers, B. & Younessi, H. (1997). *The OPEN Process Specification*. Addison Wesley.

[Graupner et al., 2009]

Graupner, S., Motahari-Nezhad, H.R., Singhal, S. & Basu, S. (2009). Making processes from best practice frameworks actionable. In *Proceedings of the 13th Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, 25-34.

[Green et al., 2005]

Green, P.F., Rosemann, M. & Indulska, M. (2005). Ontological Evaluation of Enterprise Systems Interoperability Using ebXML. In Journal of *IEEE Transactions on Knowledge and Data Engineering*, 17(5): 713-725.

[Grimm & Hitzler, 2008]

Grimm, S. & Hitzler, P. (2008). Semantic Matchmaking of Web Resources with Local Closed-World Reasoning. In *International Journal of e-Commerce*, 12 (2): 89-126.

[Gruber, 1991]

Gruber, T. (1991). The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. In *Allen, J.A., Fikes, R., Sandewall, E. (Eds.), Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, 601-602. Morgan Kaufmann, Cambridge.

[Gruber, 1995]

Gruber, T.R. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In *International Journal of Human-Computer Studies*, 43(5-6): 907-928.

[Guarino, 1998]

Guarino, N. (1998). Formal Ontology in Information Systems. In *Proceedings of FOIS'98*, Trento, Italy. IOS Press, Amsterdam.

[Guttman & Parodi, 2007]

Guttman, M and Parodi, J. (2007). *Real-Life MDA. Solving Business Problems with Model Driven Architecture*. Morgan Kaufmann Publishers.

[Hammer & Champy, 1993]

Hammer, M. & Champy, J. (1993). *Reengineering the Corporation*. Harper Business, New York.

[Havey, 2005]

Havey, M. (2005). *Essential Business Process Modeling*. O'Reilly.

[Henderson-Sellers, 2011]

Henderson-Sellers, B. (2011). Bridging Metamodels and Ontologies in Software Engineering. In *Journal of Systems and Software*, 84(2): 169-340.

[Horrocks et al., 2004]

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. & Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member Submission*. Retrieved April, 2011, from http://www.w3.org/Submission/SWRL/

[Humphrey, 1989]

Humphrey, W.S. (1989). *Managing the Software Process*. Addison-Wesley.

[ISACA, 2007]

ISACA (2007). COBIT 4.1. Retrieved April, 2011, from http://www.isaca.org/Knowledge-Center/cobit/Documents/COBIT%204.1.pdf

[ISACA, 2008]

ISACA. (2008). COBIT® Mapping: Mapping of ITIL v3 With COBIT® 4.1. Retrieved April, 2011, from http://www.itsm.hr/baza%20znanja/Mapping%20ITILV3%20COBIT41.pdf

[ISACA, 2009]

ISACA. (2009). Transforming Enterprise IT. Retrieved April, 2011, from http://www.isaca.org/Knowledge-Center/cobit/Documents/Forms/AllItems.aspx

[ISACA website]

ISACA website. COBIT Framework for IT Governance and Control. Retrieved April, 2011, from http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx

[ISO, 2005a]

ISO. (2005a). ISO 9000:2005 Quality management systems – Fundamentals and vocabulary. Retrieved April, 2011, from http://www.iso.org/iso/home.html

[ISO, 2005b]

ISO. (2005b). ISO/IEC 19502:2005 Information technology – Meta Object Facility (MOF). Retrieved April, 2011, from http://www.omg.org/spec/MOF/ISO/19502/PDF/

[ISO/IEC, 2005a]

ISO/IEC. (2005a). ISO/IEC 20000-1:2005 Information Technology – Service Management – Part 1: Specification. Retrieved April, 2011, from http://www.iso.org/iso/home.html

[ISO/IEC, 2005b]

ISO/IEC. (2005b). ISO/IEC 20000-2:2005 Information Technology – Service Management – Part 2: Code of Practice. Retrieved April, 2011, from http://www.iso.org/iso/home.html

[itSMF, 2006]

itSMF-NL. (2006). *Frameworks for IT Management*. ITSM Library, Van Haren Publishing.

[itSMF, 2007a]

itSMF International. (2007a). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

[itSMF, 2007b]

itSMF International. (2007b). ITIL V3: Glossary of Terms and Definitions. Version to Workload. Retrieved April, 2011, from http://www.itsmfi.org/content/itil-v3-glossary-acronmys-pdf

[itSMF, 2008]

itSMF International. (2008). *IT Service Management Global Best Practices – Volume 1.* Van Haren Publishing.

[ITIL website]

ITIL website. http://www.itil-officialsite.com/home/home.asp. Last visited: April 2011

[Jacobson et al., 1995]

Jacobson, I., Ericsson, M. & Jacobson, A. (1995). *The Object Advantage. Business Process Reengineering With Object Technology*. Addison-Wesley.

[Jensen, 1996]

Jensen, K. (1996). *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1 (Second Edition)*. Springer.

[Jerphanion & Kristelin, 2008]

Jerphanion, S. & Kristelin, I. (2008). Applying the five disciplines of the learning organization to ITIL. From the Book: *IT Service Management Global Best Practices – Volume 1.* Van Haren Publishing.

[Jones, 2005]

Jones, S. (2005). Toward an acceptable definition of service. In Journal of *IEEE Software*, 22(3): 87-93.

[Jordan & Evdemon, 2007]

Jordan D. & Evdemon, J. (2007). Web Services Process Execution Language Version 2.0. *Committee Specification. OASIS WS-BPEL TC*. Retrieved April, 2011, from http://www.oasis-open.org/committees/download.php/22475/wsbpel-v2.0-CS01.pdf

[Joukhadar & Al-Maghout, 2008]

Joukhadar, A. & Al-Maghout, H. (2008). Improving Agility in Business Applications using Ontology Based Multilingual Understanding of Natural Business Rules. In *International Conference on Information and Communication Technologies: From Theory to Applications* (ICTTA 2008), 1-8, Digital Object Identifier 10.1109/ICTTA. 2008.4530336.

[Kaplan & Norton, 1992]

Kaplan, R. & Norton D. (1992, January-February). The Balanced Scorecard – Measures that Drive Performance. *Harvard Business Review*, 71-79.

[KBSI, 1994]

KBSI. (1994). IDEF5 Method Report. *Information Integration for Concurrent Engineering (IICE) project, F33615-90-C-0012*. Retrieved April, 2011, from http://www.idef.com/pdf/Idef5.pdf

[Kent, 2002]

Kent, S. (2002). Model Driven Engineering. In *Proceedings of the Third International Conference on Integrated Formal Methods*, 286-298.

[Kleppe et al., 2003]

Kleppe, A, Warmer, J. & Bast, W. (2003). *MDA Explained. The Model Driven Architecture: Practice and Promise*. Addison-Wesley.

[Knorr et al., 2011]

Knorr , M., Alferes, J.J. & Hitzler, P. (2011). Local Closed World Reasoning with Description Logics under the Well-Founded Semantics. In Journal of *Artificial Intelligence*. Retrieved April, 2011, from http://knoesis.wright.edu/faculty/pascal/resources/publications/mknftheo.pdf

[Kurtev et al., 2006]

Kurtev, I., Bézivin, J., Jouault, F. & Valduriez, P. (2006). Model-based DSL Frameworks. Dynamic Languages Symposium. *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, Portland, Oregon, USA. SESSION: OOPSLA onward! track chair's welcome, 602-616. ISBN: 1-59593-491-X.

[Kühne, 2006]

Kühne, T. (2006). Matters of (Meta-)Modeling. In Journal of *Software and Systems Modeling* (SoSyM), 5(4): 369-385.

[Lenat, 1995]

Lenat, D.B. (1995). Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11): 33-38.

[Li et al., 2011]

Li, D. Li, X & Stolz, V. (2011). QVT-based model transformation using XSLT. *ACM SIGSOFT Software Engineering Notes* 36(1).

[Liu & Zhu, 2009]

Liu, X.F. & Zhu, L. (2009). Design of SOA Based Web Service Systems Using QFD for Satisfaction of Quality of Service Requirements. In *Proceedings of the 2009 IEEE International Conference on Web Services*, 567-574.

[Marshall, 2000]

Marshall, C. (2000). *Enterprise Modeling with UML. Designing Successful Software through Business Analysis*. Addison-Wesley.

[McCarthy, 1982]

McCarthy, W.E. (1982). The REA Accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review*, 57(3):554–578.

[Mellor et al., 2004]

Mellor, S.J., Scott, K., Uhl, A. & Weise, D. (2004). *MDA Distilled. Principles of Model-Driven Architecture*. Addison-Wesley.

[Mili et al., 2010]

Mili H., Guy, T., Guitta, B., Jaoude, G.B., Lefebvre, É., Elabed, L. & El Boussaidi G. (2010). *Business Process Modelling Languages: Sorting Through the Alphabet Soup*. ACM Computing Surveys, 43(1), Article 4.

[Mizoguchi & Ikeda, 1996]

Mizoguchi, R. & Ikeda, M. (1996). Towards Ontology Engineering. *Technical Report AI-TR-96-1, I.S.I.R.*, Osaka University.

[Nagarajan et al., 2006]

Nagarajan, M., Verma, K., Sheth, A., Miller, J. & Lathem, J. (2006). Semantic Interoperability of Web Services: Challenges and Experiences. In *Proceedings of the 4th IEEE Int'l Conf. Web Services, IEEE CS Press*, 373–382.

[Nextel, 2010]

Nextel S.A. (2010). *ISO/IEC 20000 para pymes. Cómo implantar un sistema de gestión de los servicios de tecnologías de la información*. AENOR ediciones.

[Niles & Pease, 2001]

Niles, I. & Pease, A. (2001). Towards a standard upper ontology. In *Proceedings of the 2nd international conference on formal ontology in information systems (FOIS-2001)*, 2–9.

[Nurmilaakso, 2008]

Nurmilaakso, J.M. (2008). EDI, XML and e-business frameworks: A survey. In Journal of *Computers in industry*, 59(4): 370-379.

[OASIS, 2001]

OASIS. (2001). Business process and business information analysis overview. version 1.0 (ebXML). Retrieved April, 2011, from http://www.ebxml.org/specs/bpOVER.pdf

[O'Connor & Das, 2009]

O'Connor, M.J. & Das, A. (2009). SQWRL: a Query Language for OWL. *OWL: Experiences and Directions (OWLED 2009), Fifth International Workshop*, Chantilly, VA.

[OGC, 2007a]

OGC. (2007a). *ITIL Service Design*. The Stationery Office (TSO). London.

[OGC, 2007b]

OGC. (2007b). ITIL Service Improvement. The Stationery Office (TSO). London.

[OGC, 2007c]

OGC. (2007c). *ITIL Service Operation*. The Stationery Office (TSO). London.

[OGC, 2007d]

OGC. (2007d). *The Official Introduction to the ITIL Service Lifecycle*. The Stationery Office (TSO). London.

[Oldham et al., 2006]

Oldham, N., Verma, K., Sheth, A., Hakimpour, F. (2006). Semantic WS-Agreement Partner Selection. In *Proceedings of the15th Int'l World Wide Web Conf. (WWW2006), ACM Press*, 697–706.

[Olivé, 2007]

Olivé, A. (2007). *Conceptual Modeling of Information Systems*. Springer-Verlag Berlin Heidelberg.

[OMG, 2003]

OMG. (2003). MDA Guide, Version 1.0.1. Retrieved April, 2011, from http://www.omg.org/cgi-bin/doc?omg/03-06-01

[OMG, 2006a]

OMG. (2006a). Business Process Modeling Notation (BPMN), Version 1.0. Retrieved April, 2011, from http://bpmn.org/Documents/OMG_Final_Adopted_BPMN_1-0_Spec_06-02-01.pdf

[OMG, 2006b]

OMG. (2006b). Meta Object Facility (MOF). Core Specification, Version 2.0. Retrieved April, 2011, from http://www.omg.org/spec/MOF/2.0/PDF/

[OMG, 2007]

OMG. (2007). MOF 2.0/XMI Mapping, Version 2.1.1 Retrieved April, 2011, from http://www.omg.org/spec/XMI/2.1.1/PDF/index.htm

[OMG, 2008]

OMG. (2008). Business Process Definition Metamodel (BPDM), Version 1.0. Retrieved April, 2011, from http://www.omg.org/spec/BPDM/1.0/

[OMG, 2009]

OMG. (2009). Ontology Definition Metamodel, Version 1.0. Retrieved April, 2011, from http://www.omg.org/spec/ODM/1.0/PDF/

[OMG, 2010a]

OMG. (2010a). Business Process Model and Notation (BPMN), Version 2.0 (Beta 2). Retrieved April, 2011, from http://www.omg.org/cgi-bin/doc?dtc/10-06-04

[OMG, 2010b]

OMG. (2010b). Object Constraint Language (OCL), Version 2.2. Retrieved April, 2011, from http://www.omg.org/spec/OCL/2.2/PDF/

[OMG, 2010c]

OMG. (2010c). Unified Modeling Language (UML), Superstructure, Version 2.3. Retrieved April, 2011, from http://www.omg.org/spec/UML/2.3/Superstructure/PDF/

[OMG/BPMI-BPMN website]

OMG/BPMI BPMN website. Last visited: April 2011.

[Ould, 1995]

Ould, M.A. (1995). *Business Processes: Modelling and Analysis for Re-engineering and Improvement*.Wiley, New York.

[OZONA website]

Ozona website. http://www.ozona.es/. Last visited: April 2011.

[Pahl, 2007]

Pahl, C. (2007). Semantic model-driven architecting of service-based software systems. In Journal of *Information and Software Technology*, 49(8): 838-850.

[Paschke & Bichler, 2008]

Paschke, A. & Bichler, M. (2008). Knowledge representation concepts for automated SLA management. In Journal of *Decision Support Systems*, 46(1): 187-205.

[Pastor & Molina, 2007]

Pastor, O. & Molina, J.C. (2007). *Model-Driven Architecture in Practice. A Software Production Environment Based on Conceptual Modeling*. Springer-Verlag.

[Plesums, 2002]

Plesums, C. A. (2002). Introduction to Workflow. *A vendor-independent tutorial*. Retrieved April, 2011, from http://www.plesums.com/image/introworkflow.html

[PMI website]

Project Management Institute (PMI). (2010). PMBOK Guide and Standards. Retrieved April, 2011, from http://www.pmi.org/PMBOK-Guide-and-Standards.aspx

[Prieto & Lozano-Tello, 2009]

Prieto, A.E. & Lozano-Tello, A. (2009). Use of Ontologies as Representation Support of Workflows Oriented to Administrative Management. In Journal of *Network and Systems Management*, 17(3): 309-325.

[Raistrick et al., 2004]

Raistrick, C., Francis, P., Wright, J., Carter, C. & Wilkie, I. (2004). *Model Driven Architecture with Executable UML*. Cambridge University Press.

[Recker et al., 2009]

Recker, J.C., Rosemann, M., Indulska, M. & Green, P. (2009). Business process modeling: A comparative analysis. In *Journal of the Association for Information Systems*, 10(4): 333-363.

[Ruiz & Hilera, 2006]

Ruiz, F. & Hilera, J.R. (2006). Using Ontologies in Software Engineering and Technology. Chapter in *Ontologies in Software Engineering and Software Technology*, *Calero, C., Ruiz, F. and Piattini, M (Eds.)*. Springer-Verlag Berlin Heidelberg, 62-119.

[Savvas & Bassiliades, 2009]

Savvas, I. & Bassiliades, N. (2009). A process-oriented ontology-based knowledge management system for facilitating operational procedures in public administration. In Journal of *Expert Systems with Applications*, 36: 4467-4478.

[Scheer, 2000]

Scheer, A.W. (2000). *ARIS- Business Process Modeling, 3rd edition*. Springer-Verlag New York.

[Schmidt, 2006]

Schmidt, D.C. (2006). Model-Driven Engineering. In Journal of *IEEE Computer*, 39(2): 25-31.

[Sedbrook & Newmark, 2008]

Sedbrook, T. & Newmark, R.I. (2008). Automatin REA Policy Level Specifications with Semantic Web Technologies. In *Journal of Information Systems*, 22(2): 249-277.

[Seidewitz, 2003]

Seidewitz, E. (2003). What Models Mean. In Journal of *IEEE Software*, 20(5): 26-32.

[Sendall & Kozaczynski, 2003]

Sendall, S. & Kozaczynski, W. (2003). Model Transformation: the Heart and Soul of Model-Driven Software Development. In Journal of *IEEE Software*, 42-51.

[Shangguan et al., 2007]

Shangguan, Z., Gao, Z., Zhu, K. (2007). Ontology-Based Process Modeling Using eTOM and ITIL. In *IFIP International Federation for Information Processing*, Vol. 255, *Research and Practical Issues of Enterprise Information Systems II*, 2: 1001-1010.

[Sharp & McDermott, 2001]

Sharp, A. & McDermott, P. (2001). *Workflow Modeling. Tools for Process Improvement and Application Development*. Artech House.

[Sicilia & Mora, 2010]

Sicilia, M.A. & Mora, M. (2010). On Using the REA Enterprise Ontology as a Foundation for Service System Representation. In *Proceedings of ONTOSE 2010*, LNBIP 62: 135-147. Springer-Verlag Berlin Heidelberg.

[Silva, 1985]

Silva, M. (1985). *Las Redes de Petri: en la Automática y la Informática*. Editorial AC.

[Silva Parreiras & Staab, 2010]

Silva Parreiras, F. & Staab, S. (2010). Using Ontologies with UML Class-based Modeling: The TwoUse Approach. In Journal of *Data & Knowledge Engineering*.

[Smith, 2008]

Smith, D. (2008). How to implement metrics for IT service management. From the Book: *IT Service Management Global Best Practices – Volume 1.* Van Haren Publishing.

[Smith et al., 2004]

Smith, M.K., Welty, C. & McGuinness, D.L. (2004). OWL Web Ontology Language Guide. *W3C Recommendation*. Retrieved April, 2011, from http://www.w3.org/ TR/owl-guide/

[Sommerville, 2010]

Sommerville, I. (2010). *Software Engineering. 9th edition*. Addison Wesley.

[Stahl & Völter, 2006]

Stahl, T. & Völter, M. (2006). *Model-Driven Software Development. Technology, Engineering, Management*. John Wiley & Sons.

[Steinberg, 2006]

Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

[Talantikite et al., 2009]

Talantikite, H.N., Aissani, D. & Boudjlida, N. (2009). Semantic annotations for web services discovery and composition. In Journal of *Computer Standards & Interfaces*, 31(6): 1108-1117.

[Tautz & Wangenheim, 1998]

Tautz, C. & Von Wangenheim, C. (1998). REFSENO: A Representation Formalism for Software Engineering Ontologies. Version 1.1. *015.98/E. Fraunhofer IESE*.

[Telefónica, 2010]

Telefónica. (2010). ISO/IEC 20000. Guía completa de aplicación para la gestión de los servicios de tecnologías de la información. AENOR ediciones.

[Thomas & Fellmann, 2009]

Thomas, O., Fellmann, M. (2009). Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes. In Journal of *Business & Information Systems Engineering*, 1(6): 438-451.

[UN/CEFACT and OASIS, 2001]

UN/CEFACT and OASIS. (2001). ebXML Business Process Specification Schema. v1.01. Retrieved April, 2011, from http://www.ebxml.org/specs/ebBPSS.pdf.

[Uschold & Grüninger, 1996]

Uschold, M. & Grüninger, M. (1996). Ontologies: Principles, Methods, and Applications. In *Knowledge Engineering Review (KER)*, 11(2): 93-113.

[Verma et al., 2005]

Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S. & Miller, J. (2005). Meteor-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *J. Information Technology and Management*, 6(1): 17–39.

[Verma & Sheth, 2007]

Verma K. & Sheth A. (2007). Semantically Annotating a Web Service. In Journal of *IEEE Internet Computing*, 11(2): 83-85.

[Vicente-Chicote & Alonso, 2007]

Vicente-Chicote, C. & Alonso D. (2007). Tutorial: Herramientas Eclipse para Desarrollo de Software Dirigido por Modelos. *XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2007)*. Actas de Talleres y Tutoriales de las Jornadas de Ingeniería del Software y Bases de Datos, 1(8). ISSN: 1988-3455.

[W3C, 2004a]

W3C. (2004). OWL-S: Semantic Markup for Web Services. *W3C Member Submission*. Retrieved April, 2011, from http://www.w3.org/Submission/OWL-S/

[Wade & Hulland, 2004]

Wade, M. & Hulland, J. (2004). Review: The Resource-Based View and Information Systems Research: Review, Extension, and Suggestions for Future Research. *MIS Quarterly*, 28(1): 107-142.

[Walter & Ebert, 2009]

Walter, T. & Ebert, J. (2009). Combining ontology-enriched Domain-Specific Languages. In Proceedings of the 2nd Workshop on Transforming and Weaving Ontologies in Model Driven Engineering (TWOMDE).

[Walter et al., 2009]

Walter, T., Silva Parreriras, F. & Staab, S. (2009). OntoDSL: An ontology-based framework for domain-specific languages. In *Proceedings of the 12th International Conference on Model Driven Engineering Languagesand Systems (MODELS 2009)*. Vol. 5795 of LNCS, 408-422. Springer-Verlag Berlin Heidelberg.

[Wand, 1996]

Wand, Y. (1996). Ontology as a foundation for meta-modeling and method engineering. In Journal of *Information and Software Technology*, 38(4): 281-287.

[Wand & Weber, 2003]

Wand, Y. & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. In *Journal of Information Systems* 3(4): 217–237.

[Watson, 2008]

Watson, A. (2008). UML vs DSLs: A false dichotomy. *Whitepaper, Object Management Group*. Retrieved April, 2011, from http://www.omg.org/cgi-bin/doc?omg/08-09-03

[WfMC, 1999]

WfMC. (1999). Workflow Management Coalition. Terminology and Glossary English. *WfMC-TC-1011 v3. Technical Report*. Retrieved April, 2011, from http://www.wfmc.org/Glossaries-FAQs/

[Wolf & Harmon, 2010]

Wolf, C. & Harmon, P. (2010). The State of Business Process Management 2010. BPTrends report. Retrieved April, 2011, from http://www.bptrends.com/members_surveys/deliver.cfm?report_id=1004&target=2009%20BPTrends%20State%20of%20Market%20Rept%20-FINAL%20PDF%20CAP%202-1-10.pdf&return=surveys_landing.cfm

[Živković et al., 2008]

Živković, S., Murzek, M., Kühn, H. (2008). Bringing Ontology Awareness into Model Driven Engineering Platforms. *Workshop on Transforming and Weaving Ontologies in Model Driven Engineering (TWOMDE 2008) at MoDELS 2008*, Toulouse, France. Vol. 395 of CEUR Workshop Proceedings, 47-54. CEUR-WS.org.

Appendix I

ITSM Ontology Concepts

This appendix summarizes the OWL Ontology that we have defined for the IT service management domain implemented in this thesis.

Classes

Class: CI

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO).

Description: An *itil:CI* is a configuration item that represents an asset, service component or other item that is, or will be, under the control of *itil:ServiceAsset_and_ConfigurationManagement* process. The *itil:CI*(s) may vary widely in complexity, size and type, ranging from an entire service or system including all hardware, software, documentation and support staff to a single software module or a minor hardware component. The *itil:CI*(s) may be grouped and managed together. For example, a set of components may be grouped into a release. The *itil:CI*(s) should be selected using established selection criteria, grouped, classified and identified in such a way that they are manageable and traceable throughout the *itil:ServiceLifecycle*.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Transition*, p. 122-123 and p. 373 (Configuration Item definition).

Object Properties: *itil:hasConfigurationRecord*

Datatype Properties: *itil:ciDescription* and *itil:ciName*

Class: ConfigurationRecord

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:ConfigurationRecord* is a record that contains the details of an *itil:CI*. Each *itil:ConfigurationRecord* documents the *itil:Lifecycle* of a single *itil:CI*. The *itil:ConfigurationRecord*(s) are stored in a *Configuration Management Database* (CMDB).

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Transition*, p. 145-146. *ITIL V3: Glossary of Terms and Definitions* (Configuration Record definition).

Object Properties: none

Datatype Properties: none

Class: Lifecycle

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:Lifecycle* represents the various stages in the life of an *itil:ITService*, *itil:CI*, *itil:Incident*, *itil:Problem*, *itil:Change*, etc. The *itil:Lifecycle* defines the categories for status and the status transitions that are permitted. For example:

- The lifecycle of an application includes requirements, design, build, deploy, operate, and optimize.
- The expanded incident lifecycle includes detect, respond, diagnose, repair, recover, and restore.
- The lifecycle of a server may include: ordered, received, in test, live, disposed, etc.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 355-356 (Lifecycle definition).

Object Properties: *itil:hasStage*

Datatype Properties: *itil:lifecycleDescription* and *itil:lifecycleName*

Class: ServiceLifecycle

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The architecture of the *ITIL V3 Service Management Model* is based on a service lifecycle. The *itil:ServiceDesign*, *itil:ServiceTransition* and *itil:ServiceOperation* stages are progressive phases of the *itil:ServiceLifecycle* class that represent change and transformation. The *itil:ServiceStrategy* stage represents policies and objectives. Finally, the Continual Service Improvement (CSI) stage, *itil:CSI*, represents learning and improvement.

Generalization: *itil:Lifecycle*

Relation to ITIL: *ITIL Service Strategy*, p. 45.

Object Properties: *itil:hasServiceStage* (subproperty of *itil:hasStage*), *itil:inITService* and inherited from *itil:Lifecycle*

Datatype Properties: Inherited from *itil:Lifecycle*

Class: Stage

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:Stage* represents any phase of a lifecycle. For example: the status shows the current stage in the lifecycle of the associated CI, incident, problem, etc.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 368 (Status definition).

Object Properties: *itil:inLifecycle*

Datatype Properties: *itil:stageDescription* and *itil:stageName*

Class: ServiceStage

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO); OGC. (2007). *The Official Introduction to the ITIL Service Lifecycle*. The Stationery Office (TSO). London.

Description: An *itil:ServiceStage* represents any phase of an *itil:ServiceLifecycle*. For example: Service Operation is a service stage in the lifecycle of an IT service. The strength of the ITIL service management model rests upon continual feedback throughout each *itil:ServiceStage* of an *itil:ServiceLifecycle*. This feedback ensures that service optimization is managed from a business perspective and is measured in terms of the value business at any point in time through the *itil:ServiceLifecycle*. The *itil:ServiceLifecycle* is non-linear in design. At every point in the *itil:ServiceLifecycle*, feedback flows between each *itil:ServiceStage* of an *itil:ServiceLifecycle* which drive decisions about the need for minor course corrections of major service improvement initiatives.

Generalization: *itil:Stage*

Relation to ITIL: *ITIL Service Strategy*, p. 366 (Service Operation definition). The Official Introduction to the ITIL Service Lifecycle, p. 21-22. According to ITIL V3, Service Strategy, Service Design, Service Transition, Service Operation and Continual Service Improvement (CSI) are the different phases of the lifecycle of an IT Service.

Object Properties: *itil:hasProcess*, *itil:inServiceLifecycle* (subproperty of *itil:inLifecycle*), *itil:isFeedback*, *itil:receivesFeedback* and inherited from *itil:Stage*

Datatype Properties: *itil:serviceStageObjective*, *itil:serviceStageScope*, *itil:serviceStageValueToBusiness* and inherited from *itil:Stage*

Class: ServiceStrategy

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The ITIL V3 Service Strategy phase establishes an overall Strategy for IT services and for ITSM. Topics covered in *itil:ServiceStrategy* include the development of markets, internal and external, service assets, service Catalog, and implementation of strategy through the service lifecycle. Financial Management, Service portfolio management, Organizational development, and Strategic risks are among other major topics.

The *itil:ServiceStrategy* is about ensuring that IT service providers are in a position to handle the costs and risks associated with their service portfolios, and are set up not just for operational effectiveness but also for distinctive performance. Decisions made with respect to *itil:ServiceStrategy* have far-reaching consequences including those with delayed effect.

Generalization: *itil:ServiceStage*

Relation to ITIL: *ITIL Service Strategy*, p. 25 and p. 367 (Service Strategy definition).

Object Properties: *itil:hasStrategyProcess* (subproperty of *itil:hasProcess*) and inherited from *itil:ServiceStage*

Datatype Properties: Inherited from *itil:ServiceStage*

Class: ServiceDesign

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:ServiceDesign* is a stage in the lifecycle of an IT service. The ITIL V3 Service Design phase includes the design and development of services and service management processes. It covers design principles and methods for converting strategic objectives into portfolios of services and service assets. The scope of *itil:ServiceDesign* is not limited to new services. It includes the changes and improvements necessary to increase or maintain value to customers over the lifecycle of services, the continuity of services, achievement of service levels, and conformance to standards and regulations. It guides IT service providers on how to develop design capabilities for service management.

Generalization: *itil:ServiceStage*

Relation to ITIL: *ITIL Service Strategy*, p. 25 and p. 365 (Service Design definition).

Object Properties: *itil:hasDesignProcess* (subproperty of *itil:hasProcess*) and inherited from *itil:ServiceStage*

Datatype Properties: Inherited from *itil:ServiceStage*

Class: ServiceTransition

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:ServiceTransition* is a stage in the lifecycle of an IT service. The ITIL V3 Service Transition phase includes the development and improvement of capabilities for transitioning new and changed services into operations. The *itil:ServiceTransition* shows how the requirements of *itil:ServiceStrategy* encoded in *itil:ServiceDesign* are effectively realized in *itil:ServiceOperation* while controlling the risks of failure and disruption. Also, *itil:ServiceTransition* includes the management of the complexity related to changes to services and service management processes, preventing undesired consequences while allowing for innovation.

Generalization: *itil:ServiceStage*

Relation to ITIL: *ITIL Service Strategy*, p. 25-26 and p. 367 (Service Transition definition).

Object Properties: *itil:hasTransitionProcess* (subproperty of *itil:hasProcess*) and inherited from *itil:ServiceStage*

Datatype Properties: Inherited from *itil:ServiceStage*

Class: ServiceOperation

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:ServiceOperation* is a stage in the lifecycle of an IT service. The ITIL V3 Service Operation phase includes the management of service operations using two major control perspectives: reactive and proactive. The *itil:ServiceOperation* enables service providers to achieve effectiveness and efficiency in the delivery and support of services so as to ensure value for the customer and the service provider. Strategic objectives are ultimately realized through service operations, therefore making it a critical capability. Also, *itil:ServiceOperation* can maintain stability in service operations, allowing for changes in design, scale, scope and service levels. With *itil:ServiceOperation*, IT service providers can make better decisions in areas such as managing the availability of services, controlling demand, optimizing capacity utilization, scheduling of operations and fixing problems.

Generalization: *itil:ServiceStage*

Relation to ITIL: *ITIL Service Strategy*, p. 26 and p. 366 (Service Transition definition).

Object Properties: *itil:hasOperationProcess* (subproperty of *itil:hasProcess*) and inherited from *itil:ServiceStage*

Datatype Properties: Inherited from *itil:ServiceStage*

Class: CSI

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The ITIL V3 Continual Service Improvement (CSI) is a stage in the lifecycle of an IT service. The *itil:CSI* phase is responsible for managing improvements to IT service management processes and IT services. The performance of the IT service provider is continually measured and improvements are made to processes, IT services and IT infrastructure in order to increase efficiency, effectiveness, and cost effectiveness. Also, *itil:CSI* combines principles, practices, and methods from quality management, change management and capability improvement. IT service providers learn to realize incremental and large-scale improvements in service quality, operational efficiency and business continuity. The *itil:CSI* allows IT service providers to link improvement efforts and outcomes with service strategy, design, and transition. A closed-loop feedback system, based on the Plan–Do–Check–Act (PDCA) model specified in ISO/IEC 20000, is established and capable of receiving inputs for change from any planning perspective.

Generalization: *itil:ServiceStage*

Relation to ITIL: *ITIL Service Strategy*, p. 26 and p. 347 (Continual Service Improvement definition).

Object Properties: *itil:hasCSIProcess* (subproperty of *itil:hasProcess*) and inherited from *itil:ServiceStage*

Datatype Properties: Inherited from *itil:ServiceStage*

Class: Specification

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An oc:Specification is an abstract work that constitutes a description of the properties of a *oc:Situation* or a *oc:SomethingExisting*, and sometimes even entire collections of such things. Things are made, bought, and searched for according to specifications, which can be instantiated as printed instructions or as diagrams. This collection is modally neutral with regard to the descriptive character of its instances. Thus, it includes descriptions of how things are, were, should be, must be, etc.

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies relating our ITIL-based service management data to other data expressed on the Semantic Web (independent of a particular domain), we use the OpenCyc concept *oc:Specification* to classify the ITIL concepts that are considered specifications, such as *itil:Process* (subclassing from *oc:ProgramSpecification*). In our modeling approach for ITSMSs, *oc:Specifications* are composed of *itil:Activity* that describe the specification in terms of workflows enriched with ontological knowledge.

Object Properties: *itil:specifiesActivity*

Datatype Properties: *itil:specDescription* and *itil:specName*

Class: ProgramSpecification

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:ProgramSpecification* is a specialization of *oc:Specification*. Each instance of this collection is not a computer program itself (i.e. lines of code), but an abstract characterization of how a program should behave. For example, a sorting program can be specified by requiring that the program's output be a list of the same elements as the input such that no element follows an element that is greater than it. A notable example of a *oc:ProgramSpecification* is UNIX, which is not (contrary to popular belief) an operating system per se, but a specification to which many different operating systems (instances of oc:UnixOS) conform. Note that instances of *oc:ProgramSpecification* do not necessarily specify single, discrete programs, thus many of the internet's Request For Comments (RFC) protocol-establishing documents fall into this collection.

Generalization: *oc:Specification*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:ProgramSpecification* to specify the behavior of the different business activities in the ITIL processes.

Object Properties: Inherited from *oc:Specification*

Datatype Properties: Inherited from *oc:Specification*

Class: Process

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:Process* is a structured set of activities designed to accomplish a specific objective. It takes one or more defined inputs and turns them into defined outputs. An *itil:Process* has an owner and it may include any of the roles, responsibilities, tools and management controls required to reliably deliver the outputs. Also, an *itil:Process* may define policies, standards, guidelines, activities, and work instructions if they are needed.

Generalization: *oc:ProgramSpecification*

Relation to ITIL: *ITIL Service Strategy*, p. 360-361 (Process definition). We use the *itil:Process* class to model the different processes that are part of each stage in the lifecycle of an IT Service. According to the ITIL framework, we have grouped the different processes into the next categories (subclasses): *itil:StrategyProcess*, *itil:DesignProcess*, *itil:TransitionProcess*, *itil:OperationProcess* and *itil:CSIProcess*. Since our objective is to implement the ITIL processes, we consider them a subclass of *oc:ProgramSpecification*.

Object Properties: *itil:hasInterfaceRelation*, *itil:inServiceStage*, *itil:managesEvent*, *itil:measuredBy*, *itil:processOwner* and inherited from *oc:ProgramSpecification*

Datatype Properties: *itil:processChallenge*, *itil:processInput*, *itil:processName*, *itil:processObjective*, *itil:processOutput*, *itil:processRisk*, *itil:processScope*, *itil:processTechnology*, *itil:processValueToBusiness* and inherited from *oc:ProgramSpecification*

Class: InterfaceRelation

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: Each *itil:Process* may have interfaces to other *itil:Process*(s) that are part of the same or other service management lifecycle stages. That is, this *itil:Process* will be supported and executed by *itil:*Process(s) during the same or other phases of the service management lifecycle, but the *itil:Process* is driven by the phase in which it is part of. For example, interfaces to the *itil:IncidentManagement* process include:

- *itil:ProblemManagement* (*itil:ServiceOperation*),

- *itil:ServiceAsset_and_ConfigurationManagement* (*itil:ServiceTransition*),

- *itil:ChangeManagement* (*itil:ServiceTransition*),

- *itil:CapacityManagement* (*itil:ServiceDesign*),

- *itil:AvailabilityManagement* (*itil:ServiceDesign*) and

- *itil:ServiceLevelManagement* (*itil:ServiceDesign*).

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Operation*, p. 100-101.

Object Properties: *itil:hasInterfaceRelationType* and *itil:interfaceValue*

Datatype Properties: *itil:interfaceRelationDescription* and *itil:interfaceRelationName*

Class: StrategyProcess

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:StrategyProcess* concept represents the structured set of activities designed to accomplish the Service Strategy phase.

Generalization: itil:Process

Relation to ITIL: We use the *itil:StrategyProcess* class to classify the processes that support the Service Strategy phase (subclasses): *itil:DemandManagement*, *itil:FinancialManagement* and *itil:ServicePortfolioManagement*.

Object Properties: *itil:inStrategyStage* (subproperty of *itil:inServiceStage*) and inherited from *itil:Process*

Datatype Properties: Inherited from *itil:Process*

Class: DemandManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:DemandManagement* process represents the activities that understand and influence customer demand for services and the provision of capacity to meet these demands. At a strategic level, *itil:DemandManagement* can involve analysis of PBAs and UPs. At a tactical level it can involve use of differential charging to encourage customers to use IT services at less busy times.

The *itil:DemandManagement* process is a critical aspect of service management. Poorly managed demand is a source of risk for service providers because of uncertainty in demand. Excess capacity generates cost without creating value that provides a basis for cost recovery. Customers are reluctant to pay for idle capacity unless it has value for them.

Business processes are the primary source of demand for services. PBAs influence the demand patterns seen by the service providers. It is very important to study the customer's business to identify, analyze and codify such patterns to provide sufficient basis for Capacity Management. Visualize the customer's business activity and plans in terms of the demand for supporting services. For example, the fulfillment of a purchase order (business activity) may result in a set of requests (demand) generated by the order-to-cash process (business process of customer). Analyzing and tracking the activity patterns of the business process makes it possible to predict demand for services in the catalogue that support the process. It is also possible to predict demand for underlying service assets that support those services. Every additional unit of demand generated by business activity is allocated to a unit of service capacity.

Generalization: *itil:StrategyProcess*

Relation to ITIL: *ITIL Service Strategy*, p. 201-215 and p. 349 (Demand Management definition).

Object Properties: Inherited from *itil:StrategyProcess*

Datatype Properties: Inherited from *itil:StrategyProcess*

Class: FinancialManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:FinancialManagement* process provides the business and IT with the quantification, in financial terms, of the value of IT Services, the value of the assets underlying the provisioning of those services, and the qualification of operational forecasting. Talking about IT in terms of services is the crux of changing the perception of IT and its value to the business. Therefore, a significant portion of *itil:FinancialManagement* process is working in tandem with IT and the business to help identify, document and agree on the value of the services being received, and the

enablement of service demand modeling and management. The *itil:FinancialMangement* process is responsible for managing an IT service provider's budgeting, accounting and charging requirements.

The *itil:FinancialManagement* process as a strategic tool is equally applicable to all three service provider types. Internal service providers are increasingly asked to operate with the same levels of financial visibility and accountability as their business unit and external counterparts. Moreover, technology and innovation have become the core revenue-generating capabilities of many companies:

- *Type I – Internal service provider:* Type I providers are typically business functions embedded within the business units they serve. The business units themselves may be part of a larger enterprise or parent organization. Business functions such as finance, administration, logistics, human resources, and IT provide services required by various parts of the business. They are funded by overheads and are required to operate strictly within the mandates of the business. Type I providers have the benefit of tight coupling with their owner-customers, avoiding certain costs and risks associated with conducting business with external parties.
- *Type II – Shared Services Unit:* Functions such as finance, IT, human resources, and logistics are not always at the core of an organization's competitive advantage. Hence, they need not be maintained at the corporate level where they demand the attention of the chief executive's team. Instead, the services of such shared functions are consolidated into an autonomous special unit called a Shared Services Unit (SSU). This model allows a more devolved governing structure under which SSU can focus on serving business units as direct customers. SSU can create, grow, and sustain an internal market for their services and model themselves along the lines of service providers in the open market. Like corporate business functions, they can leverage opportunities across the enterprise and spread their costs and risks across a wider base. Unlike corporate business functions, they have fewer protections under the banner of strategic value and core competence. They are subject to comparisons with external service providers whose business practices, operating models and strategies they must emulate and whose performance they should approximate if not exceed. Performance gaps are justified through benefits received through services within their domain of control.
- *Type III – External service provider:* The business strategies of customers sometimes require capabilities readily available from a Type III provider. The additional risks that Type III providers assume over Type I and Type II are justified by increased flexibility and freedom to pursue opportunities. Type III providers can offer competitive prices and drive down unit costs by consolidating demand. Certain business strategies are not adequately served by internal service providers such as Type I and Type II. Customers may pursue sourcing strategies requiring services from external providers. The motivation may be access to knowledge, experience, scale, scope, capabilities, and resources that are either beyond the reach of the organization or outside the scope of a carefully considered investment portfolio. Business strategies often require reductions in the asset base, fixed costs, operational risks, or the

redeployment of financial assets. Competitive business environments often require customers to have flexible and lean structures. In such cases it is better to buy services rather than own and operate the assets necessary to execute certain business functions and processes. For such customers, Type III is the best choice for a given set of services. The experience of such providers is not limited to any one enterprise or market. The breadth and depth of such experience is often the single most distinctive source of value for customers. The breadth comes from serving multiple types of customers or markets. The depth comes from serving multiples of the same type. As a counter-balance, Type III providers mitigate a type of risk inherent to Types I and II: business functions and shared service units are subject to the same system of risks as their business unit or enterprise parent. This sets up a vicious cycle, whereby risks faced by the business units or the enterprise are transferred to the service units and then fed back with amplification through the services utilized. Customers may reduce systemic risks by transferring them to external service providers who spread those risks across a larger value network.

Like its business equivalent, *itil:FinancialManagement* responsibilities and activities do not exist solely within the IT finance and accounting domain. Rather, many parts of the enterprise interact to generate and consume IT financial information, including operations and support units, project management organizations, application development, infrastructure, change management, business units, end users etc. These entities aggregate, share and maintain the financial data they need. The financial management data used by an IT organization may reside in, and be owned by the accounting and finance domain, but responsibility for generating and utilizing it extends to other areas.

The *itil:FinancialManagement* process is a key input to the *itil:ServicePortfolioManagement*. By understanding cost structures applied in the provisioning of an IT service, an organization can benchmark that service cost against other IT service providers. In this way, organizations can use IT financial information, together with service demand and internal capability information to make beneficial decisions regarding whether a certain service should be provisioned internally. For example, if an organization identifies its internal cost of providing *Service A* to be 80€ per month per user, and then finds a provider with the economics of scale and the focused skill set required to offer the identical service for 55€ per month, the organization may decide that it would rather focus its resources on other IT services where it possesses a greater ability to offer lower cost and/or higher quality, and to outsource Service A to the other IT service provider.

The *itil:FinancialManagement* process provides key inputs for Service Provisioning Optimization (SPO). SPO examines the financial inputs and constraints of service components or delivery models to determine if alternatives should be explored relating to how a service can be provisioned differently to make it more competitive in terms of cost or quality.

One goal of *itil:FinancialManagement* is to ensure proper funding for the delivery and consumption of services. Planning provides financial translation and qualification of expected future demand for IT Services. Financial management planning departs from historical IT planning by focusing on demand and supply variances resulting from

business strategy, capacity inputs and forecasting, rather than traditional individual line item expenditures or business cost accounts. As with planning for any other business organization, input should be collected from all areas of the IT organization and the business. Planning can be categorized into three main areas, each representing financial results that are required for continued visibility and service valuation:

- Operating and Capital (general and fixed asset ledgers)
- Demand (need and use of IT services)
- Regulatory and Environmental (compliance).

The *itil:FinancialManagement* provides the shared analytical models and knowledge used throughout an enterprise in order to assess the expected value and/or return of a given initiative, solution, program or project in a standardized fashion. It sets the thresholds that guide the organization in determining what level of analytical sophistication is to be applied to various projects based on size, scope, resources, cost and related parameters.

Accounting within *itil:FinancialManagement* differs from traditional accounting in that additional category and characteristics must be defined that enable the identification and tracking of service-oriented expense or capital items. The *itil:FinancialManagement* process plays a translational role between corporate financial systems and service management. The result of a service-oriented accounting function is that far greater detail and understanding is achieved regarding service provisioning and consumption, and the generation of data that feeds directly into the planning process. The functions and accounting characteristics that come into play are discussed below:

- *Service recording:* the assignment of a cost entry to the appropriate service. Depending on how services are defined, and the granularity of the definitions, there may be additional sub-service components.
- *Cost Types:* these are higher level expenses categories such as hardware, software, labor, administration, etc. These attributes assist with reporting and analyzing demand and usage of services and their components in commonly used financial terms.
- *Cost classifications:* there are also classifications within services that designate the end purpose of the cost. These include classifications such as:
  - Capital/operational: this classification addresses different accounting methodologies that are required by the business and regulatory agencies.
  - Direct/indirect: this designation determines whether a cost will be assigned directly or indirectly to a consumer or service.
    - Direct costs are charged directly to a service since it is the only consumer of the expense.
    - Indirect or ‘shared’ costs are allocated across multiple services since each service may consume a portion of the expense.
  - Fixed/variable: this segregation of costs is based on contractual commitments of time or price. The strategic issue around this classification is that the business should seek to optimize fixed service

costs and minimize the variable in order to maximize predictability and stability.

- Cost Units: a cost unit is the identified unit of consumption that is accounted for a particular service or service asset.

Variable Cost Dynamics (VCD) within *itil:FinancialManagement* focuses on analyzing and understanding the multitude of variables that impact service cost, how sensitive those elements are to variability, and the related incremental value changes that result. Among other benefits, VCD analysis can be used to identify a marginal change in unit cost resulting from adding or subtracting one or more incremental units of a service. Such an analysis is helpful when applied toward the analysis of expected impacts from events such as acquisitions, divestitures, changes to the service portfolio or service provisioning alternatives etc.

On the other hand, funding addresses the financial impacts from changes to current and future demand for IT services and the way in which IT will retain the funds to continue operations. There are various traditional models for the funding of IT services. Since each model assumes a different perspective, yet rests on the same financial data, an increased ability to generate the requisite information translates to increased visibility into service costs and perceived value. The model chosen should always take into account and be appropriate for the current business culture and expectations:

- *Rolling Plan Funding:* In a rolling plan, as one cycle completes another cycle of funding is added. This plan encourages a constant cycle of funding. However, it only addresses timing and does not necessarily increase accuracy. This type of model for funding would work well with an *itil:ServiceLifecycle* treatment where a commitment to fund a service is made at the beginning of the lifecycle and rolls until changes are made or the lifecycle has ended.
- *Trigger-Based Plans:* Trigger-based funding occurs when identified critical triggers occur and set off planning for a particular event. For example, the *itil:ChangeManagement* process would be a trigger to the planning process for all approved changes that have financial impacts. Another trigger might be capacity planning where insight into capacity variances would affect the financial translation of IT services. This type of planning alleviates timing issues with accounting for past events, since the process requires future planning at the time of the change. It would be a good plan to use with portfolio service management since it deals with services on a lifecycle basis.
- *Zero-Based Funding:* This funding refers to how funding of IT occurs. Funding is only enough to bring the balance of the IT financial centre back to zero or to bring the balance of the funding of a service back to zero until another funding cycle. This equates to funding only the actual costs to deliver the IT services.

Finally, a *Business Impact Analysis* (BIA) seeks to identify a company's most critical business services through analysis of outage severity translated into a financial value, coupled with operational risk. This information can help shape and enhance operational performance by enabling better decision making regarding prioritization of incident handling, problem management focus, change and release management operations, project priority, and so on. It is a beneficial tool for identifying the cost of service outage to a company, and the relative worth of a service. These two concepts are not

identical. The cost of service outage is a financial value placed on a specific service, and is meant to reflect the value of lost productivity and revenue over a specific period of time. The worth of a service relative to other services in a portfolio may not result exclusively from financial characteristics. Service Value is derived from characteristics that may go beyond *itil:FinancialManagement*, and represent aspects such as the ability to complete work or communicate with customers that may not be directly related to revenue generation. Both of these elements can be identified to a very adequate degree by the use of a BIA.

Generalization: *itil:StrategyProcess*

Relation to ITIL: *ITIL Service Strategy*, p. 69-76, p. 148-173, p. 343 (Business Impact Analysis definition) and p. 352 (Financial Management definition).

Object Properties: Inherited from *itil:StrategyProcess*

Datatype Properties: Inherited from *itil:StrategyProcess*

Class: ServicePortfolioManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:ServicePortfolioManagement* process responsible for managing the *itil:ServicePortfolio*. An *itil:ServicePortfolio* describes a provider's services in terms of business value. It articulates business needs and the provider's response to those needs. By definition, business value terms correspond to marketing terms, providing a means for comparing service competitiveness across alternative providers. By acting as the basis of a decision framework, a service portfolio either clarifies or helps to clarify the following strategic questions:

- Why should a customer buy these services?
- Why should they buy these services from us?
- What are the pricing or chargeback models?
- What are our strengths and weaknesses, priorities and risk?
- How should our resources and capabilities be allocated?

The *itil:ServicePortfolioManagement* considers services in terms of the business value that they provide. The *itil:ServicePortfolioManagement* is a dynamic method for governing investments in service management across the enterprise and managing them for value.

The operative word is method. Often the term portfolio is marginalized to a list of services, applications, assets or projects. A portfolio is essentially a group of investments that share similar characteristics. They are grouped by size, discipline or strategic value. There are few fundamental differences between IT portfolio management, project portfolio management and service portfolio management (SPM).

All are enabling techniques for governance. The difference is in the implementation details.

As a dynamic and ongoing process set, the *itil:ServicePortfolioManagement* should include the following work methods:

- *Define:* inventory services, ensure business cases and validate portfolio data
- *Analyze:* maximize portfolio value, align and prioritize and balance supply and demand
- *Approve:* finalize proposed portfolio, authorize services and resources
- *Charter:* communicate decisions, allocate resources and charter services.

Generalization: *itil:StrategyProcess*

Relation to ITIL: *ITIL Service Strategy*, p. 186-200 and p. 367 (Service Portfolio Management definition).

Object Properties: Inherited from *itil:StrategyProcess*

Datatype Properties: Inherited from *itil:StrategyProcess*

Class: DesignProcess

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:DesignProcess* concept represents the structured set of activities designed to accomplish the Service Design phase.

Generalization: *itil:Process*

Relation to ITIL: We use the *itil:DesignProcess* class to classify the processes that support the Service Design phase (subclasses): *itil:AvailabilityManagement*, *itil:CapacityManagement*, *itil:InformationSecurityManagement*, *itil:ITServiceContinuityManagement*, *itil:ServiceCatalogManagement*, *itil:ServiceLevelManagement* and *itil:SupplierManagement*.

Object Properties: *itil:inDesignStage* (subproperty of *itil:inServiceStage*) and inherited from *itil:Process*

Datatype Properties: Inherited from *itil:Process*

Class: AvailabilityManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:AvailabilityManagement* is the process responsible for defining, analyzing, planning, measuring and improving all aspects of the availability of IT services. *itil:AvailabilityManagement* is responsible for ensuring that all IT infrastructure, processes, tools, roles etc. are appropriate for the agreed service level

targets for availability. The purpose of *itil:AvailabilityManagement* is to provide a point of focus and management for all availability-related issues, relating to both services and resources, ensuring that availability targets in all areas are measured and achieved.

The *itil:AvailabilityManagement* process does not include *Business Continuity Management* (BCM) and the resumption of business processing after a major disaster.

The support of BCM is included within the *itil:ITServiceContinuityManagement* process. However, *itil:AvailabilityManagement* does provide key inputs to *itil:ITServiceContinuityManagement*, and the two processes have a close relationship, particularly in the assessment and management of risks and in the implementation of risk reduction and resilience measures.

The *itil:AvailabilityManagement* process has two key elements:

(1) *Reactive activities:* the reactive aspect of *itil:AvailabilityManagement* involves the monitoring, measuring, analysis and management of all events, incidents and problems involving unavailability. These activities are principally involved within operational roles.

(2) *Proactive activities:* the proactive activities of *itil:AvailabilityManagement* involve the proactive planning, design and improvement of availability. These activities are principally involved within design and planning roles.

The *itil:AvailabilityManagement* process relies on the monitoring, measurement, analysis and reporting of the following aspects:

- *Availability:* the ability of a service, component or CI to perform its agreed function when required. It is often measured and reported as a percentage:

$$Availability(\%) = \frac{(Agreed\ Service\ Time\ (AST) - downtime)}{Agreed\ Service\ Time\ (AST)} \times 100\%$$

  Downtime should only be included in the above calculation when it occurs within the *Agreed Service Time* (AST). However, total downtime should also be recorded and reported.

- *Reliability:* a measure of how long a service, component or CI can perform its agreed function without interruption. The reliability of the service can be improved by increasing the reliability of individual components or by increasing the resilience of the service to individual component failure (i.e. increasing the component redundancy, for example, by using load-balancing techniques). It is often measured and reported as *Mean Time Between Service Incidents* (MTBSI) or *Mean Time Between Failures* (MTBF):

$$Reilability(MTBSI\ in\ hours) = \frac{Available\ time\ in\ hours}{Number\ of\ breaks}$$

$$Reilability(MTBF\ in\ hours) = \frac{Available\ time\ in\ hours - Total\ downtime\ in\ hours}{Number\ of\ breaks}$$

- *Maintainability:* a measure of how quickly and effectively a service, component or CI can be restored to normal working after a failure. It is measured and reported as *Mean Time to Restore Service* (MTRS) and should be calculated using the following formula:

$$Maintainability(MTRS\ in\ hours) = \frac{Total\ downtime\ in\ hours}{Number\ of\ service\ breaks}$$

  MTRS should be used to avoid the ambiguity of the more common industry term *Mean Time To Repair* (MTTR), which in some definitions includes only repair time, but in others includes recovery time. The downtime in MTRS covers all the contributory factors that make the service, component or CI unavailable:

  - Time to record
  - Time to respond
  - Time to resolve
  - Time to physically repair or replace
  - Time to recover

- *Serviceability:* the ability of a third-party supplier to meet the terms of their contract. Often this contract will include agreed levels of availability, reliability and/or maintainability for a supporting service or component.

A key output from the *itil:AvailabilityManagement* process is the measurement and reporting of IT availability. Availability measures should be incorporated into SLAs, *Operational Level Agreements* (OLAs) and *Underpinning Contracts* (UCs). These should be reviewed regularly at *service level review meetings*. Measurement and reporting provide the basis for:

- Monitoring the actual availability delivered versus agreed targets
- Establishing measures of availability and agreeing availability targets with the business
- Identifying unacceptable levels of availability that impact the business and users
- Reviewing availability with the IT support organization
- Continual improvement activities to optimize availability

Component Failure Impact Analysis (CFIA) can be used to predict and evaluate the impact on IT service arising from component failures within the technology. The output from a CFIA can be used to identify where additional resilience should be considered to prevent or minimize the impact of component failure to the business operation and users. This is particularly important during the Service Design stage, where it is necessary to predict and evaluate the impact on IT service availability arising from component failures within the proposed IT Service Design. However, the technique can also be applied to existing services and infrastructure.

The *itil:AvailabilityManagement* process should also maintain an *Availability Management Information System* (AMIS) that contains all of the measurements and information required to complete the *itil:AvailabilityManagement* process and provide the appropriate information to the business on the level of IT service provided. This information, covering services, components and supporting services, provides the basis for regular, ad hoc and exception availability reporting and the identification of trends

within the data for the instigation of improvement activities. These activities and the information contained within the AMIS provide the basis for developing the content of the availability plan.

Generalization: *itil:DesignProcess*

Relation to ITIL: *ITIL Service Design*, p. 167-215 and p. 417 (Availability Management definition).

Object Properties: Inherited from *itil:DesignProcess*

Datatype Properties: Inherited from *itil:DesignProcess*

Class: CapacityManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:CapacityManagement* is the process responsible for ensuring that the capacity of IT services and the IT infrastructure is able to deliver agreed service level targets in a cost effective and timely manner. The *itil:CapacityManagement* process considers all resources required to deliver the IT service, and plans for short-, medium- and long-term business requirements. The purpose of *itil:CapacityManagement* is to provide a point of focus and management for all capacity- and performance-related issues, relating to both services and resources.

The *itil:CapacityManagement* process provides the necessary information on current and planned resource utilization of individual components to enable IT service providers to decide, with confidence:

- *Which components to upgrade:* i.e. more memory, faster storage devices, faster processors, greater bandwidth.
- *When to upgrade:* ideally this is not too early, resulting in expensive overcapacity, nor too late, failing to take advantage of advances in new technology, resulting in bottle-necks, inconsistent performance and, ultimately, customer dissatisfaction and lost business opportunities.
- *How much the upgrade will cost:* the forecasting and planning elements of the *itil:CapacityManagement* process feed into budgetary lifecycles, ensuring planned investment.

The *Capacity Management Information System* (CMIS) is the cornerstone of a successful *itil:CapacityManagement* process. Information contained within the CMIS is stored and analyzed by all the subprocesses of *itil:CapacityManagement* because it is a repository that holds a number of different types of data, including business, service, resource or utilization and financial data, from all areas of technology.

However, the CMIS is unlikely to be a single database, and probably exists in several physical locations. Data from all areas of technology, and all components that make up the IT services, can then be combined for analysis and provision of technical and management reporting. Only when all of the information is integrated can ‘end-to-end’ service reports be produced. The integrity and accuracy of the data within the CMIS

needs to be carefully managed. If the CMIS is not part of an overall *Configuration Management System* (CMS) or *Service Knowledge Management System* (SKMS), then links between these systems need to be implemented to ensure consistency and accuracy of the information recorded within them.

The information in the CMIS is used to form the basis of performance and capacity management reports and views that are to be delivered to customers, IT management and technical personnel. Also, the data is used to generate future capacity forecasts and allow *itil:CapacityManagement* to plan for future capacity requirements. Often a Web interface is provided to the CMIS to provide the different access and views required outside of the *itil:CapacityManagement* process itself.

The full range of data types stored within the CMIS is as follows:

- *Business data:* The business data is used to forecast and validate how changes in business drivers affect the capacity and performance of the IT infrastructure. Business data should include business transactions or measurements such as the number of accounts, the number of invoices generated, the number of product lines.
- *Service data:* To achieve a service-orientated approach to the *itil:CapacityManagement* process, service data should be stored within the CMIS. Typical service data are transaction response times, transaction rates, workload volumes, etc. In general, the *itil:SLA*(s) and *Service Level Requirements* (*itil:SLR*(s)) provide the service targets for which the *itil:CapacityManagement* process needs to record and monitor data. To ensure that the targets in the *itil:SLA*(s) are achieved, *Service Level Management* (SLM) thresholds should be included, so that the monitoring activity can measure against these service thresholds and raise exception warnings and reports before service targets are breached.
- *Component utilization data:* The CMIS also needs to record resource data consisting of utilization, threshold and limit information on all of the technological components supporting the services. Most of the IT components have limitations on the level to which they should be used. Beyond this level of utilization, the resource will be over-utilized and the performance of the services using the resource will be impaired. For example, the maximum recommended level of utilization on a processor could be 80%, or the utilization of a shared Ethernet LAN segment should not exceed 40%.

  Also, components have various physical limitations beyond which greater connectivity or use is impossible. For example, the maximum number of connections through an application or a network gateway is 100, or a particular type of disk has a physical capacity of 15 Gb. The CMIS should therefore contain, for each component and the maximum performance and capacity limits, current and past utilization rates and the associated component thresholds. Over time this can require vast amounts of data to be accumulated, so there need to be good techniques for analyzing, aggregating and archiving this data.
- *Financial data:* The *itil:CapacityManagement* process requires financial data. For evaluating alternative upgrade options, when proposing various scenarios in the capacity plan, the financial cost of the upgrades to the components of the IT

infrastructure, together with information about the current IT hardware budget, must be known and included in the considerations. Most of this data may be available from the Financial Management for IT services process (*itil:FinancialManagement*), but the *itil:CapacityManagement* process needs to consider this information when managing the future business requirements.

Generalization: *itil:DesignProcess*

Relation to ITIL: *ITIL Service Design*, p. 134-166 and p. 420 (Capacity Management definition).

Object Properties: Inherited from *itil:DesignProcess*

Datatype Properties: Inherited from *itil:DesignProcess*

Class: InformationSecurityManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:InformationSecurityManagement* is the process that ensures the confidentiality, integrity and availability of an organization's assets, information, data and IT services. The *itil:InformationSecurityManagement* process usually forms part of an organizational approach to security management that has a wider scope than the IT service provider, and includes handling of paper, building access, phone calls, etc., for the entire organization.

The term 'information' is used as a general term and includes data stores, databases and metadata. The objective of information security is to protect the interests of those relying on information, and the systems and communications that deliver the information, from harm resulting from failures of availability, confidentiality and integrity.

The framework or the *Information Security Management System* (ISMS) provides a basis for the development of a cost-effective information security program that supports the business objectives. It will involve the four Ps of People, Process, Products and Partners as well as technology and suppliers to ensure high levels of security are in place. ISO 27001 is the formal standard against which organizations may seek independent certification of their ISMS (meaning their frameworks to design, implement, manage, maintain and enforce information security processes and controls systematically and consistently throughout the organizations).

Generalization: *itil:DesignProcess*

Relation to ITIL: *ITIL Service Design*, p. 244-259 and p. 429 (Information Security Management definition).

Object Properties: Inherited from *itil:DesignProcess*

Datatype Properties: Inherited from *itil:DesignProcess*

Class: ITServiceContinuityManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:ITServiceContinuityManagement* is the process responsible for managing risks that could seriously affect IT services. The *itil:ITServiceContinuityManagement* process ensures that the IT service provider can always provide minimum agreed service levels, by reducing the risk to an acceptable level and planning for the recovery of IT services. The *itil:ITServiceContinuityManagement* process should be designed to support business continuity management. Therefore, ITSM should maintain a set of IT service continuity plans and IT recovery plans that support the overall *Business Continuity Plans* (BCPs) of the organization.

The *itil:ITServiceContinuityManagement* process primarily considers the IT assets and configurations that support the business processes. If (following a disaster) it is necessary to relocate to an alternative working location, provision will also be required for items such as office and personnel accommodation, copies of critical paper records, courier services and telephone facilities to communicate with customers and third parties.

Like all elements of ITSM, successful implementation of the *itil:ITServiceContinuityManagement* process can only be achieved with senior management commitment and the support of all members of the organization. Ongoing maintenance of the recovery capability is essential if it is to remain effective. The purpose of the *itil:ITServiceContinuityManagement* process is to maintain the necessary ongoing recovery capability within the IT services and their supporting components.

The *itil:ITServiceContinuityManagement* process includes:

- The agreement of the scope of the *itil:ITServiceContinuityManagement* process and the policies adopted.
- *Business Impact Analysis* (BIA) to quantify the impact loss of IT service would have on the business.
- *Risk Analysis* (RA): the risk identification and risk assessment to identify potential threats to continuity and the likelihood of the threats becoming reality. This also includes taking measures to manage the identified threats where this can be cost-justified.
- Production of an overall *IT service continuity management* (ITSCM) strategy that must be integrated into the BCM strategy. This can be produced following the two steps identified above, and is likely to include elements of risk reduction as well as selection of appropriate and comprehensive recovery options.
- Production of an ITSCM plan, which again must be integrated with the overall BCM plans.
- Testing of the plans.
- Ongoing operation and maintenance of the plans.

Generalization: *itil:DesignProcess*

Relation to ITIL: *ITIL Service Design*, p. 216-243 and p. 430 (IT Service Continuity Management definition).

Object Properties: Inherited from *itil:DesignProcess*

Datatype Properties: Inherited from *itil:DesignProcess*

Class: ServiceCatalogManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:ServiceCatalogManagement* is the process that provides a single source of consistent information on all of the agreed services, and ensures that it is widely available to those who are approved to access it.

The objective of the *itil:ServiceCatalogManagement* process is to manage the information contained within the service catalog, and to ensure that it is accurate and reflects the current details, status, interfaces and dependencies of all services that are being run, or being prepared to run, in the live environment.

The service catalog has two aspects:

- *The Business Service Catalog:* containing details of all the IT services delivered to the customer, together with relationships to the business units and the business process that rely on the IT services. This is the customer view of the service catalog.
- *The Technical Service Catalog:* containing details of all the IT services delivered to the customer, together with relationships to the supporting services, shared services, components and CIs necessary to support the provision of the service to the business. This should underpin the *Business Service Catalog* and not form part of the customer view.

Some organizations only maintain either a *Business Service Catalog* or a *Technical Service Catalog*. The preferred situation adopted by the more mature organizations maintains both aspects within a single service catalog, which is part of a totally integrated ITSM activity and service portfolio. The *Business Service Catalog* facilitates the development of a much more proactive or even pre-emptive *itil:ServiceLevelManagement* process, allowing it to develop more into the field of *Business Service Management* (BSM). The *Technical Service Catalog* is extremely beneficial when constructing the relationship between services, SLAs, OLAs and other underpinning agreements and components, as it will identify the technology required to support a service and the support group(s) that support the components. The combination of a *Business Service Catalog* and a *Technical Service Catalog* is invaluable for quickly assessing the impact of incidents and changes on the business.

Generalization: *itil:DesignProcess*

Relation to ITIL: *ITIL Service Design*, p. 101-108.

Object Properties: Inherited from *itil:DesignProcess*

Datatype Properties: Inherited from *itil:DesignProcess*

Class: ServiceLevelManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:ServiceLevelManagement* is the process responsible for negotiating the *itil:SLA*(s), and ensuring that these are met. The *itil:ServiceLevelManagement* process is responsible for ensuring that all *itil:Process*(s), *itil:OLA*(s), and *itil:UC*(s), are appropriate for the agreed service level targets. The *itil:SLA(s)* provide the basis for managing the relationship between the service provider and the customer, and the *itil:ServiceLevelManagement* process provides that central point of focus for a group of customers, business units or lines of business. Using the service catalog as an aid, the *itil:ServiceLevelManagement* process must design the most appropriate *itil:SLA* structure to ensure that all services and all customers are covered in a manner best suited to the organization's needs. Also, the *itil:ServiceLevelManagement* process monitors and reports on service levels, and holds regular customer reviews.

The *itil:ServiceLevelManagement* process needs to manage the expectation and perception of the business, customers and users and ensure that the quality of service delivered by the service provider is matched to those expectations and needs. In order to do this effectively, the *itil:ServiceLevelManagement* process should establish and maintain *itil:SLA*(s) for all current live services and manage the level of service provided to meet the targets and quality measurements contained within the *itil:SLA*(s). The *itil:ServiceLevelManagement* process should also produce and agree *itil:SLR*(s) for all planned new or changed services.

The goal of the *itil:ServiceLevelManagement* process is to ensure that an agreed level of IT service is provided for all current IT services, and that future services are delivered to agreed achievable targets. If the targets are not aligned with business needs, then service provider activities and service levels will not be aligned with business expectations and problems will develop. Proactive measures are also taken to seek and implement improvements to the level of service delivered.

The *itil:ServiceLevelManagement* process should include instigation and coordination of a *Service Improvement Plan* (SIP) for the management, planning and implementation of all service and process improvements. A SIP is an overall program or plan of prioritized improvement actions, encompassing all services and all processes, together with associated impacts and risks. In other words, a SIP is a formal plan to implement improvements to a process or IT service.

The *itil:ServiceLevelManagement* process should also include activities and procedures for the logging and management of all complaints and compliments. The logging procedures are often performed by the itil:SERVICE_DESK (*itil:RoleType* instance) as they are similar to those of Incident Management and Request Fulfillment processes. The definition of a complaint and compliment should be agreed with the customers, together with agreed contact points and procedures for their management and analysis. All complaints and compliments should be recorded and communicated to the relevant parties. All complaints should also be actioned and resolved to the satisfaction of the originator. If not, there should be an escalation contact and procedure for all complaints

that are not actioned and resolved within an appropriate timescale. All outstanding complaints should be reviewed and escalated to senior management where appropriate. Reports should also be produced on the numbers and types of complaints, the trends identified and actions taken to reduce the numbers received. Similar reports should also be produced for compliments.

Generalization: *itil:DesignProcess*

Relation to ITIL: *ITIL Service Design*, p. 109-133, p. 441 (Service Improvement Plan definition) and p. 442 (Service Level Management definition).

Object Properties: Inherited from *itil:DesignProcess*

Datatype Properties: Inherited from *itil:DesignProcess*

Class: SupplierManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:SupplierManagement* is the process responsible for ensuring that all contracts with suppliers support the needs of the business, and that all suppliers meet their contractual commitments. The purpose of the *itil:SupplierManagement* process is to obtain value for money from suppliers and to ensure that suppliers perform to the targets contained within their contracts and agreements, while conforming to all of the terms and conditions. All *itil:SupplierManagement* process activity should be driven by a supplier strategy and policy from *itil:ServiceStrategy*. In order to achieve consistency and effectiveness in the implementation of the policy, a *Supplier and Contracts Database* (SCD) should be established, together with clearly defined roles and responsibilities.

SCDs are beneficial because they can be used to promote preferred suppliers and to prevent purchasing of unapproved or incompatible items. By coordinating and controlling the buying activity, the organization is more likely to be able to negotiate preferential rates.

It is essential that *itil:SupplierManagement* processes and planning are involved in all stages of the service lifecycle, from strategy and design, through transition and operation, to improvement. The complex business demands require the complete breadth of skills and capability to support provision of a comprehensive set of IT services to a business, therefore the use of value networks and the suppliers and the services they provide are an integral part of any end-to-end solution. Suppliers and the management of suppliers and partners are essential to the provision of quality IT services.

Ideally the SCD should form an integrated element of a comprehensive CMS or SKMS, recording all supplier and contract details, together with details of the type of *itil:ITService*(s) provided by each supplier (*itil:ITServiceProvider*), and all other information and relationships with other associated *itil:CI*s. The services provided by suppliers will also form a key part of the *itil:ServicePortfolio*. The relationship between the *itil:SupportingService*(s) and the IT and *itil:CoreService*(s) they support are key to providing quality IT services.

Adding new suppliers or contracts to the SCD needs to be handled via the *itil:ChangeManagement* process, to ensure that any impact is assessed and understood. In most *itil:ITServiceProvider*(s), the SCD is owned by the *itil:SupplierManagement* process or the procurement or purchasing department. The SCD provides a single, central focal set of information for the management of all suppliers and contracts.

Generalization: *itil:DesignProcess*

Relation to ITIL: *ITIL Service Design*, p. 260-286 and p. 445 (Supplier Management definition).

Object Properties: Inherited from *itil:DesignProcess*

Datatype Properties: Inherited from *itil:DesignProcess*

Class: TransitionProcess

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO).

Description: The *itil:TransitionProcess* concept represents the structured set of activities designed to accomplish the Service Transition phase.

Generalization: *itil:Process*

Relation to ITIL: We use the *itil:TransitionProcess* class to classify the processes that support the Service Transition phase (subclasses): *itil:ChangeManagement*, *itil:Evaluation*, *itil:KnowledgeManagement*, *itil:Release_and_DeploymentManagement*, *itil:ServiceAsset_and_ConfigurationManagement*, *itil:ServiceValidation_and_Testing* and *itil:TransitionPlanning_and_Support*.

Object Properties: *itil:inTransitionStage* (subproperty of *itil:inServiceStage*) and inherited from *itil:Process*

Datatype Properties: Inherited from *itil:Process*

Class: ChangeManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO).

Description: The *itil:ChangeManagement* is the process responsible for controlling the lifecycle of all changes. The primary objective of *itil:ChangeManagement* is to enable beneficial changes to be made, with minimum disruption to IT services.

Generalization: *oc:TransitionProcess*

Relation to ITIL: *ITIL Service Transition*, p. 77-117 and p. 371 (Change Management definition).

Object Properties: Inherited from *oc:TransitionProcess*

Datatype Properties: Inherited from *oc:TransitionProcess*

Class: Evaluation

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO).

Description: The *itil:Evaluation* is the process responsible for assessing a new or changed IT service to ensure that risks have been managed and to help determine whether to proceed with the change. The *itil:Evaluation* process is also used to mean comparing an actual outcome with the intended outcome, or comparing one alternative with another.

Generalization: *oc:TransitionProcess*

Relation to ITIL: *ITIL Service Transition*, p. 245-255 and p. 376 (Evaluation definition).

Object Properties: Inherited from *oc:TransitionProcess*

Datatype Properties: Inherited from *oc:TransitionProcess*

Class: KnowledgeManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO).

Description: The *itil:KnowledgeManagement* is the process responsible for gathering, analyzing, storing and sharing knowledge and information within an organization. The primary purpose of the *itil:KnowledgeManagement* process is to improve efficiency by reducing the need to rediscover knowledge.

Generalization: *oc:TransitionProcess*

Relation to ITIL: *ITIL Service Transition*, p. 256-273 and p. 381 (Knowledge Management definition).

Object Properties: Inherited from *oc:TransitionProcess*

Datatype Properties: Inherited from *oc:TransitionProcess*

Class: Release_and_DeploymentManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO).

Description: The *itil:Release_and_DeploymentManagement* is the process responsible for both release management and deployment. The release management process is responsible for planning, scheduling and controlling the movement of releases to test and live environments. The primary objective of release management is to ensure that the integrity of the live environment is protected and that the correct components are

released. Deployment is the activity responsible for movement of new or changed hardware, software, documentation, process, etc. to the live environment.

Generalization: *oc:TransitionProcess*

Relation to ITIL: *ITIL Service Transition*, p. 152-206, p. 375 (Deployment definition) and p. 388 (Release and Deployment Management definition) (Release Management definition).

Object Properties: Inherited from *oc:TransitionProcess*

Datatype Properties: Inherited from *oc:TransitionProcess*

Class: ServiceAsset_and_ConfigurationManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The *itil:ServiceAsset_and_ConfigurationManagement* is the process responsible for both configuration management and asset management. The asset management process is responsible for tracking and reporting the value and ownership of financial assets throughout their lifecycle. The configuration management process is the responsible for maintaining information about CIs required to deliver an IT Service, including their Relationships. This information is managed throughout the Lifecycle of the CI.

Generalization: *oc:TransitionProcess*

Relation to ITIL: *ITIL Service Transition*, p. 118-151, p. 366 (Asset Management definition) and p. 373 (Configuration Management definition). *ITIL V3: Glossary of Terms and Definitions* (Service Asset and Configuration Management (SACM) definition).

Object Properties: Inherited from *oc:TransitionProcess*

Datatype Properties: Inherited from *oc:TransitionProcess*

Class: ServiceValidation_and_Testing

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The *itil:ServiceValidation_and_Testing* is the process responsible for validation and testing of a new or changed IT service. The *itil:ServiceValidation_and_Testing* process ensures that the IT service matches its design specification and will meet the needs of the business. Validation is an activity that ensures a new or changed IT service, process, plan, or other deliverable meets the

needs of the business. Validation ensures that business requirements are met even though these may have changed since the original design (do not be confused by the term verification: an activity that ensures a new or changed IT service, process, plan, or other deliverable is complete, accurate, reliable and matches its design specification). Test is an activity that verifies that a CI, IT service, process, etc. meets its specification or agreed requirements. Acceptance is a formal agreement that an IT service, process, plan, or other deliverable is complete, accurate, reliable and meets its specified requirements. Acceptance is usually preceded by evaluation or testing and is often required before proceeding to the next stage of a project or process.

Generalization: *oc:TransitionProcess*

Relation to ITIL: *ITIL Service Transition*, p. 207-244, p. 365 (Acceptance definition), p. 396 (Test definition), p. 397 (Validation definition) and p. 398 (Verification definition). *ITIL V3: Glossary of Terms and Definitions* (Service Validation and Testing definition).

Object Properties: Inherited from *oc:TransitionProcess*

Datatype Properties: Inherited from *oc:TransitionProcess*

Class: TransitionPlanning_and_Support

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The *itil:TransitionPlanning_and_Support* is the process responsible for planning all service transition processes and coordinating the resources that they require. These service transition processes are: *itil:ChangeManagement*, *itil:Evaluation*, *itil:KnowledgeManagement*, *itil:Release_and_DeploymentManagement*, *itil:ServiceAsset_and_ConfigurationManagement* and *itil:ServiceValidation_and_Testing*. Planning is an activity responsible for creating one or more plans. For example, capacity planning.

Generalization: *oc:TransitionProcess*

Relation to ITIL: *ITIL Service Transition*, p. 63-76 and p. 385-386 (Planning definition). *ITIL V3: Glossary of Terms and Definitions* (Transition Planning and Support definition).

Object Properties: Inherited from *oc:TransitionProcess*

Datatype Properties: Inherited from *oc:TransitionProcess*

Class: OperationProcess

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The *itil:OperationProcess* concept represents the structured set of activities designed to accomplish the Service Operation phase.

Generalization: *itil:Process*

Relation to ITIL: We use the *itil:OperationProcess* class to classify the processes that support the Service Operation phase (subclasses): *itil:AccessManagement*, *itil:EventManagement*, *itil:IncidentManagement*, *itil:ProblemManagement*, and *itil:RequestFulfillment*.

Object Properties: *itil:inOperationStage* (subproperty of *itil:inServiceStage*) and inherited from *itil:Process*

Datatype Properties: Inherited from *itil:Process*

Class: AccessManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The *itil:AccessManagement* is the process responsible for allowing users to make use of IT services, data, or other assets. The *itil:AccessManagement* process helps to protect the confidentiality, integrity and availability of assets by ensuring that only authorized users are able to access or modify the assets. The *itil:AccessManagement* process is sometimes referred to as rights management or identity management. The *itil:AccessManagement* process does not decide who has access to which IT services. Rather, The *itil:AccessManagement* process executes the policies and regulations defined during *itil:ServiceStrategy* and *itil:ServiceDesign*. The *itil:AccessManagement* process enforces decisions to restrict or provide access, rather than making the decision.

Generalization: *oc:OperationProcess*

Relation to ITIL: *ITIL Service Operation*, p. 126-135. *ITIL V3: Glossary of Terms and Definitions* (Access Management definition).

Object Properties: Inherited from *oc:OperationProcess*

Datatype Properties: Inherited from *oc:OperationProcess*

Class: EventManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The *itil:EventManagement* is the process responsible for managing *itil:Event*(s) throughout their lifecycle. The *itil:EventManagement* process is one of the main activities of IT operations. The *itil:EventManagement* process monitors all *itil:Event*(s) that occur throughout the IT infrastructure, to monitor normal operation

and to detect and escalate exception conditions. The *itil:EventManagement* process is the basis for operational monitoring and control. In addition, if the *itil:Event*(s) are programmed to communicate operational information as well as warnings and exceptions, they can be used as a basis for automating many routine operations management activities, for example executing scripts on remote devices, or submitting jobs for processing, or even dynamically balancing the demand for a service across multiple devices to enhance performance.

The *itil:EventManagement* therefore provides the entry point for the execution of many *itil:ServiceOperation* processes and activities. In addition, it provides a way of comparing actual performance and behavior against design standards and *itil:SLA*(s). As such, the *itil:EventManagement* process also provides a basis for service assurance and reporting, and service improvement.

Generalization: *oc:OperationProcess*

Relation to ITIL: *ITIL Service Operation*, p. 35, p. 52, p. 67 and p. 374 (Event Management definition). Note that although the *itil:EventMangement* process monitors all the *itil:Event*(s), other *itil:Process*(s) can managed specific *itil:Event*(s). For example, an *itil:Incident* is an *itil:Event* managed by the *itil:IncidentManagement* process.

Object Properties: Inherited from *oc:OperationProcess*

Datatype Properties: Inherited from *oc:OperationProcess*

Class: IncidentManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The *itil:IncidentManagement* is the process for dealing with all incidents; this can include failures, questions or queries reported by the users (usually via a telephone call to the itil:SERVICE_DESK, the *itil:RoleType* instance, of the *itil:ITServiceProvider*), by technical staff, or automatically detected and reported by event monitoring tools. The primary goal of the *itil:IncidentManagement* process is to restore normal service operation as quickly as possible and minimize the adverse impact on business operations, thus ensuring that the best possible levels of service quality and availability are maintained. *Normal service operation* is defined here as service operation within *itil:SLA* limits.

Note that, although both *itil:Incident*(s) and *itil:ServiceRequest*(s) are reported to the itil:SERVICE_DESK, this does not mean that they are the same. The *itil:ServiceRequest*(s) do not represent a disruption to agreed *itil:ITservice*, but are a way of meeting the *itil:Customer*'s needs and may be addressing an agreed *itil:ServiceLevelTarget* in an *itil:SLA*. The *itil:ServiceRequest*(s) are dealt with by the *itil:RequestFulfillment* process and not by the *itil:IncidentManagement* process.

Generalization: *itil:OperationProcess*

Relation to ITIL: *ITIL Service Operation*, p. 86-104 and p. 376 (Incident Management definition).

Object Properties: Inherited from *itil:OperationProcess*

Datatype Properties: Inherited from *itil:OperationProcess*

Class: ProblemManagement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The *itil:ProblemManagement* is the process responsible for managing the lifecycle of all problems. The primary objectives of *itil:ProblemManagement* are to prevent incidents from happening, and to minimize the impact of incidents that cannot be prevented.

Generalization: *oc:OperationProcess*

Relation to ITIL: *ITIL Service Operation*, p. 111-125. *ITIL V3: Glossary of Terms and Definitions* (Problem Management definition).

Object Properties: Inherited from *oc:OperationProcess*

Datatype Properties: Inherited from *oc:OperationProcess*

Class: RequestFulfillment

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The *itil:RequestFulfillment* is the process responsible for managing the lifecycle of all service requests. The term 'service request' is used as a generic description for many varying types of demands that are placed upon the IT department by the users. Many of these are actually small changes, low risk, frequently occurring, low cost, etc. (e.g., a request to change a password, a request to install an additional software application onto a particular workstation, a request to relocate some items of desktop equipment) or maybe just a question requesting information, but their scale and frequent, low-risk nature means that they are better handled by a separate process, rather than being allowed to congest and obstruct the normal incident and change management processes. The value of *itil:RequestFulfillment* is to provide quick and effective access to standard services which business staff can use to improve their productivity or the quality of business services and products. The *itil:RequestFulfillment* process effectively reduces the bureaucracy involved in requesting and receiving access to existing or new services, thus also reducing the cost of providing these services. Centralizing fulfilment also increases the level of control over these services. This in turn can help reduce costs through centralized negotiation with suppliers, and can also help to reduce the cost of support.

Generalization: *oc:OperationProcess*

Relation to ITIL: *ITIL Service Operation*, p. 105-110 and p. 386 (Request Fulfillment definition).

Object Properties: Inherited from *oc:OperationProcess*

Datatype Properties: Inherited from *oc:OperationProcess*

Class: CSIProcess

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO).

Description: The *itil:CSIProcess* concept represents the structured set of activities designed to accomplish the Continual Service Improvement phase.

Generalization: *itil:Process*

Relation to ITIL: We use the *itil:CSIProcess* class to classify the processes that support the Continual Service Improvement phase (subclasses): itil:The7StepImprovement.

Object Properties: *itil:inCSIStage* (subproperty of *itil:inServiceStage*) and inherited from *itil:Process*

Datatype Properties: Inherited from *itil:Process*

Class: The7StepImprovement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO).

Description: The *itil:The7StepImprovement* is a process that spans not only the management organization but the entire service lifecycle. This is a cornerstone of CSI.

Steps:

(1) *Define what you should measure:* At the onset of the service lifecycle, Service Strategy and Service Design should have identified this information. CSI can then start its cycle all over again at *Where are we now?* This identifies the ideal situation for both the Business and IT.

(2) *Define what you can measure:* This activity related to the CSI activities of *Where do we want to be?* By identifying the new service level requirements of the business, the IT capabilities (identified through Service Design and implemented via Service Transition) and the available budgets, CSI can conduct a gap analysis to identify the opportunities for improvement as well as answering the question *How do we get there?*.

(3) *Gathering the data:* In order to properly answer the *Did we get there?* question, data must first be gathered (usually through Service Operations). Data is

gathered based on goals and objectives identified. At this point the data is raw and no conclusions are drawn.

(4) *Processing the data:* Here the data is processed in alignment with the CSFs and KPIs specified. This means that timeframes are coordinated, unaligned data is rationalized and made consistent, and gaps in the data are identified. The simple goal of this step is to process data from multiple disparate sources into an 'apples to apples' comparison. Once we have rationalized the data we can then begin analysis.

(5) *Analyzing the data:* Here the data becomes information as it is analyzed to identify service gaps, trends and the impact on business. It is the analyzing step that is most often overlooked or forgotten in the rush to present data to management.

(6) *Presenting and using the information:* Here the answer to *Did we get there?* is formatted and communicated in whatever way necessary to present to the various stakeholders an accurate picture of the results of the improvement efforts. Knowledge is presented to the business in a form and manner that reflects their needs and assists them in determining the next steps.

(7) *Implementing corrective action:* The knowledge gained is used to optimize, improve and correct services. Managers identify issues and present solutions. The corrective actions that need to be taken to improve the service are communicated and explained to the organization. Following this step the organization establishes a new baseline and the cycle begins anew.

While these seven steps of measurement appear to form a circular set of activities, in fact, they constitute a knowledge spiral. In actual practice, knowledge gathered and wisdom derived from that knowledge at one level of the organization becomes a data input to the next.

Generalization: *itil:CSIProcess*

Relation to ITIL: *ITIL Continual Service Improvement*, p. 54-55 and p. 68-90.

Object Properties: Inherited from *itil:CSIProcess*

Datatype Properties: Inherited from *itil:CSIProcess*

Class: ComputerProgram-CW

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: The OpenCyc concept *oc:ComputerProgram-CW* is a specialization of *oc:PropositionalConceptualWork*, *oc:ComputerFile-CW* and *oc:SoftwareObject-Individual*. Each instance of *oc:ComputerProgram-CW* is a deliberately created abstract object composed of propositions (described by specifications using the *oc:programSpecifications* property) that together constitute a list of instructions for computers to execute. Example instances include *oc:Emacs-TheProgram* and *oc:LinuxKernel-TheProgram*. Instances of this collection are distinct from computer code and from both running and installed programs. The instructions that comprise an

instance of *oc:ComputerProgram-CW* can be expressed as abstract computer code (see *oc:ComputerCode*), but no list of instructions expressed in code constitutes an instance of *oc:ComputerProgram-CW*. Rather, the code in which an instance of *oc:ComputerProgram-CW* is expressed constitutes an instance of *oc:AbstractInformationStructure* that can be related to the program it expresses using the predicate *oc:programCode*.

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use we use the OpenCyc concept *oc:ComputerProgram-CW* to classify the applications that will be implemented in an IT service provider for an ITSMS.

Object Properties: *oc:programCode* and *oc:programSpecifications*

Datatype Properties: none

Class: AbstractInformationStructure

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:AbstractInformationStructure* is a specialization of *oc:AbstractStructure*. Each instance of *oc:AbstractInformationStructure* is an abstract individual comprising abstract symbols and relations between them. Important specializations of this collection include *oc:CharacterString* and *oc:Sentence*. The OpenCyc concept *oc:AbstractInformationStructure* also includes abstract diagrams, graphs, and bit strings. The collection can be more precisely defined as follows: Each *oc:AbstractInformationStructure* is such that each of its physical instantiations consists of instantiations of instances of *oc:AtomicSymbol-Abstract*, arranged in a certain way. For example, the abstract sentence 'The pig flies' is an *oc:AbstractInformationStructure*. Each written instantiation of it consists of an instantiation of the words (symbols) 'The', 'pig' and 'flies', written in that order. (If the *oc:AbstractInformationStructure* 'The pig flies' were spoken, the same words would appear in the same order, i.e. 'The' first, etc., but the sequence would be determined by the arrangement of the spoken words in time, rather than space.) Likewise with abstract diagrams, graphs, etc. Each of these is such that its physical instantiations consist of arrangements of instantiations of instances of *oc:AtomicSymbol-Abstract*. A hard copy of a wiring diagram consists of a group of concrete symbols representing various circuit components, in which these symbols are spatially arranged in a certain in way. The arrangement of the concrete symbols in an instantiation of an *oc:AbstractInformationStructure* is not always a simple matter of arrangement in space or time. The sequence of symbols '0010010111011001' can be instantiated in written, spoken, or electronic forms. In the last case, the order of the symbols is determined by conventions concerning the electronic medium in which it is stored, rather than by any common criterion for precedence or subsequence in space or time.

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use we use the OpenCyc concept *oc:AbstractInformationStructure* to classify the applications that will be implemented in an IT service provider for an ITSMS.

Object Properties: none

Datatype Properties: none

Class: ComputerAIS

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:ComputerAIS* is a specialization of *oc:AbstractInformationStructure*. Each instance of *oc:ComputerAIS* is the abstract information structure of an abstract work whose instantiation in computer memory is intended to have meaning.

Generalization: *oc:AbstractInformationStructure*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use we use the OpenCyc concept *oc:ComputerAIS* to classify the applications that will be implemented in an IT service provider for an ITSMS.

Object Properties: Inherited from *oc:AbstractInformationStructure*

Datatype Properties: Inherited from *oc:AbstractInformationStructure*

Class: ComputerCode

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:ComputerCode* is a specialization of *oc:ComputerAIS*. Each instance of *oc:ComputerCode* is an abstract list of instructions expressed in some computer language including executable binary code.

Generalization: *oc:ComputerAIS*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use we use the OpenCyc concept *oc:ComputerCode* to classify the applications that will be implemented in an IT service provider for an ITSMS.

Object Properties: Inherited from *oc:ComputerAIS*

Datatype Properties: *itil:computerLanguage* and inherited from *oc:ComputerAIS*

Class: Application

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:Application* is software that provides functions that are required by an IT service. Each *itil:Application* implements an *itil:Activity* and it may be part of more than one IT service. An *itil:Application* runs on one or more servers or customers.

Generalization: *oc:ComputerCode*

Relation to ITIL: *ITIL Service Strategy*, p. 340 (Application definition). In our modeling approach for ITSMSs, an *itil:Application* is the code (computer tool) that implements an *itil:Activity*.

Object Properties: *itil:implementsActivity*, *itil:supportsITService* and inherited from *oc:ApplicationProgram*

Datatype Properties: *itil:appDescription*, *itil:appName* and inherited from *oc:ComputerCode*

Class: Situation

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:Situation* is a state or event consisting of one or more objects having certain properties or bearing certain relations to each other. Notable specializations of *oc:Situation* are *oc:Event* and *oc:StaticSituation*.

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:Situation* for the classification of some ITIL concepts such as incident or IT service.

Object Properties: none

Datatype Properties: *itil:situationDescription* and *itil:situationName*

Class: Event

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:Event* is a dynamic situation in which the state of the real-world changes; each instance is something one would say ‘happens’. The *oc:Event*(s) are intangible because they are changes per se, not tangible objects that effect and undergo changes.

Generalization: *oc:Situation*

Relation to ITIL: We use the OpenCyc concept *oc:Event* in order to take advantage of existing upper ontologies. The *oc:Event* is the parent class of *oc:Action*.

Object Properties: *itil:inEvent*, *oc:subEvents* and inherited from *oc:Situation*

Datatype Properties: Inherited from *oc:Situation*

Class: Action

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:Action* is the collection of events that are carried out by some 'doer'. Instances of *oc:Action* include any event in which one or more actors effect some change in the (tangible or intangible) state of the real-world, typically by an expenditure of effort or energy. Note that it is not required that any tangible object be moved, changed, produced, or destroyed for an action to occur; the effects of an action might be intangible (such as a change in a bank balance or the intimidation of a subordinate). Note also that the doer of an action need not be (for example, a falling rock that dents a car's roof). Depending upon the context, doers of actions might be animate or inanimate, conscious or non conscious.

Generalization: *oc:Event*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use we use the OpenCyc concept *oc:Action* as the super class for all concrete *oc:Action* types in the ITSM model.

Object Properties: *oc:performedBy* and inherited from *oc:Event*

Datatype Properties: Inherited from *oc:Event*

Class: PurposefulAction

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:PurposefulAction* is an *oc:Action* consciously, volitionally, and purposefully done by at least one actor.

Generalization: *oc:Action*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:PurposefulAction* to classify activities in an ITIL workflow process (i.e., the set of events, the order in which they must be performed, and the performers who participate in the process) and to classify service events associated with the ITSM model.

Object Properties: Inherited from *oc:Action*

Datatype Properties: Inherited from *oc:Action*

Class: BpmnDiagram

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:BpmnDiagram* is the Workflow representation in form of a BPMN diagram which is composed of messages (*wf:MessagingEdge*) and pools (*wf:Pool*).

Generalization: *oc:PurposefulAction*, *wf:Identifiable* and *wf:ArtifactsContainer*

Relation to ITIL: We use the Workflow concept *wf:BpmnDiagram* in order to relate the business process flow to the *itil:Activity* that defines it. In our modeling approach for ITSMSs, the *wf:BpmnDiagram* is considered a subclass of *oc:PurposefulAction* and is the parent class of *itil:Activity*.

Object Properties: *wf:diagramComposedOf* and inherited from *oc:PurposefulAction*, *wf:Identifiable* and *wf:ArtifactsContainer*

Datatype Properties: *wf:diagramAuthor*, *wf:diagramTitle* and inherited from *oc:PurposefulAction*, *wf:Identifiable* and *wf:ArtifactsContainer*

Class: Activity

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:Activity* is a set of actions designed to achieve a particular result. The *itil:Activity* is usually defined as part of processes or plans, and it is documented in procedures.

Generalization: *wf:BpmnDiagram*

Relation to ITIL: *ITIL V3: Glossary of Terms and Definitions* (Activity definition). In our modeling approach for ITSMSs, the *itil:Activity* is a *wf:BpmnDiagram* that contains the workflow of an *itil:Process*. Following the approach defined in [Ferrario & Guarino, 2009] we present an *itil:Activity* as the service process that implements the service, i.e., the actions that ultimately lead to service production performed by the IT service provider. These activities are carried out and coordinated by the specifications as part of a business process, during which documents or information are passed from one participant to another, according to a set of procedural rules.

Object Properties: *itil:coordinatedBySpecification*, *itil:implementedByApplication* and inherited from *wf:BpmnDiagram*

Datatype Properties: Inherited from *wf:BpmnDiagram*

Class: ServiceEvent

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:ServiceEvent* is an event in which one or more agents (related to the event via the predicate *oc:providerOfService*) do something for one or more other agents (related to the event via the predicate *oc:recipientOfService*). An *oc:ServiceEvent* may involve maintenance, repair, or refurbishing of some object belonging to the recipient(s) of the service (including care of his/her person), or it may involve gathering or transmitting information, providing advice, entertainment, transportation, etc. to the recipient(s). The *oc:ServiceEvent*s may or may not be done for payment. Those done for payment are instances of *oc:ServiceProduct*.

Generalization: *oc:PurposefulAction*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:ServiceEvent* for the classification of the different service products associated with the ITSM model.

Object Properties: *oc:providerOfService*, *oc:recipientOfService* and inherited from *oc:PurposefulAction*

Datatype Properties: Inherited from *oc: PurposefulAction*

Class: ServiceProduct

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:ServiceProduct* is the collection of all *oc:ServiceEvent*(s) for which payment is made.

Generalization: *oc:ServiceEvent*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:ServiceProduct* for the classification of IT services, and events that are managed by specific ITIL processes.

Object Properties: Inherited from *oc:ServiceEvent*

Datatype Properties: Inherited from *oc:ServiceEvent*

Class: Event

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: An *itil:Event* is any detectable or discernible occurrence that has significance for the management of the IT infrastructure or the delivery of an IT service and evaluation of the impact a deviation might cause to the services. The *itil:Event*(s) are typically notifications created by an *itil:ITService*, *itil:CI* or monitoring tool and they have an *itil:Lifecycle*. An *itil:Event* typically requires IT Operations personnel to

take actions, and often lead to incidents being logged. Events occur continuously, but not all of them are detected or registered. It is therefore important that everybody involved in designing, developing, managing and supporting IT services and the IT infrastructure that they run on understands what types of events need to be detected and considered *itil:Event*s. Also, activities undertaken to manage a specific *itil:Event* should be documented.

There are many different types of *itil:Event*s, for example:

- Events that signify regular operation:
  - Notification that a scheduled workload has completed.
  - A user has logged in to use an application.
  - An e-mail has reached its intended recipient.
- Events that signify an exception:
  - A user attempts to log on to an application with the incorrect password.
  - An unusual situation has occurred in a business process that may indicate an exception requiring further business investigation (for example, a Web page alert indicates that a payment authorization site is unavailable – impacting financial approval of business transactions).
  - A device's CPU is above the acceptable utilization rate.
  - A PC scan reveals the installation of unauthorized software.
- Events that signify unusual, but not exceptional, operation. These are an indication that the situation may require closer monitoring. In some cases the condition will resolve itself, for example in the case of an unusual combination of workloads – as they are completed, normal operation is restored. In other cases, operator intervention may be required if the situation is repeated or if it continues for too long. These rules or policies are defined in the Monitoring and Control Objectives for that device or service. Examples of this type of event are:
  - A server's memory utilization reaches within 5% of its highest acceptable performance level.
  - The completion time of a transaction is 10% longer than normal.

Generalization: *oc:ServiceProduct*

Relation to ITIL: *ITIL Service Operation* p. 67, p. 69, p. 91 and p. 373-374 (Event definition). We use the *itil:Event* class to specify all the events that are included in an *itil:Process* for proactive and reactive event management. Since *itil:ITServiceProvider*(s) wants to make sure that the *itil:ITService* will remain available to meet the *itil:SLA*(s), the IT employee must take actions when an event occurs. According to ITIL, some events could be part of different processes, or even a combination of two or more of them. Therefore an *itil:ITServiceProvider* must decide and indicate what *itil:Process* (or processes) is going to manage a specific *itil:Event*.

Object Properties: *itil:hasEventCategoryCode*, *itil:hasEventLifecycle*, *itil:hasEventType*, *itil:hasManagedEventType*, *itil:hasTechnicalManagementType*, *itil:managedByProcess*, *itil:undertakesActivity* and inherited from *oc:ServiceProduct*

Datatype Properties: Inherited from *oc:ServiceProduct*

Class: Incident

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); Pilot project documentation.

Description: An *itil:Incident* is an unplanned interruption to an *itil:ITService* or reduction in the quality of an *itil:ITService*. Failure of an *itil:CI* that has not yet impacted the *itil:ITService* is also an *itil:Incident*, for example failure of one disk from a mirror set.

The *itil:IncidentManagement* process includes any event which disrupts, or which could disrupt, a service. This includes *itil:Event*(s) which are communicated directly by users, either through the itil:SERVICE_DESK (*itil:RoleType* instance) or through an interface from the *itil:EventManagement* process to incident management tools.

The *itil:Incident*(s) can also be reported and/or logged by technical staff (if, for example, they notice something untoward with a hardware or network component they may report or log an incident and refer it to the itil:SERVICE_DESK). This does not mean, however, that all *itil:Event*(s) are *itil:Incident*(s). Many classes of *itil:Event*(s) are not related to disruptions at all, but are indicators of normal operation or are simply informational.

Each *itil:Incident* may have links to the *itil:Event*(s) concerned (*oc:subEvents* property) (for example, relationship with other *itil:Incident*(s), *itil:Problem*(s), *itil:Change*(s) or *itil:KnownError*(s)), and to the *itil:Activity* undertaken to resolve the *itil:Incident* (*itil:undertakesActivity* property).

Generalization: *itil:Event*

Relation to ITIL: *ITIL Service Operation*, p. 77, p. 86, p. 91, p. 101 and p. 376 (Incident definition). In our pilot project, an *itil:Incident* is allocated to different support groups/persons that could resolve the *itil:Incident* (*oc:performedBy* property).

Object Properties: *itil:hasIncidentRecord* and inherited from *itil:Event*

Datatype Properties: Inherited from *itil:Event*

Class: ServiceRequest

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); Pilot project documentation.

Description: An *itil:ServiceRequest* is a request from an *itil:User* for information or advice, or for a standard change or for access to an *itil:ITService*. For example to reset a password, or to provide standard *itil:ITService*(s) for a new *itil:User*. To be an *itil:ServiceRequest*, it is normal for some prerequisites to be defined and met (e.g., needs to be proven, repeatable, pre-approved, proceduralized). The

itil:ServiceRequest(s) do not represent a disruption to agreed *itil:ITService*, but are a way of meeting the customer's needs and may be addressing an agreed target in an *itil:SLA*. The *itil:ServiceRequest*(s) are usually handled by the itil:SERVICE_DESK (*RoleType* instance), and do not require an *itil:RFC* to be submitted.

Generalization: *itil:Event*

Relation to ITIL: *ITIL Service Operation*, p. 36, p. 86 and p. 390 (Service Request definition). In our pilot project, each *itil:ServiceRequest* has a type and it is allocated to different support groups/persons that could deal with the *itil:ServiceRequest* (*oc:performedBy* property).

Object Properties: Inherited from *itil:Event*

Datatype Properties: Inherited from *itil:Event*

Class: RFC

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: A *Request for Change* (RFC) is a formal proposal for a change to be made. An *itil:RFC* includes details of the proposed change, and may be recorded on paper or electronically. Authorized *itil:RFC*(s) should be passed to the relevant technical groups for building of the changes. Each service change arrives into service evaluation and qualification in the form of an *itil:RFC* from the *itil:ChangeManagement* process.

All *itil:RFC*(s) received should be logged and allocated an identification number (in chronological sequence). Where *itil:RFC*(s) are submitted in response to a trigger such as a resolution to an *itil:ProblemRecord*, it is important that the reference number of the triggering document is retained to provide traceability.

Generalization: *itil:Event*

Relation to ITIL: *ITIL Service Operation*, p. 94, p. 102, p. 246 and p. 388 (Request for Change definition).

Object Properties: *itil:hasChangeRecord*, *itil:proposesChange* and inherited from *itil:Event*

Datatype Properties: Inherited from *itil:Event*

Class: Change

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO); Pilot project documentation.

Description: An *itil:Change* represents the addition, modification or removal of authorized, planned or supported service or service component and its associated documentation.

Generalization: *itil:Event*

Relation to ITIL: *ITIL Service Transition*, p. 78 and p. 371 (Change definition). In our pilot project, changes are considered urgent when they restore a service after the identification of a problem and pre-approved when the approval of the *Change Advisory Board* (CAB) is not required.

Object Properties: Inherited from *itil:Event*

Datatype Properties: *itil:urgentChange*, *itil:preApprovedChange* and inherited from *itil:Event*

Class: Problem

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); Pilot project documentation.

Description: An *itil:Problem* is the cause of one or more incidents. The cause is not usually known at the time an *itil:ProblemRecord* is created, and the *itil:ProblemManagement* process is responsible for further investigation.

Generalization: *itil:Event*

Relation to ITIL: *ITIL Service Operation*, p. 111 and p. 383 (Problem definition). In our pilot project, each *itil:Problem* is allocated to an specfic support group/person that could resolve the *itil:Problem* (*oc:performedBy* property).

Object Properties: *itil:hasProblemRecord* and inherited from *itil:Event*

Datatype Properties: Inherited from *itil:Event*

Class: KnownError

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: An *itil:KnownError* is an *itil:Problem* that has a documented root cause and a workaround. The workaround describes how to reduce or eliminate the impact of an *itil:Problem* for which a full resolution is not yet available. For example, by restarting a failed *itil:CI*. The *itil:KnownError*(s) are created and managed throughout their *itil:Lifecycle* by the *itil:ProblemManagement* process. The *itil:KnownError*(s) may also be identified by development or suppliers.

The known error record (*itil:ProblemRecord*) should hold exact details of the fault and the symptoms that occurred, together with precise details of any workaround or resolution action that can be taken to restore the service and/or resolve the problem. An *itil:Incident* count will also be useful to determine the frequency with which *itil:Incident*(s) are likely to recur and influence priorities, etc.

Generalization: *itil:Problem*

Relation to ITIL: *ITIL Service Operation*, p. 123, p. 378 (Known Error definition) and p. 395 (Workaround definition).

Object Properties: Inherited from *itil:Problem*

Datatype Properties: Inherited from *itil:Problem*

Class: IncidentRecord

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: An *itil:IncidentRecord* represents a record containing the details of an *itil:Incident*. Each *itil:IncidentRecord* documents the *itil:Lifecycle* of a single *itil:Incident* and the responsible (group/person) of the resolution of the reported incident, i.e., the *oc:Agent-Generic* that records the *itil:Incident*.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Operation*, p. 86 and p. 376 (Incident Record definition).

Object Properties: *itil:hasIncidentGroup*, *itil:hasIncidentStatus* and *itil:hasIncidentResponsible*

Datatype Properties: *itil:incidentImpact*, *itil:incidentLevel*, *itil:incidentPriority*, *itil:incidentResolution*, *itil:incidentResolutionDatetime*, *itil:incidentStartDatetime* and *itil:incidentUrgency*

Class: ChangeRecord

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Transition*; The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:ChangeRecord* holds the full history of the change, incorporating information from the *itil:RFC* and subsequently recording agreed parameters such as priority and authorization, implementation and review information. There may be many different types of *itil:ChangeRecord*(s) used to record different types of *itil:Change*. The documentation should be defined during the process design and planning stage.

An *itil:ChangeRecord* is created for every *itil:RFC* that is received, even those that are subsequently rejected. The *itil:ChangeRecord*(s) should reference the *itil:CI*(s) that are affected by the requested change. The *itil:ChangeRecord*(s) are stored in the CMS.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Transition*, p. 93-94. *ITIL V3: Glossary of Terms and Definitions* (Change Record definition).

Object Properties: *itil:affectsCI*

Datatype Properties: none

Class: ProblemRecord

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:ProblemRecord* represents a record containing the details of an *itil:Problem*. Each *itil:ProblemRecord* documents the *itil:Lifecycle* of a single *itil:Problem*.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Operation*, p. 86. *ITIL V3: Glossary of Terms and Definitions* (Problem Record definition).

Object Properties: none

Datatype Properties: none

Class: ITService

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:ITService* is a service provided to one or more *itil:Customer*(s) by an *itil:ITServiceProvider*. That is, an *itil:ITService* represents the means of delivering value to customers by facilitating outcomes, and it should be defined in an *itil:SLA*. An *itil:ITService* is based on the use of information technology and supports the customer's business processes (in fact, many business processes rely on IT services). As customers and suppliers become the direct users of IT services, the expectations and *service level requirements* (SLRs) have become more demanding, requiring a value net approach. An outcome-based definition of service moves IT service providers beyond business-IT alignment towards business-IT integration. An *outcome* is the result of carrying out an activity; following a process; delivering an IT service etc. The term outcome is used to refer to intended results, as well as to actual results.

Each *itil:ITService* defines a set of *itil:Metric*(s) whose purpose is to measure the quality and effectiviness of that service in order to take timely actions that make sure service are delivered in line with business needs. These are the metrics that really matter in order to demonstrate the value of the service and for the operation in a cycle of continuous improvement. Also, *itil:ITService*(s) are managed according to an *itil:ServiceLifecycle* and they are composed of *itil:Application*(s) and other *itil:CI*(s) necessary to support the provision of the *itil:ITService* to the business.

Generalization: *oc:ServiceProduct*

Relation to ITIL: *ITIL Service Strategy*, p. 36, p. 81, p. 340 (Application definition), p. 343 (Business Process definition), p. 354 (IT Service definition) and p. 358-359

(Outcome definition). In our modeling approach for ITSMSs, just like the approach of [Ferrario & Guarino, 2009], we consider *itil:ITService*(s) to be events based on agreements. In [Ferrario & Guarino, 2009], services are modeled by means of a layered set of interrelated activities (events), each one with its own participants and spatio-temporal location. Therefore, *itil:ITServicesProvider*(s) deliver not the service itself, but its content: *"the actions to be performed in the interest of the customer."*

Object Properties: *itil:definesMetric*, *itil:doneForCustomer* (subproperty of *oc:recipientOfservice*), *itil:hasApplication*, *itil:hasCustomerReq*, *itil:hasServiceLifecycle*, *itil:inServicePortfolio*, *itil:managesCI*, *itil:supportsPBA* and inherited from *oc:ServiceProduct*

Datatype Properties: *itil:internalService, itil:serviceImportanceCode*, *itil:serviceUsers*, *itil:visibleToCustomer* and inherited from *oc:ServiceProduct*

Class: CoreService

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO);); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:CoreService* represents an *itil:ITService* that delivers the basic outcomes desired by the *itil:Customer*. The *itil:CoreService*(s) represent the value that the *itil:Customer* wants and for which they are willing to pay. The *itil:CoreService*(s) anchor the value proposition for the *itil:Customer* and provide the basis for their continued utilization and satisfaction. The *itil:SupportingService*(s) either enable or enhance the value proposition. Enabling services are basic factors and enhancing services are excitement factors.

Generalization: *itil:ITService*

Relation to ITIL: *ITIL Service Strategy*, p. 207. *ITIL V3: Glossary of Terms and Definitions* (Core Service definition).

Object Properties: *itil:hasSupportingService* and inherited from *itil:ITService*

Datatype Properties: Inherited from *itil:ITService*

Class: SupportingService

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:SupportingService* is an *itil:ITService* that enables or enhances an *itil:CoreService*. For example, the *itil:DirectoryService* or the *itil:BackupService* service instances.

Generalization: *itil:ITService*

Relation to ITIL: *ITIL Service Strategy*, p. 207. *ITIL V3: Glossary of Terms and Definitions* (Supporting Service definition).

Object Properties: Inherited from *itil:ITService*

Datatype Properties: Inherited from *itil:ITService*

Class: PBA

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: *Pattern of Business Activity* (PBA) defines dynamics of a business and includes interactions with customers, suppliers, partners and other stakeholders. An *itil:PBA* represents a workload profile of one or more business activities, where workload is the resources required to deliver an identifiable part of an *itil:ITService*. Workloads may be categorized by users, groups of users, or functions within the *itil:ITservice*. This is used to assist in analyzing and managing the capacity, performance and utilization of *itil:CI*(s) and *itil:ITService*(s). The term workload is sometimes used as a synonym for the design concept throughput. Throughput is a measure of the number of transactions, or other operations, performed in a fixed time. For example, 5,000 e-mails sent per hour, or 200 disk I/Os per second.

An *itil:PBA* is used to help the *itil:ITServiceProvider* understand and plan for different levels of business activity. The *itil:ITService*(s) often directly support *itil:PBA*. Since *itil:PBA*(s) generate revenue, income and costs they account for a large proportion of business outcomes.

The *itil:PBA*(s) are identified, codified, and shared across process for clarity and completeness of detail. One or more attributes such as frequency, volume, location and duration describe business activity. They are associated with requirements such as security, privacy and latency or tolerance for delays. This profile of business activity can change over time with changes and improvements in business processes, people, organization, applications and infrastructure. The *itil:PBA*(s) are placed under change control.

Each *itil:PBA* has to be substantially different from another *itil:PBA* in order to be coded with a unique reference. Codifying patterns helps multidimensional analysis, using criteria such as likeness and nearness. This provides efficiency and robustness in developing a catalogue of patterns with simplification and standardization to reduce the number of patterns, make analysis easier, and avoid complicated solutions.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 204-205 and p. 359 (Pattern of Business Activity definition), p. 370 (Throughput definition) and p. 372-373 (Workload definition).

Object Properties: none

Datatype Properties: *itil:pbaDescription* and *itil:pbaName*

Class: UP

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: *User Profile* (UP) is a pattern of user demand for IT Services. Each *itil:UP* includes one or more *itil:PBA*. That is, *itil:UP*(s) are constructed using one or more predefined *itil:PBA*(s). Pattern matching using *itil:PBA* and *itil:UP* ensure a systematic approach to understanding and managing demand from customers. They also require customers to better understand their own business activities and view them as consumers of services and producers of demand. When they are used to communicate demand, service providers have the information necessary to sort and serve the demand with appropriately matched services, service levels, and service assets. This leads to improved value for both customers and service providers by eliminating waste and poor performance.

The *itil:UP*(s) are based on roles and responsibilities within organizations for people, and functions and operations for processes and applications. Business processes and applications are treated as users in many business contexts. Many processes are not actively executed or controlled by staff or personnel. Process automation allows for processes to consume services on their own. Processes and applications can have user profiles. Whether they should is a matter of judgment.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 205-207 and p. 359 (User Profile definition).

Object Properties: *itil:includesPBA*

Datatype Properties: *itil:upDescription* and *itil:upName*

Class: SLR

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: A *Service Level Requirement* (SLR) is a customer requirement for an aspect of an *itil:ITService*. A set of targets and responsibilities should be documented and agreed within an *itil:SLR* for each proposed new or changed *itil:ITService*. An *itil:SLR* is based on business objectives and it is used to negotiate agreed *itil:ServiceLevelTarget*(s).

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Design*, p. 127 and p. 442 (Service Level Requirement definition).

Object Properties: *itil:usedForNegotiation*

Datatype Properties: *itil:slrBusinessObjective*, *itil:slrDescription* and *itil:slrName*, *itil:slrResponsibility* and *itil:slrTarget*

Class: ServiceLevelTarget

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: An *itil:ServiceLevelTarget* is a commitment that is documented in an *itil:SLA*. The *itil:ServiceLevelTarget*(s) are based on *itil:SLR*(s), and are needed to ensure that the *itil:ServiceDesign* is fit for purpose. The *itil:ServiceLevelTarget*(s) should be smart, and are usually based on *itil:KPI*(s).

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Design*, p. 442 (Service Level Target definition).

Object Properties: *itil:basedOnKPI* and *itil:basedOnSLR*

Datatype Properties: *itil:targetDescription* and *itil:targetName*

Class: ServicePortfolio

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The *itil:ServicePortfolio* is the complete set of *itil:ITService*(s) that are managed by an *itil:ITServiceProvider*. The *itil:ServicePortfolio* is used to manage the entire lifecycle of all *itil:ITService*(s), and includes three categories (*itil:ServicePortfolioType* enumeration class): Service Pipeline (proposed or in development), Service Catalog (live or available for deployment) and Retired Services. In other words, *itil:ServicePortfolio* represents the commitments and investments made by an *itil:ITServiceProvider* across all customers and market spaces. It represents present contractual commitments, new service development, and ongoing service improvement plans initiated by *itil:CSI*. The *itil:ServicePortfolio* also includes third-party services, which are an integral part of service offerings to customers. Some third-party services are visible to the customers while others are not.

Changes to *itil:ServicePortfolio* are governed by policies and procedures. The *itil:ServicePortfolio*(s) instill a certain financial discipline necessary to avoid making investments that will not yield value.

The *itil:ServicePortfolio* represents all the resources presently engaged or being released in various phases of the *itil:ServiceLifecycle*. Each phase requires resources for completion of projects, initiatives and contracts. This is a very important governance aspect of the *itil:ServicePortfolioManagement* process. Entry, progress and exit are approved only with approved funding and a financial plan for recovering costs or showing profit as necessary. The *itil:ServicePortfolio* should have the right mix of services in the pipeline and catalog to secure the financial viability of the IT service provider. The Service Catalog is the only part of the *itil:ServicePortfolio* that recovers costs or earns profits.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 116-117 and p. 367 (Service Portfolio definition).

Object Properties: *itil:detailsITService* and *itil:hasServicePortfolioType*

Datatype Properties: *itil:portfolioDescription* and *itil:portfolioName*

Class: ServicePackage

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:ServicePackage* is detailed description of an *itil:ITService* that is available to be delivered to *itil:Customer*(s). The *itil:ServicePackage*(s) come with one or more *itil:SLP*(s) and one or more *itil:CoreService*(s) and *itil:SupportingService*(s). An *itil:ServicePackage* is considered a core *itil:ServicePackage* when it represents a detailed description of an *itil:CoreService* that may be shared by two or more *itil:ServiceLevelPackage*(s).

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 209. *ITIL V3: Glossary of Terms and Definitions* (Core Service Package definition and Service Package definition). Note that an *itil:ServicePackage* only can be associated with more that one *itil:SLP* and *itil:ITService* when the service is representing a *line of service* (LOS). A LOS is an *itil:CoreService* or *itil:SupportingService* that has multiple *service level packages* (SLP). A LOS is managed by a product manager and each SLP is designed to support a particular market segment.

Object Properties: *itil:hasITService*, *itil:hasSLP*

Datatype Properties: *itil:isCorePackage*, itil:*packageDescription* and *itil:packageName*

Class: SLP

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:SLP* is a defined level of *utility* and *warranty* for a particular *itil:ServicePackage*. Each *itil:SLP* is designed to meet the needs of a particular *itil:PBA*. In other words, *itil:SLP*(s) are effective in developing *itil:ServicePackage*(s) for providing value to a segment of users with *utility* and *warranty* appropriate to their needs and in a cost-effective way. *Utility* is the functionality offered by a product or service to meet a particular need. *Utility* is often summarized as 'what it does'. *Warranty* is a promise or guarantee that a product or service will meet its agreed requirements. *Warranty* is often summarized as 'how well it does it'.

The *itil:SLP*(s) are associated with a set of service levels, pricing policies, and a service package. Combinations of *itil:ServicesPackage*(s) and *itil:SLP*(s) are used to serve customer segments with differentiated value. Common attributes of *itil:SLP*(s) are subsumed into the supporting *itil:ServicePackage*(s). This is like the popular game of

Tetris in which the bottom-most layer of bricks gets subsumed when all its gaps are filled with the falling bricks. This follows the principle of modularity to reduce complexity, increase asset utilization across *itil:SLP*(s), and to reduce the overall cost of services. The *itil:ServicePackage*(s) and *itil:SLP*(s) are loosely coupled to allow for local optimization while maintaining efficiency over the entire supported service catalog. Improvements made to *itil:ServicePackage*(s) are automatically available to all *itil:SLP*(s) following the principle of inheritance and encapsulation.

The *itil:ServicePackage*(s) and *itil:SLP*(s) are each made up of reusable components many of which themselves can be services. Other components include software applications, hardware, licenses, third-party services and public infrastructure services. Some service components are assets owned by customers.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 209-212, p. 366 (Service Level Package definition), p. 371 (Utility definition) and p. 372 (Warranty definition).

Object Properties: *itil:meetsPBA*

Datatype Properties: *itil:slpDescription* and *itil:slpName*

Class: Agent-Generic

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:Agent-Generic* is a specialization of *oc:SomethingExisting*. An *oc:Agent-Generic* is a being that has desires or intentions, and the ability to act on those desires or intentions.

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:Agent-Generic* for the classification of the agents that participate in the ITSM model.

Object Properties: *oc:responsibleFor*

Datatype Properties: *itil:agentDescription* and *itil:agentName*

Class: IntelligentAgent

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:IntelligentAgent* is a specialization of *oc:Agent-Generic* and *oc:InformationStore*. An agent is an *oc:IntelligentAgent* if and only if it is capable of knowing and acting, and capable of employing its knowledge in its actions. An *oc:IntelligentAgent* typically knows about certain things, and its beliefs concerning those things influences its actions. As with agents generally, an *oc:IntelligentAgent*

might either be a single individual, such as a person, or a group consisting of two or more individual agents, such as a business or government organization.

Generalization: *oc:Agent-Generic*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:IntelligentAgent* for the classification of some ITIL concepts such as *itil:Customer* or *itil:ITServiceProvider* and to assign the roles to the agents that participates in the management of an *itil:ITService*.

Object Properties: *itil:hasRoleRelation* and inherited from *oc:Agent-Generic*

Datatype Properties: Inherited from *oc:Agent-Generic*

Class: ActorSlot

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:ActorSlot* is a collection of binary predicates; a specialization of *oc:Role*. Each instance of *oc:ActorSlot* relates some instance of *oc:Event* to a temporal thing involved in that event (here called a 'participant', although the thing in question might not be playing an active role in the event). The first argument of every instance of *oc:ActorSlot* is constrained to be an instance of some specialization of *oc:Event*, and the second argument is constrained to be an instance of some specialization of *oc:SomethingExisting* (e.g., *oc:Agent-Generic*).

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:ActorSlot* for the definition of the *itil:RoleRelation* class that relates an *oc:PurposefulAction* to an *oc:IntelligentAgent*.

Object Properties: none

Datatype Properties: none

Class: RoleRelation

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:RoleRelation* is used to build a RACI chart that is needed to identify/define, on the one hand, the functional roles and, on the other hand, responsibilities of the various roles (i.e., RACI codes). In some organizations this could be a full-time individual and in others it could be several people, or it could be a part-time role. In smaller organizations many of these roles may be performed by a single person. This will depend on the size and volatility of the organization. The roles or job titles often vary between organizations. However, what is important is that the roles,

responsibilities, processes, dependencies and interfaces are clearly defined and scoped for each individual organization.

Generalization: *oc:ActorSlot*

Relation to ITIL: *ITIL Service Design*, p. 323-339. An *oc:IntelligentAgent* can participate in *oc:PurposefulAction*(s) using different roles and assigned with different RACI codes.

Object Properties: *itil:roleAction*, *itil:roleRACI*, *itil:roleCode* and inherited from *oc:ActorSlot*

Datatype Properties: Inherited from *oc:ActorSlot*

Class: Organization

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:Organization* is the collection of all organizations. Each instance of *oc:Organization* is a group whose group-members are instances of *oc:IntelligentAgent*. In each instance of *oc:Organization*, certain relationships and obligations exist between the members of the *oc:Organization*, or between the *oc:Organization* and its members. Instances of *oc:Organization* include both informal and legally constituted organizations. Each instance of *oc:Organization* can undertake projects, enter into agreements, own property, and do other tasks characteristic of agents.

Generalization: *oc:IntelligentAgent*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:Organization* for the classification of some ITIL concepts such as *itil:Customer* or *itil:ITServiceProvider* (subclassing from *oc:ServiceOrganization*).

Object Properties: *oc:hasMembers* and inherited from *oc:IntelligentAgent*

Datatype Properties: Inherited from *oc:IntelligentAgent*

Class: Customer

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:Customer* is someone who buys goods or services. The *itil:Customer* of *an itil:ITServiceProvider* is the person or group that defines and agrees the *itil:ServiceLevelTarget*(s).

Generalization: *oc:Organization*

Relation to ITIL: *ITIL Service Strategy*, p. 348 (Customer definition). In our modeling approach for ITSMSs, the *itil:ServiceLevelTarget*(s) are associated with *itil:SLA*(s). The term customer is also sometimes informally used to mean user. However, as mentioned

earlier, *itil:User*s are distinct from *itil:Customer*s, as some *itil:Customer*s do not use the IT service directly.

Object Properties: Inherited from *oc:Organization*

Datatype Properties: Inherited from *oc:Organization*

Class: ServiceOrganization

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: A *oc:ServiceOrganization* is an organization whose main function is to provide some service or services (as opposed, for example, to mainly selling goods or manufacturing products). An *oc:ServiceOrganization* might or might not be a subsidiary or department in some larger organization; it might or might not be a for-profit organization.

Generalization: *oc:Organization*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:ServiceOrganization* for the classification of organizations that are providers of services.

Object Properties: Inherited from *oc:Organization*

Datatype Properties: Inherited from *oc:Organization*

Class: ITServiceProvider

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO); OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: An *itil:ITServiceProvider* provides *itil:*ITService(s) to an *itil:Customer* within a business. A business is an overall corporate entity or organization formed of a number of business units, i.e., segments of the business that has their own plans, metrics, income and costs. In the context of ITSM, the term business includes public sector and not-for-profit organizations, as well as companies. The *itil:ITServiceProvider* may be part of the same business as its customer (internal service provider), or part of another business (external service provider).

According to ITIL V3, an *itil:SLA* is defined as a written agreement between an *itil:ITServiceProvider* and the *itil:Customer*(s) that documents agreed service levels for an *itil:ITService*. The *itil:ITServiceProvider* should be aware that *itil:SLA*(s) are widely used to formalize service-based relationships, both internally and externally, and that while conforming to the definition above, these agreements vary considerably in the detail covered.

Generalization: *oc:ServiceOrganization*

Relation to ITIL: *ITIL Service Strategy*, p. 343 (Business definition), p. 344 (Business Unit definition). *ITIL Service Design*, p. 269.

Object Properties: *itil:managesServicePortfolio* and inherited from *oc:ServiceOrganization*

Datatype Properties: *itil:internalProvider* and inherited from *oc:ServiceOrganization*

Class: OrganizationOfPeopleOnly

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:OrganizationOfPeopleOnly* is an *oc:Organization* each of whose members (see the predicate *oc:hasMembers*) is an instance of *oc:Person*. Examples of *oc:OrganizationOfPeopleOnly* include a human nuclear family, a carpool, or a sports team. Negative examples include *oc:UnitedNationsOrganization* or *oc:OrganizationOfAmericanStates*.

Generalization: *oc:Organization*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:OrganizationOfPeopleOnly* for the classification of groups or team of people that are participating in an IT service delivery process.

Object Properties: Inherited from *oc:Organization*

Datatype Properties: Inherited from *oc:Organization*

Class: Shift

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: An *itil:Shift* is a group or team of people who carry out a specific role for a fixed period of time. For example there could be four *itil:Shift*(s) of IT operations control personnel to support an IT service that is used 24 hours a day.

Generalization: *oc:OrganizationOfPeopleOnly*

Relation to ITIL: *ITIL Service Operation*, p. 259-260 and p. 390 (Shift definition).

Object Properties: Inherited from *oc:OrganizationOfPeopleOnly*

Datatype Properties: Inherited from *oc:OrganizationOfPeopleOnly*

Class: SupportGroup

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: An *itil:SupportGroup* is a group of people with technical skills. The *itil:SupportGroup*(s) provide the technical support needed by all of the ITSM processes (*itil:Process*(s)).

Generalization: *oc:OrganizationOfPeopleOnly*

Relation to ITIL: *ITIL Service Operation*, p. 392 (Support Group definition).

Object Properties: Inherited from *oc:OrganizationOfPeopleOnly*

Datatype Properties: Inherited from *oc:OrganizationOfPeopleOnly*

Class: User

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:User* is a person who uses the IT service on a day-to-day basis. The *itil:User* class is distinct from the *itil:Customer* class, as some *itil:Customer*s do not use the IT service directly.

Generalization: *oc:OrganizationOfPeopleOnly*

Relation to ITIL: *ITIL Service Strategy*, p. 371 (User definition).

Object Properties: Inherited from *oc:OrganizationOfPeopleOnly*

Datatype Properties: Inherited from *oc:OrganizationOfPeopleOnly*

Class: SuperUser

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: An *itil:SuperUser* is an *itil:User* who helps other users, and assists in communication with the itil:SERVICE_DESK (*itil:RoleType* instance) or other parts of the *itil:ITServiceProvider*. The *itil:SuperUser*(s) typically provide support for minor *itil:Incident*(s) and training. Many organizations find it useful to appoint or designate a number of *itil:SuperUser*(s) throughout the user community, to act as liaison points with IT in general and the itil:SERVICE_DESK in particular.

Generalization: *itil:User*

Relation to ITIL: *ITIL Service Operation*, p. 210-211. *ITIL V3: Glossary of Terms and Definitions* (Super User definition).

Object Properties: Inherited from *itil:User*

Datatype Properties: Inherited from *itil:User*

Class: Metric

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:Metric* is something that is measured and reported to help manage a process, IT service or activity.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Strategy*, p. 357 (Metric definition). In our modeling approach for ITSMSs, we use metrics to measure the *itil:Process*(s). The *itil:Metric*(s) provide the feedback mechanism allowing management to steer, control and guide IT toward strategic objectives [Smith, 2008]. For example, 'Number and percentage of the incidents resolved remotely, without the need for a visit' is a metric that should be monitored and reported upon to judge the efficiency and effectiveness of the *Incident Management* process.

Object Properties: *itil:hasAnalyticalMetric, itil:hasMetricType*, *itil:includesMeasurement* and *itil:measures*

Datatype Properties: *itil:metricDescription*, *itil:metricName*, *itil:metricValue*

Class: OperationalMetric

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: An *itil:OperationalMetric* is a basic observation of operational events that provides live data from ITSM process (i.e., *itil:Process*) reporting and other infrastructure measurements and observations.

Generalization: *itil:Metric*

Relation to ITIL: *Measuring ITIL*, p. 20-21. In our modeling approach for ITSMSs, we use operational metrics to measure, for example, the number of IT changes that have been implemented, the number of incidents of some type that have occurred, the current peak utilization of components such as network lines or servers, or the availability of an application or system.

Object Properties: Inherited from *itil:Metric*

Datatype Properties: Inherited from *itil:Metric*

Class: KPI

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO); Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling -*

the IT Service Management Metrics That Matter Most to IT Senior Executives. Trafford Publishing; Pilot project documentation.

Description: An *itil:KPI* is an *itil:Metric* that is used to help manage an *itil:Process*, *itil:ITService* or *itil:Activity*. Many metrics may be measured, but only the most important of these are defined as *itil:KPI*(s) and used to actively manage and report on the process, IT service or activity. The *itil:KPI*(s) should be selected to ensure that efficiency, effectiveness, and cost effectiveness are all managed. Also, the provision of *itil:KPI*(s) is essential to supporting *itil:CSI*. The *itil:KPI*(s) are used to provide a basis for actionable management decisions. Each *itil:KPI* is trying to answer a question. While *itil:OperationalMetric*(s) are generally historical in nature, *itil:KPI*(s) are really the *"metrics that matter."* These *itil:KPI*(s) become the data inputs to analyze and identify improvement opportunities. For example, 'Increasing first-contact resolution' is a common *itil:KPI* for the *itil:IncidentManagement* process. In order to compute the *itil:KPI*, we must identify the metrics and measurements required. There are two basic kinds of *itil:KPI*, qualitative and quantitative. '10 percent increase in customer satisfaction rating for handling incidents over the next 6 months' is an example of a qualitative *itil:KPI* that requires the metrics 'Original customer satisfaction score for handling incidents' and 'Ending customer satisfaction score for handling incidents', and the measurements 'Incident handling survey score' and 'Number of survey scores.' On the other hand, '10 percent reduction in the costs of handling printer incidents' is an example of quantitative *itil:KPI* that requires the metrics 'Original cost of handling a printer incidents', 'Final cost of handling a printer incidents' and 'Cost of the improvement effort', and the measurements 'Time spent on the incident by first-level operative and their average salary', 'Time spent on the incident by second-level operative and their average salary', 'Time spent on Problem Management activities by second-level operative and their average salary', 'Time spent on the training first-level operative on the workaround', 'Cost of a service call to third-party vendor' and 'Time and material from third-party vendor.' Note that all *itil:KPI*(s) require calculation. The *itil:CapacityManagement* process, for example, is an *itil:OperationalMetric* (observed from a process audit) and simply carries over as an *itil:KPI*.

Generalization: *itil:Metric*

Relation to ITIL: *ITIL Continual Service Improvement*, p. 290 (Key Performance Indicator definition), p. 41 and p. 99-100. *Measuring ITIL*, p. 20-22. For example, in our pilot project, *itil:Incident_resolution_rate* and *itil:Customer_satisfaction_level* are instances of *itil:KPI* for the *itil:Process_IncidentManagement* instance. In our approach, according to Steinberg [2006], the *itil:KPI*(s) are calculated or derived from one or more *itil:OperationalMetric*(s). For example, in our pilot project, the *itil:KPI* of *itil:Incident_resolution_rate* is the result of dividing *itil:Number_of_incidents_resolved_within_agreed_service_levels* by *itil:Total_number_of_incidents* (instances of *itil:OperationalMetric*). The results of these calculations are then compared to an *itil:Tolerance* range to identify whether those results fall within acceptable levels.

Object Properties: *itil:fallsIntoToleranceRange*, *itil:questionBeingAnswered*, *itil:requiresOperationalMetric* and inherited from *itil:Metric*

Datatype Properties: Inherited from *itil:Metric*

Class: Tolerance

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing; Pilot project documentation.

Description: The *itil:Tolerance*(s) represent the boundaries for acceptable and non-acceptable *itil:KPI* values (i.e., service target and warning level). They should be set by the IT service manager and agreed by IT and business senior management. These are critical, as they form the basis for when management needs to take action or make a key decision. Tolerance values are based on desired service and performance levels that the business is willing to tolerate.

Generalization: *itil:Metric*

Relation to ITIL: *Measuring ITIL*, p. 20 and p. 23. In our modeling approach for ITSMSs, we use tolerances to associate *itil:Tolerance* values to *itil:KPI*(s). For example, in our pilot project, if the service target *itil:Tolerance* value for the *itil:KPI* of *itil:Average_Incident_Resolution_Hours* is 2.0 it means that the service target for this *itil:KPI* would be 2.0 hours. On the other hand, if the warning level *itil:Tolerance* value for the *itil:KPI* of *itil:Average_Incident_Resolution_Hours* is 3.5, it means that the performance of this *itil:KPI* would be considered acceptable as long as it is not higher than 3.5 hours. If it is higher, management actions may need to take place to raise the performance back to acceptable levels.

Object Properties: Inherited from *itil:Metric*

Datatype Properties: *itil:toleranceCode*, *itil:toleranceServiceTarget*, *itil:toleranceWarningLevel* and inherited from *itil:Metric*

Class: CSF

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO); OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO); Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing; Pilot project documentation.

Description: An *itil:CSF* is something that must happen if a process, project, plan, or IT service is to succeed. An *itil:CSF* is an *itil:Metric* that represents key operational performance requirements which indicate whether a process, IT service or activity is performing successfully from a customer or business perspective. One way to define *itil:CSF*(s) is by customer assets and the service archetypes. For example, in healthcare, IT service providers have extensive knowledge of hospital procedures, medical equipment, interactions between physicians, clinicians and pharmacists, insurance policies and privacy regulations. IT service providers present in market spaces related to the quality of outcomes in healthcare typically have physicians and clinicians on their payroll. Service strategies for the healthcare market spaces take into account the need to deal with users with highly specialized skills, special-purpose equipment, low tolerance

for error, and the need to balance security with usability of services. These are *itil:CSF*(s) for a cluster of market spaces related to healthcare. A subset of these *itil:CSF*(s) is shared by other market spaces such as military applications. The *itil:CSF*(s) can therefore span more than one market space. They represent opportunities for leveraging economies of scale and scope.

The *itil:KPI*(s) are used to measure the achievement of each *itil:CSF*. A recommended approach for deriving an *itil:CSF* is to first identify which *itil:KPI*(s) relate to it and then rate the *itil:CSF* based on the lowest valued observed in any one of those *itil:KPI*(s).

Generalization: *itil:Metric*

Relation to ITIL: *ITIL Continual Service Improvement*, p. 100 and p. 283 (Critical Success Factor definition). *ITIL Service Strategy*, p. 137. *Measuring ITIL*, p. 20 and p. 24. For example, in our pilot project, itil:Quickly_resolve_incidents is a instance of *itil:CSF* measured by the *itil:KPI*(s) of itil:Incident_reopen_rate, itil:Average_time_to_resolve_severity1_and_severity2_incidents_hours and itil:Incident_management_tooling_support_level. In another example, the *itil:KPI* of itil:10_percent_increase_in_customer_satisfaction_rating_for handling_incidents_over_the_next_6_months would measure an *itil:CSF* of itil:Improving_IT_service_quality, and the *itil:KPI* of itil:10_percent_reduction_in_the_costs_of_handling_printer_incidents would measure an *itil:CSF* of itil:Reducing_IT_costs.

Object Properties: *itil:hasPerformanceLevel*, *itil:measuredByKPI* and inherited from *itil:Metric*

Datatype Properties: Inherited from *itil:Metric*

Class: Dashboard

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: An *itil:Dashboard* is a graphical representation of overall IT service performance and availability. The *itil:Dashboard* images may be updated in real-time, and can also be included in management reports and Web pages. Therefore, *itil:Dashboard*(s) can be considered as key *itil:Metric*(s) that are represented on a report or graphical interface that indicates the success, at risk or failure of a business activity. They are used to quickly asses the state of operation and take timely actions to correct operational deficiencies.

The *itil:CSF*(s) are used to determine *itil:Dashboard* measures, i.e., *itil:Dashboard* results are derived from *itil:CSF* results. The *itil:Dashboard*(s) can be used to support service level management, event management or incident diagnosis.

Generalization: *itil:Metric*

Relation to ITIL: *ITIL Service Operation*, p. 283 and p. 371-372 (Dashboard definition). *Measuring ITIL*, p. 20 and p. 25-28.

Object Properties: *itil:hasCSFRelation* and inherited from *itil:Metric*

Datatype Properties: Inherited from *itil:Metric*

Class: CSFRelation

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: The *itil:Dashboard* results are derived from *itil:CSF* results. The *itil:CSF*(s) can contribute to one or more dashboards and each dashboard may have one or more multiple *itil:CSF*(s).

Generalization: *owl:Thing*

Relation to ITIL: *Measuring ITIL*, p. 20, p. 25 and p. 28.

Object Properties: *itil:factorValue* and *itil:hasScorecardType*

Datatype Properties: none

Class: Outcome

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: The *itil:Outcome*(s) are key indicators of general business risk areas, that is, they are the kind of things that IT is trying to protect against. These are associated with performance indicators that identify the success, at risk or failure of *itil:KPI*(s) or *itil:CSF*(s). The *itil:CSF*(s) are used to determine *itil:Outcome*(s) (operational risks). Legal exposure, service outages, rework, waste, security breaches, unexpected costs, slow response to business needs and changes, fines and penalties, loss of market share and dissatisfied customers are examples of *itil:Outcome*(s).

Generalization: *itil:Metric*

Relation to ITIL: *Measuring ITIL*, p. 20 and p. 29-30.

Object Properties: *itil:derivedFromCSF*, *itil:hasPerformanceLevel* and inherited from *itil:Metric*

Datatype Properties: Inherited from *itil:Metric*

Class: AnalyticalMetric

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: An *itil:AnalyticalMetric* is a used to separate out certain metrics that are really more helpful for supporting research into an issue, incident or service problem. The *itil:AnalyticalMetric*(s) are metrics that IT service providers may report on only on a one-time basis or as part of a drill-down (such as for an *itil:Dashboard*).

IT frequently makes the mistake of including these in regular reporting to senior management 'just in case'. This results in a lot of wasted labor in building reports and clouds real management issues that need to be addressed.

Generalization: *itil:Metric*

Relation to ITIL: *Measuring ITIL*, p. 33.

Object Properties: Inherited from *itil:Metric*

Datatype Properties: Inherited from *itil:Metric*

Class: Measurement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO).

Description: In general, an *itil:Metric* is a scale of *itil:Measurement* defined in terms of a standard, i.e. in terms of a well-defined unit. The quantification of an event through the process of measurement relies on the existence of an explicit or implicit metric, which is the standard to which measurements are referenced.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Continual Service Improvement*, p. 98. We use the *itil:Measurement* class to define the things that need to be measured in order to obtain an *itil:Metric*.

Object Properties: none

Datatype Properties: *itil:measureDescription* and *itil:measureName*

Class: Contract

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:Contract* is a collection of agreements. Each instance of *oc:Contract* is a legal agreement in which two or more *oc:agreeingAgents* promise to

do (or not do) something. There are legal consequences to breaking the promises made in an *oc:Contract*.

Generalization: *owl:Thing*

Relation to ITIL: In order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:Contract* as the legal agreements between *itil:Customer*(s) and *itil:ITServiceProvider*(s).

Object Properties: *oc:agreeingAgents* and *itil:agreesContractDocument*

Datatype Properties: none

Class: ContractDocument

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: An *oc:ContractDocument* is a document which outlines the contents of a legally-binding agreement.

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, in order to take advantage of existing upper ontologies, we use the OpenCyc concept *oc:ContractDocument* for the definition of legal documents that form part of a specific *oc:Contract*. In our modeling approach for an ITSMF, the *itil:Agreement* is a subclass of the OpenCyc concept *oc:ContractDocument*.

Object Properties: none

Datatype Properties: none

Class: Agreement

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: An *itil:Agreement* is a document that describes a formal understanding between two or more parties. An *itil:Agreement* is not legally binding, unless it forms part of a contract.

Generalization: *oc:ContractDocument*

Relation to ITIL: *ITIL Service Strategy*, p. 339 (Agreement definition). In our modeling approach for an ITSMF, each *itil:Agreement* defines a business process that enables the delivery of an *itil:ITService*.

Object Properties: *itil:definesBusinessProcess* and inherited from *oc:ContractDocument*

Datatype Properties: *itil:agreementCustomer*, *itil:agreementDescription*, *itil:agreementITServiceProvider*, *itil:agreementName*, *itil:agreementResponsibility*,

itil:agreementService and *itil:agreementTarget* and inherited from *oc:ContractDocument*

Class: SLA

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: An *itil:SLA* is a written agreement between an *itil:ITServiceProvider* and the *itil:Customer*(s), defining the key service targets and responsibilities of both parties. That is, an *itil:SLA* describes the *itil:ITService*, *itil:ServiceLevelTarget*(s), and specifies the responsibilities of the *itil:ITServiceProvider* and the *itil:Customer*. A single *itil:SLA* may cover multiple *itil:ITService*(s) or multiple *itil:Customer*(s).

The emphasis must be on agreement, and *itil:SLA*(s) should not be used as a way of holding one side or the other to ransom. A true partnership should be developed between the *itil:ITServiceProvider* and the *itil:Customer*, so that a mutually beneficial agreement is reached, otherwise the *itil:SLA* could quickly fall into disrepute and a 'blame culture' could develop that would prevent any true service quality improvements from taking place.

The *itil:SLA*(s) provide the basis for managing the relationship between the *itil:ITServiceProvider* and the *itil:Customer*. There are a number of potential options for *itil:SLA*(s):

(1) *Service-based SLA:* This is where an *itil:SLA* covers one service, for all the customers of that service. For example, an *itil:SLA* may be established for an organization's e-mail service covering all the customers of that service. This may appear fairly straightforward. However, difficulties may arise if the specific requirements of different customers vary for the same service, or if characteristics of the infrastructure mean that different service levels are inevitable (for example, head office staff may be connected via a high-speed LAN, while local offices may have to use a lower-speed WAN line). In such cases, separate targets may be needed within the one agreement. Difficulties may also arise in determining who should be the signatories to such an agreement. However, where common levels of service are provided across all areas of the business, for example, e-mail or telephony, the service-based SLA can be an efficient approach to use. Multiple classes of service, for example, gold, silver and bronze, can also be used to increase the effectiveness of service-based SLAs;

(2) *Customer-based SLA:* This is an agreement with an individual customer group, covering all the services they use. For example, agreements may be reached with an organization's finance department covering, say, the finance system, the accounting system, the payroll system, the billing system, the procurement system, and any other IT systems that they use. Customers often prefer such an agreement, as all of their requirements are covered in a single document. Only one signatory is normally required, which simplifies this issue. A combination of either of these structures might be appropriate, providing all services and customers are covered, with no overlap or duplication.

(3) *Multi-level SLAs:* Some organizations have chosen to adopt a multi-level SLA structure. For example, a three-layer structure as follows:

- Corporate level: covering all the generic SLM issues appropriate to every customer throughout the organization. These issues are likely to be less volatile, so updates are less frequently required.
- Customer level: covering all SLM issues relevant to the particular customer group or business unit, regardless of the service being used.
- Service level: covering all SLM issues relevant to the specific service, in relation to a specific customer group (one for each *itil:ITService* covered by the *itil:SLA*).

Generalization: *itil:Agreement*

Relation to ITIL: *ITIL Service Design*, p. 43, p. 111, p. 114-115 and p. 442 (Service Level Agreement definition). In our modeling approach for ITSMSs, the *itil:SLA* represents the document that describes a formal understanding of an agreement between *itil:Customer*(s) and the *itil:ITServiceProvider*.

Object Properties: *itil:coveringITService*, *itil:definesServiceTarget*, *itil:hasCustomerRelation*, *itil:hasITServiceProviderRelation*, *itil:hasSLAIncidentResolution*, *itil:supportedByOLA*, *itil:supportedByUC* and inherited from *itil:Agreement*

Datatype Properties: Inherited from *itil:Agreement*

Class: OLA

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: An *itil:OLA* is an agreement between an *itil:ITServiceProvider* and a third party that assists with the provision of *itil:ITService*(s) to *itil:Customer*(s). However, in this case, the third party is another part of the same *itil:Organization*. In ITIL, a third party is a person, group, or business who is not part of an *itil:SLA* for an *itil:ITService*, but is required to ensure successful delivery of that *itil:ITService* (e.g., a software supplier, a hardware maintenance company, or a facilities department). The *itil:OLA* defines the goods or services to be provided and the responsibilities of both parties. For example there could be an *itil:OLA*: (i) between the *itil:ITServiceProvider* and a facilities department that maintains the air conditioning; (ii) between the *itil:ITServiceProvider* and the network support team that supports the network service; (iii) between the *itil:ITServiceProvider* and a procurement department to obtain hardware in agreed times; and (iv) between the itil:SERVICE_DESK (*itil:RoleType* instance) and an *itil:SupportGroup* to provide *itil:Incident* resolution in agreed times. An *itil:OLA* should contain targets that underpin those within an *itil:SLA* to ensure that targets will not be breached by failure of the supporting activity. In other words, an *itil:OLA* is any underpinning agreement necessary to deliver the quality of service agreed within the *itil:SLA*.

Generalization: *itil:Agreement*

Relation to ITIL: *ITIL Service Design*, p. 43, p. 112, p. 434 (Operational Level Agreement definition) and p. 446 (Third Party definition).

Object Properties: Inherited from *itil:Agreement*

Datatype Properties: Inherited from *itil:Agreement*

Class: UC

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: An *itil:UC* is an *itil:Agreement* between an *itil:ITServiceProvider* and a third party. In this case, the third party is another *itil:Organization*. The *itil:UC* defines targets and responsibilities that are required to meet agreed *itil:ServiceLevelTarget*(s) in an *itil:SLA*.

Generalization: *itil:Agreement*

Relation to ITIL: *ITIL Service Design*, p. 43, p. 112 and p. 447 (Underpinning Contract definition).

Object Properties: Inherited from *itil:Agreement*

Datatype Properties: Inherited from *itil:Agreement*

Class: CustomerRelation

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:CustomerRelation* is used to specify the responsibilities of the *itil:Customer*(s) in a specific *itil:SLA*.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Design*, p. 109-111.

Object Properties: none

Datatype Properties: *itil:customerResponsibility*

Class: ITServiceProviderRelation

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO).

Description: The *itil:ITServiceProviderRelation* is used to specify the responsibilities of the *itil:ITServiceProvider* in a specific *itil:SLA*.

Generalization: *owl:Thing*

Relation to ITIL: *ITIL Service Design*, p. 109-111.

Object Properties: none

Datatype Properties: *itil:erviceproviderResponsibility*

Class: SLAIncidentResolution

Ontology: ITIL (itil:)

Source: Pilot Project documentation.

Description: The *itil:SLAIncidentResolution* is used to specify the agreed incident resolution times for *itil:Customer*(s) in a specific *itil:SLA*.

Generalization: *owl:Thing*

Relation to ITIL: Although this concept is not part of the ITIL documentation, we use this class to specify the agreed incident resolution times.

Object Properties: none

Datatype Properties: *itil:slaIncidentPriority* and *itil:slaIncidentResolutionTime*

Class: Identifiable

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:Identifiable* provides a mechanism to assign a unique identifier to a Workflow model element.

Generalization: *owl:Thing*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: none

Datatype Properties: *wf:elementID*

Class: NamedBpmnObject

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:NamedBpmnObject* represents Workflow model elements that may have a name and additional information.

Generalization: *owl:Thing*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: none

Datatype Properties: *wf:objectDocumentation*, *wf:objectName*, *wf:objectNcname*

Class: Artifact

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:Artifact* provides modelers with the capability of showing additional information about a business activity that is not directly related to the sequence flows or message flows of the *wf:Activity*. Three standard *wf:Artifact*(s) are provided: *wf:Association*, *wf:Group* and *wf:TextAnnotation*.

Generalization: *wf:Identifiable* and *wf:NamedBpmnObject*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:composedOfAssociations*, *wf:inArtifactsContainer* and inherited from *wf:Identifiable* and *wf:NamedBpmnObject*

Datatype Properties: Inherited from *wf:Identifiable* and *wf:NamedBpmnObject*

Class: DataObject

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:DataObject* is an *wf:Artifact* that provides provide information about what the what activities require to be performed and/or what they produce. That is, how documents, data, and other objects are used and updated during the business process. A *wf:DataObject* can represent a singular object or a collection of objects.

Generalization: *wf:Artifact*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: Inherited from *wf:Artifact*

Datatype Properties: Inherited from *wf:Artifact*

Class: Group

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:Group* is an *wf:Artifact* that provides a visual mechanism to group elements of a diagram informally.

Generalization: *wf:Artifact*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *itil:hasActivities* and inherited from *wf:Artifact*

Datatype Properties: Inherited from *wf:Artifact*

Class: TextAnnotation

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:TextAnnotation* is an *wf:Artifact* that provides a mechanism to introduce additional text information for the reader of a BPMN Diagram.

Generalization: *wf:Artifact*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: Inherited from *wf:Artifact*

Datatype Properties: Inherited from *wf:Artifact*

Class: ArtifactsContainer

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:ArtifactsContainer* provides a container for the *wf:Artifact*(s) in a BPMN diagram.

Generalization: *wf:NamedBpmnObject*

Relation to ITIL: Workflow concept associated with an itil:Activity.

Object Properties: *wf:hasArtifacts* and inherited from *wf:NamedBpmnObject*

Datatype Properties: Inherited from *wf:NamedBpmnObject*

Class: Association

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:Association* is used to associate information between *wf:Artifact*(s) and flow objects (*wf:AssociationTarget*).

Generalization: *owl:Thing*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:hasDirectionType*, *wf:source* and *wf:target*

Datatype Properties: none

Class: AssociationTarget

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:AssociationTarget* is used to obtain the targets of the *wf:Association*(s).

Generalization: *wf:Identifiable*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:hasAssociations* and inherited from *wf:Identifiable*

Datatype Properties: Inherited from *wf:Identifiable*

Class: Graph

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:Graph* is the workflow model graphical element used to define pools (*wf:Pool*) and subprocesses (*wf:SubProcess*). A *wf:Graph* is composed of vertices (*wf:Vertex*) and edges (*wf:SequenceEdge*).

Generalization: *wf:AssociationTarget* and *wf:ArtifactsContainer*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:graphComposedOf* and inherited from *wf:AssociationTarget* and *wf:ArtifactsContainer*

Datatype Properties: Inherited from *wf:AssociationTarget* and *wf:ArtifactsContainer*

Class: Vertex

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:Vertex* is a a given node in a diagram, which is a graph of diagram elements.

Generalization: *wf:AssociationTarget*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:incomingEdges*, *wf:inGraph*, *wf:outgoingEdges* and inherited from *wf:AssociationTarget*

Datatype Properties: Inherited from *wf:AssociationTarget*

Class: MessageVertex

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The *wf:MessageVertex* represents nodes that can send and/or receive messages.

Generalization: *wf:Identifiable* and *wf:NamedBpmnObject*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:incomingMessages*, *wf:outgoingMessages* and inherited from *wf:Identifiable* and *wf:NamedBpmnObject*

Datatype Properties: Inherited from *wf:Identifiable* and *wf:NamedBpmnObject*

Class: Activity

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: An *wf:Activity* is work that is performed within a business process. An *wf:Activity* can be atomic or non-atomic (compound). The *wf:Activity* represents points in a process flow where work is performed. The *wf:Activity*(s) are the executable elements of a business process.

Generalization: *wf:MessageVertex* and *wf:Vertex*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:eventHandlerFor*, *wf:hasActivityType*, *wf:inActivityGroup* and inherited from *wf:MessageVertex* and *wf:Vertex*

Datatype Properties: *wf:looping* and inherited from *wf:MessageVertex* and *wf:Vertex*

Class: SubProcess

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:SubProcess* is a *wf:Activity* that represents a behavior whose internal details have been modeled using activities, gateways, events, and sequence flows.

Generalization: *wf:Activity* and *wf:Graph*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:eventHandlers* and inherited from *wf:Activity* and *wf:Graph*

Datatype Properties: *wf:isTransaction*, *wf:adhoc* and inherited from *wf:Activity* and *wf:Graph*

Class: MessagingEdge

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:MessagingEdge* is used to connect messages nodes (*wf:MessageVertex*).

Generalization: *wf:AssociationTarget* and *wf:NamedBpmnObject*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:inBpmnDiagram*, *wf:messageVertexSource*, *wf:messageVertexTarget* and inherited from *wf:AssociationTarget* and *wf:NamedBpmnObject*

Datatype Properties: Inherited from *wf:AssociationTarget* and *wf:NamedBpmnObject*

Class: SequenceEdge

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:SequenceEdge* is used to connect nodes (*wf:Vertex*) in a graph. In *wf:SequenceEdge*, the *wf:objectName* represents the guard of the edge (i.e., the specification evaluated at runtime to determine if the edge can be traversed).

Generalization: *wf:AssociationTarget* and *wf:NamedBpmnObject*

Relation to ITIL: Workflow concept associated with an *itil:Activity*.

Object Properties: *wf:hasSequenceFlowConditionType*, *wf:inGraph*, *wf:vertexSource*, *wf:vertexTarget* and inherited from *wf:AssociationTarget* and *wf:NamedBpmnObject*

Datatype Properties: *wf:isDefault* and inherited from *wf:AssociationTarget* and *wf:NamedBpmnObject*

Class: Pool

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:Pool* is the graphical representation of a participant in a collaboration. A participant represents a specific partner entity (e.g., a company) and/or a more general partner role (e.g., a buyer, seller, or manufacturer) that are participants in a collaboration.

Generalization: *oc:Agent-Generic*, *wf:Graph* and *wf:MessageVertex*

Relation to ITIL: Workflow concept associated with an *itil:Activity*. In our modeling approach for ITSMSs, a *wf:Pool* is a subclass of the Opencyc concept *oc:Agent-Generic* representing the actor that participates in an *itil:Activity*.

Object Properties: *wf:composedOfLanes*, *wf:inBpmnDiagram* and inherited from *wf:Graph* and *wf:MessageVertex*

Datatype Properties: Inherited from *wf:Graph* and *wf:MessageVertex*

Class: Lane

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn /; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: A *wf:Lane* is a sub-partition within a *wf:Pool* which extends the entire length of the workflow level, either vertically or horizontally.

Generalization: *oc:Agent-Generic*, *wf:AssociationTarget* and *wf:NamedBpmnObject*

Relation to ITIL: Workflow concept associated with an *itil:Activity*. Just like a *wf:Pool*, in our modeling approach for ITSMSs, a *wf:Lane* is a subclass of the Opencyc concept *oc:Agent-Generic* representing the actor that participates in an *itil:Activity*.

Object Properties: *wf:hasActivities*, *wf:inPool* and inherited from *wf:AssociationTarget* and *wf:NamedBpmnObject*

Datatype Properties: Inherited from *wf:AssociationTarget* and *wf:NamedBpmnObject*

Enumerations

Class: ActivityType

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The specific value that represents the type of functionality of a specific *wf:Activity*.

Data Literals:

EventEndCancel

EventEndCompensation

EventEndEmpty

EventEndError

EventEndLink

EventEndMessage

EventEndMultiple

EventEndSignal

EventEndTerminate

EventIntermediateCancel

EventIntermediateCompensation

EventIntermediateEmpty

EventIntermediateError

EventIntermediateLink

EventIntermediateMessage

EventIntermediateMultiple

EventIntermediateRule

EventIntermediateSignal

EventIntermediateTimer

EventStartEmpty

EventStartLink

EventStartMessage

EventStartMultiple

EventStartRule

EventStartSignal

EventStartTimer

GatewayComplex

GatewayDataBasedExclusive

GatewayDataBasedInclusive

GatewayEventBasedExclusive

GatewayParallel

Subprocess

Task

Relation to ITIL: We use the *wf:ActivityType* class to model the workflow associated with an *itil:Activity*.

Class: DirectionType

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The specific value that represents the type of direction of a specific *wf:Association*.

Data Literals:

NO_DIRECTION

TO

FROM

BOTH

Relation to ITIL: We use the *wf:DirectionType* class to model the workflow associated with an *itil:Activity*.

Class: EventCategoryCode

Ontology: itil (itil)

Source: Pilot project documentation.

Description: According to our pilot project, there are four types of events depending on the business area where the event must be resolved: (i) Teaching; (ii) Systems and users; (iii) Development; and (iv) Communications.

Data Literals:

TEACHING

SYSTEMS_AND_USERS

DEVELOPMENT

COMMUNICATIONS

Relation to ITIL: We use the *itil:EventCategory* class to represent the class of a specific *itil:Event*.

Class: EventType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The specific value that represents the type of a specific *itil:Event*. Every IT service provider will have its own categorization of the significance of an *itil:Event*, but it is suggested that at least these three broad categories be represented:

- *Informational:* This refers to an event that does not require any action and does not represent an exception. They are typically stored in the system or service log files and kept for a predetermined period. Informational events are typically used to check on the status of a device or service, or to confirm the successful completion of an activity. Informational events can also be used to generate statistics (such as the number of users logged on to an application during a certain period) and as input into investigations (such as which jobs completed successfully before the transaction processing queue hung). Examples of informational events include:
  - A user logs onto an application.
  - A job in the batch queue completes successfully.
  - A device has come online.
  - A transaction is completed successfully.
- *Warning:* A warning is an event that is generated when a service or device is approaching a threshold. Warnings are intended to notify the appropriate person, process or tool so that the situation can be checked and the appropriate action taken to prevent an exception. Warnings are not typically raised for a device failure. Although there is some debate about whether the failure of a redundant device is a warning or an exception (since the service is still available). A good rule is that every failure should be treated as an exception, since the risk of an incident impacting the business is much greater. Examples of warnings are:
  - Memory utilization on a server is currently at 65% and increasing. If it reaches 75%, response times will be unacceptably long and the OLA for that department will be breached.
  - The collision rate on a network has increased by 15% over the past hour.
- *Exception:* An exception means that a service or device is currently operating abnormally (however that has been defined). Typically, this means that an *itil:OLA* and *itil:SLA* have been breached and the business is being impacted. Exceptions could represent a total failure, impaired functionality or degraded performance. Please note, though, that an exception does not always represent an

incident. For example, an exception could be generated when an unauthorized device is discovered on the network. This can be managed by using either an Incident Record or a Request for Change (or even both), depending on the organization's Incident and Change Management policies. Examples of exceptions include:

- A server is down.
- Response time of a standard transaction across the network has slowed to more than 15 seconds.
- More than 150 users have logged on to the General Ledger application concurrently.
- A segment of the network is not responding to routine requests.

Data Literals:

INFORMATIONAL

WARNING

EXCEPTION

Relation to ITIL: *ITIL Service Operation*, p. 71-73.

Class: IncidentGroupType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The user group that can report an *itil:Incident*.

Data Literals:

GOVERNANCE

ICTD

STAFF

STUDENT

OTHER

Relation to ITIL: *ITIL Service Operation*, p. 91.

Class: IncidentStatusType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The information needed for each incident is likely to include the incident status (active, waiting, closed, etc.).

Data Literals:

NEW

ACCEPTED

ACTIVE

WAITING

PLANNED

RESOLVED

CLOSED

Relation to ITIL: *ITIL Service Operation*, p. 91.

Class: InterfaceRelationType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: An *itil:Process* may have input and output interfaces with other *itil:Process*(s).

Data Literals:

INPUT

OUTPUT

Relation to ITIL: *ITIL Service Operation*, p. 100-101.

Class: ManagedEventType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The specific value that represents the type of monitoring and control systems used for a specific *itil:Event*. An effective service operation is dependent on knowing the status of the infrastructure and detecting any deviation from normal or expected operation. This is provided by good monitoring and control systems, which are based on two types of tools (note that reactive and proactive monitoring could be active or passive):

(1) *Active versus Passive Monitoring*: active monitoring tools poll key CIs to determine their status and availability. Any exceptions will generate an alert that needs to be communicated to the appropriate tool or team for action. On the other hand, passive monitoring tools detect and correlate operational alerts or communications generated by CIs.

(2) *Reactive versus Proactive*: reactive monitoring is designed to request or trigger action following a certain type of event or failure. For example, server performance degradation may trigger a reboot, or a system failure will generate an incident. Reactive monitoring is not only used for exceptions. It can also be

used as part of normal operations procedures, for example a batch job completes successfully, which prompts the scheduling system to submit the next batch job. On the other hand, proactive monitoring is used to detect patterns of events which indicate that a system or service may be about to fail. Proactive monitoring is generally used in more mature environments where these patterns have been detected previously, often several times. Proactive monitoring tools are therefore a means of automating the experience of seasoned IT staff and are often created through the proactive problem management process. Generally, it is better to manage IT services proactively, but achieving this is not easily planned or achieved.

Data Literals:

PROACTIVE_ACTIVE

PROACTIVE_PASSIVE

REACTIVE_ACTIVE

REACTIVE_PASSIVE

Relation to ITIL: *ITIL Service Operation*, p. 159-160.

Class: MetricType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO); OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO).

Description: The specific value that represents the type of a specific *itil:Metric*. There are three types of metrics that an organization will need to collect to support CSI activities as well as other process activities. The types of metrics are:

(1) *Technology metrics:* these metrics are often associated with component and application based metrics such as performance, availability etc.

(2) *Process metrics:* these metrics are captured in the form of CSFs, KPIs and activity metrics for the service management processes. These metrics can help determine the overall health of a process. Four key questions that KPIs can help answer are around quality, performance, value and compliance of following the process. CSI would use these metrics as input in identifying improvement opportunities for each process.

(3) *Service metrics:* these metrics are the results of the end-to-end service. Component/technology metrics are used to compute the service metrics.

Also, there are four types of metrics that can be used to measure the capability and performance of processes:

(1) *Progress:* milestones and deliverables in the capability of the process.

(2) *Compliance:* compliance of the process to governance requirements, regulatory requirements and compliance of people to the use of the process.

(3) *Effectiveness:* the accuracy and correctness of the process and its ability to deliver the 'right result.'

(4) *Efficiency:* the productivity of the process, its speed, throughput and resource utilization.

Data Literals:

PROCESS

SERVICE

TECHNOLOGY

PROGRESS

COMPLIANCE

EFFECTIVENESS

EFFICIENCY

Relation to ITIL: *ITIL Service Design*, p. 77. *ITIL Continual Service Improvement*, p. 72.

Class: PerformanceLevel

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: The specific value that represents the level of performance associated with *itil:CSF*(s) and *itil:Outcome*(s).

Data Literals:

HIGH

MEDIUM

LOW

Relation to ITIL: *Measuring ITIL*, p. 24 and p. 29-30.

Class: RACICode

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO); OGC. (2007). *ITIL Service Transition*. The Stationery Office (TSO); OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO).

Description: The specific value that represents the RACI code of a specific *oc:IntelligentAgent*. RACI is a model used to help define roles and responsibilities in ITIL V3. The RACI model will be beneficial in enabling decisions to be made with

pace and confidence. RACI stands for Responsible, Accountable, Consulted and Informed:

(1) *Responsible*: the individual who is responsible to perform the actions.

(2) *Accountable*: the individual who is ultimately accountable has the power of veto. Only one accountable can be assigned to an action.

(3) *Consulted*: the individual(s) to be consulted prior to a final decision or action being taken.

(4) *Informed*: the individual(s) who needs to be informed after a decision or action is taken.

To build a RACI chart the following steps are required:

(1) Identify the activities/processes.

(2) Identify/define the functional roles.

(3) Conduct meetings and assign the RACI codes.

(4) Identify any gaps or overlaps – for example, where there are two Rs or no Rs (see analysis below).

(5) Distribute the chart and incorporate feedback.

(6) Ensure that the allocations are being followed.

Developing an authority matrix (RACI matrix) can be a tedious and time-consuming exercise but it's a crucially important one. The authority matrix clarifies to all involved which activities they are expected to fulfill, as well as identifying any gaps in service delivery and responsibilities. It is especially helpful in clarifying the staffing model necessary for improvement.

Data Literals:

R

A

C

I

Relation to ITIL: *ITIL Service Design*, p. 323-324 and p. 437 (RACI definition). *ITIL Service Transition*, p. 136-138 and p. 288-290. *ITIL Continual Service Improvement*, p. 215-218.

Class: RoleType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO).

Description: The specific value that represents the type of role of a specific *oc:IntelligentAgent*. Roles and responsibilities are defined within organizations for people. The key to effective ITSM is ensuring that there is clear accountability and roles defined to carry out the practice of Service Operation. A role is a set of responsibilities,

activities and authorities granted to a person or team. A role is defined in a process. One person or team may have multiple Roles, for example the roles of configuration manager and change manager may be carried out by a single person. The size of an organization, how it is structured, the existence of external partners and other factors will influence how roles are assigned. Whether a particular role is filled by a single individual or shared between two or more, the importance is the consistency of accountability and execution, along with the interaction with other roles in the organization.

Data Literals:

ACCESS_MANAGER

BPO

CARS

CEO

CHA

CIO

FIRST_LINE_SUPPORT

HA

HD

HO

HELP_DESK

INCIDENT_MANAGER

IT_FACILITIES_MANAGER

IT_OPERATIONS_MANAGER

IT_OPERATOR

MAJOR_INCIDENT_TEAM

PROBLEM_MANAGER

PRODUCT_MANAGER

SECOND_LINE_SUPPORT

SERVICE_DESK

SERVICE_REQUEST_FULFILLMENT_GROUP

THIRD_LINE_SUPPORT

Relation to ITIL: *ITIL Service Operation*, 6.6. Service Operation roles and responsibilities, p. 256-267 and p. 387 (Role definition).

Class: ScorecardType

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: The specific value that represents the type of scorecard of a specific *itil:Dashboard*. The *itil:Dashboard*(s) come in all forms, shapes and sizes. For the purpose of our modeling approach for ITSMSs, just like the approach of [Steinberg, 2006], we use the *Balanced Scorecard* originally developed in [Kaplan & Norton, 1992]. The Balanced Scorecard was originally developed around the concept that financial measures alone are not critical for business success. The Balanced Scorecard has been generally recognized as an acceptable approach for senior management levels. The scorecard categories recommended for ITSM are:

- *Customer:* The Customer category represents the customer view of the services being delivered. Are they satisfied? Are they serviced in accordance with agreements and expectations? 'Protect services when making changes' and 'Make changes quickly and accurately in line with business needs' are examples of some Change Management CSFs that contribute to Customer. Both of these CSFs impact how a customer might be receiving (or not receiving) their services.
- *Capabilities:* The Capabilities category represents, in the ITSM sense, the capability of the IT service provider to meet business needs. Is there enough capacity to handle planned business volumes? Is there enough capacity to handle anticipated business and IT changes? Does the IT staff possess the right skills? 'Provide services with appropriate capacity to match business need' and 'Provide accurate capacity forecasts' are examples of some Capacity Management CSFs that contribute to Capabilities. These CSFs represent whether the IT service provider is capable of delivering needed capacity to support services by accurately predicting capacity needs and providing needed capacity at the right time to match business requirements.
- *Operational:* The Operational category represents, in the ITSM sense, how well the IT service provider is delivering their services on a day-to-day basis. Are services levels being met? Are incidents resolved on a timely basis? 'Quickly resolve incidents' and 'Maintain IT service quality' are examples of some Incident Management CSFs that contribute to Operational. These CSFs relate to everyday tasks (in this case Incident Management tasks) and whether those tasks are operating in a repeatable, consistent, efficient and effective manner to quickly resolve incidents and take actions to maintain the quality of the services being delivered.
- *Financial:* The Financial category represents, in the ITSM sense, how well the IT service provider is managing and controlling costs as well as protecting and enhancing revenue. Are IT costs effectively managed? Are costs staying within budget? Does revenue received for IT chargeback cover the costs for the services being charged for? 'Provide effective stewardship of IT Finances', 'Maintain overall effectiveness of the IT Financial Management Process' and 'Recapture IT costs through chargeback for delivery of IT services' are examples of some Financial Management CSFs that contribute to Financial.

- *Regulatory:* The Regulatory category represents, in the ITSM sense, how well the IT service provider is operating in a manner that protects it again regulatory risks for fines, penalties and audit issues. While not part of the original Balanced Scorecard approach, it has been included because of the recent emphasis on IT regulatory issues. Is effective stewardship maintained over IT costs? Is the infrastructure protected from unauthorized changes? Is the infrastructure adequately protected from security risks? 'Provide effective stewardship of IT finances', 'Use a repeatable process for handling changes', 'Provide a repeatable process for rolling out releases' and 'Maintain viability of IT Service Continuity Plans' are examples of some CSFs that contribute to Regulatory.

Data Literals:

CUSTOMER

CAPABILITIES

OPERATIONAL

FINANCIAL

REGULATORY

Relation to ITIL: *Measuring ITIL*, p. 25-28.

Class: SequenceFlowConditionType

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/; Object Management Group (OMG), 2010. Business Process Model and Notation (BPMN) Version 2.0. Available at: http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

Description: The specific value that represents the type of condition of a specific *wf:SequenceEdge*.

Data Literals:

NONE

EXPRESSION

DEFAULT

Relation to ITIL: We use the *wf:SequenceFlowConditionType* class to model the workflow associated with an *itil:Activity*.

Class: ServicePortfolioType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO); OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO).

Description: The specific value that represents the type of a specific *itil:ServicePortfolio*. The *itil:ServicePortfolio* is used to manage the entire lifecycle of

all *itil:ITService*(s), and it includes three categories: Service pipeline (proposed or in development), service catalog (live or available for deployment) and retired services.

- *Service Pipeline:* The Service Pipeline is a database or structured document listing all *itil:ITService*(s) that are under consideration or development, but are not yet available to customers. The Service Pipeline provides a business view of possible future *itil:ITService*(s) and is part of the *itil:ServicePortfolio* which is not normally published to *itil:Customer*(s). These services are to be phased into operation by *itil:ServiceTransition* after completion of design, development, and testing. The pipeline represents the IT service provider's growth and strategic outlook for the future. The general health of the *itil:ITServiceProvider* is reflected in the pipeline. It also reflects the extent to which new service concepts and ideas for improvement are being fed by *itil:ServiceStrategy*, *itil:ServiceDesign* and *itil:CSI*. Good *itil:FinancialManagement* is necessary to ensure adequate funding for the pipeline.

- *Service Catalog:* The Service Catalog is a database or structured document with information about all live *itil:ITService*(s), including those available for deployment. The Service Catalog is basic aspect of all *itil:ITServiceProvider*, and it is the only part of the *itil:ServicePortfolio* published to customers, and is used to support the sale and delivery of *itil:ITService*(s). As mentioned earlier, the *itil:ServicePortfolio* is the complete set of *itil:ITService*(s) that are managed by an *itil:ITServiceProvider*. The Service Catalog includes information about deliverables, prices, contact points, ordering and request processes.

  The Service Catalog is a key element containing valuable information on the complete set of services offered. It should preferably be stored as a set of 'service' CIs within a *Configuration Management System* (CMS), maintained under the *itil:ChangeManagement* process. As it is such a valuable set of information it should be available to anyone within the *itil:Organization*. Every new *itil:ITService* should immediately be entered into the Service Catalog once its initial definition of requirements has been documented and agreed. The Service Catalog should record the status of every *itil:ITService*, through the *itil:ServiceStage*(s) of its defined *itil:ServiceLifecycle*.

  The Service Catalog will also show the relationship between *itil:ITService*(s) and *itil:Application*(s). A single *itil:Application* could be part of more than one *itil:ITService*, and a single *itil:ITService* could use more than one *itil:Application*.

  A Service Catalog is also a collection of LOS, each under the control of a product manager.

- *Retired Services:* Some services in the *itil:ServicePortfolio* are phased out or retired. Phasing out of services is part of *itil:ServiceTransition*. This is to ensure that all commitments made to customers are duly fulfilled and service assets are released from contracts. When services are retired, the related knowledge and information are stored in a knowledge base for future use: Retired Services. Retired Services are not available to new customers or contracts unless a special business case is made. Such services may be reactivated into operations under special conditions and SLAs that are to be approved by senior management. This

is necessary because such services may cost a lot more to support and may disrupt economies of scale and scope.

Data Literals:

SERVICE_PIPELINE

SERVICE_CATALOG

RETIRED_SERVICES

Relation to ITIL: *ITIL Service Design*, p. 84, p. 390, p. 441 (Service Catalog definition) and p. 446 (Third Party definition). *ITIL Service Strategy*, p. 116-117, p. 120, p. 213-215 and p. 367 (Service Portfolio definition).

Class: TechnicalManagementType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO); Pilot project documentation.

Description: The specific value that represents the type of intervention in a specific *itil:Event*. Technical management is not normally provided by a single department or group. One or more technical support teams or departments will be needed to provide technical management and support for the IT Infrastructure. In all but the smallest organizations, where a single combined team or department may suffice, separate teams or departments will be needed for each type of infrastructure being used.

Data Literals:

PHYSICAL

NON_PHYSICAL

AUTOMATED

Relation to ITIL: *ITIL Service Operation*, p. 222-223. According to our pilot project, there are three types of technical support depending on the type of the intervention: physical (i.e., it is managed by an agent), non physical or automated.

Object Properties

Property: affectsCI

Ontology: ITIL (itil:)

Source: see the class *itil:ChangeRecord*.

Description: (itil:affectsCI itil:ChangeRecord itil:CI) means that the *itil:CI* is affected by the change detailed in *itil:ChangeRecord*.

Functional: No

Inverse: none

Domain: *itil:ChangeRecord*

Range: *itil:CI*

Subproperties: none

Property: agreeingAgents

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:agreeingAgents oc:Contract oc:IntelligentAgent) means that the *oc:Contract* has the *oc:IntelligentAgent*(s) among its agreeing parties. This property relates an agreement to the agents who made or are making the agreement.

Functional: No

Inverse: none

Domain: *oc:Contract*

Range: *oc:IntelligentAgent*

Subproperties: none

Property: agreesContractDocument

Ontology: ITIL (itil:)

Source: see the class *oc:Contract*.

Description: (itil:agreesContractDocument oc:Contract oc:ContractDocument) means that the *oc:ContractDocument* outline the contents of the *oc:Contract*.

Functional: No

Inverse: none

Domain: *oc:Contract*

Range: *oc:ContractDocument*

Subproperties: none

Property: basedOnKPI

Ontology: ITIL (itil:)

Source: see the class *itil:ServiceLevelTarget*.

Description: (itil:basedOnKPI itil:ServiceLevelTarget itil:KPI) means that the *itil:ServiceLevelTarget* is based on the *itil:KPI*.

Functional: No

Inverse: none

Domain: *itil:ServiceLevelTarget*

Range: itil:KPI

Subproperties: none

Property: basedOnSLR

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Service Level Requirement definition).

Description: (itil:basedOnSLR itil:ServiceLevelTarget itil:SLR) means that the *itil:ServiceLevelTarget* is based on the *itil:SLR*.

Functional: No

Inverse: *itil:usedForNegotiation*

Domain: *itil:ServiceLevelTarget*

Range: *itil:SLR*

Subproperties: none

Property: composedOfAssociations

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:composedOfAssociations wf:Artifact wf:Association) means that the *wf:Artifact* is composed of the *wf:Association*.

Functional: No

Inverse: *wf:source*

Domain: *wf:Artifact*

Range: *wf:Association*

Subproperties: none

Property: composedOfLanes

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:composedOfLanes wf:Pool wf:Lane) means that the *wf:Pool* is composed of *wf:Lane*.

Functional: No

Inverse: *wf:inPool*

Domain: *wf:Pool*

Range: *wf:Lane*

Subproperties: none

Property: coordinatedBySpecification

Ontology: ITIL (itil:)

Source: see the class *itil:Activity*.

Description: (itil:coordinatedBySpecification itil:Activity oc:Specification) means that the *itil:Activity* is defined according to the *oc:Specification*.

Functional: No

Inverse: *itil:specifiesActivity*

Domain: *itil:Activity*

Range: *oc:Specification*

Subproperties: none

Property: coveringITService

Ontology: ITIL (itil:)

Source: see the class: *itil:SLA*.

Description: (itil:coveringITService itil:SLA itil:ITService) means that the *itil:SLA* is defined for the *itil:ITService*.

Functional: No

Inverse: none

Domain: *itil:SLA*

Range: *itil:ITService*

Subproperties: none

Property: definesBusinessProcess

Ontology: ITIL (itil:)

Source: see the class *itil:Agreement*.

Description: (itil:definesBusinessProcess itil:Agreement itil:Activity) means that the *itil:Agreement* includes the *itil:Activity* in its content.

Functional: Yes

Inverse: none

Domain: *itil:Agreement*

Range: *itil:Activity*

Subproperties: none

Property: definesMetric

Ontology: ITIL (itil:)

Source: see the class *itil:ITService*.

Description: (itil:definesMetric itil:ITService itil:Metric) means that the *itil:ITService* measures the service processes using the *itil:Metric*.

Functional: No

Inverse: none

Domain: *itil:ITService*

Range: *itil:Metric*

Subproperties: none

Property: definesServiceTarget

Ontology: ITIL (itil:)

Source: see the class *itil:SLA*.

Description: (itil:definesServiceTarget itil:SLA itil:ServiceLevelTarget) means that the *itil:SLA* defines the *itil:ServiceLevelTarget*.

Functional: No

Inverse: none

Domain: *itil:SLA*

Range: *itil:ServiceLevelTarget*

Subproperties: none

Property: derivedFromCSF

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 20 and p. 29-30.

Description: (itil:derivedFromCSF itil:Outcome itil:CSF) means that the *itil:Outcome* derives from the *itil:CSF*.

Functional: No

Inverse: none

Domain: *itil:Outcome*

Range: *itil:CSF*

Subproperties: none

Property: detailsITService

Ontology: ITIL (itil:)

Source: see the class *itil:ServicePortfolio*.

Description: (itil:detailsITService itil:ServicePortfolio itil:ITService) means that the *itil:ITService*(s) are contained within the *itil:ServicePortfolio*.

Functional: No

Inverse: *itil:inServicePortfolio*

Domain: *itil:ServicePortfolio*

Range: *itil:ITService*

Subproperties: none

Property: diagramComposedOf

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:diagramComposedOf wf:BpmnDiagram wf:MessagingEdge/wf:Pool) means that the *wf:BpmnDiagram* is composed of the *wf:MessagingEdge*/*wf:Pool*.

Functional: No

Inverse: *wf:inBpmnDiagram*

Domain: *wf:BpmnDiagram*

Range:

wf:MessagingEdge

wf:Pool

Subproperties: none

Property: eventHandlerFor

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:eventHandlerFor wf:Activity wf:SubProcess) means that the *wf:Activity* is the event handler for the *wf:SubProcess*.

Functional: Yes

Inverse: *wf:eventHandlers*

Domain: *wf:Activity*

Range: *wf:SubProcess*

Subproperties: none

Property: eventHandlers

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:eventHandlers wf:SubProcess wf:Activity) means that the *wf:Activity* is an event handler for the *wf:SubProcess*.

Functional: No

Inverse: *wf:eventHandlerfor*

Domain: *wf:SubProcess*

Range: *wf:Activity*

Subproperties: none

Property: factorValue

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 20 and p. 25.

Description: (itil:factorValue itil:CSFRelation itil:CSF) means that the *itil:CSFRelation* represents the *itil:CSF*.

Functional: Yes

Inverse: none

Domain: *itil:CSFRelation*

Range: *itil:CSF*

Subproperties: none

Property: fallsIntoToleranceRange

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 20 and p. 23.

Description: (itil:fallsIntoToleranceRange itil:KPI itil:Tolerance) means that the *itil:KPI* results will fall into a *itil:Tolerance* range. Each *itil:KPI* should be associated with one or more *itil:Tolerance* values. For example, an upper value can represent a desired service target for the *itil:KPI* and a lower value can represent a warning level or point at which some further action should occur.

Functional: Yes

Inverse: none

Domain: *itil:KPI*

Range: *itil:Tolerance*

Subproperties: none

Property: graphComposedOf

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:graphComposedOf wf:Graph wf:SequenceEdge/wf:Vertex) means that the *wf:Graph* is composed of the *wf:SequenceEdge*/*wf:Vertex*.

Functional: No

Inverse: *wf:inGraph*

Domain: *wf:Graph*

Range:

wf:SequenceEdge

wf:Vertex

Subproperties: none

Property: hasActivities

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:hasActivities wf:Group/wf:Lane wf:Activity) means that the *wf:Group*/*wf:Lane* includes the *wf:Activity*.

Functional: No

Inverse: *wf:inActivityGroup*

Domain:

wf: Group

wf:Lane

Range: *wf:Activity*

Subproperties: none

Property: hasActivityType

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:hasActivityType wf:Activity wf:ActivityType) means that the *wf:Activity* has the type *wf:ActivityType*.

Functional: Yes

Inverse: none

Domain: *wf:Activity*

Range: *wf:ActivityType*

Subproperties: none

Property: hasAnalyticalMetric

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 33.

Description: (itil:hasAnalyticalMetric itil:Metric itil:AnalyticalMetric) means that the *itil:AnalyticalMetric* is a subset of subdivision of an *itil:Metric*. For example, the *itil:OperationalMetric* of *Total number of incidents* for analytical purposes could be broken out by the next *itil:AnalyticalMetric*(s):

- Geographic region,
- Department of business unit,
- Technology platform,
- IT service delivered,
- Time of day,
- etc.

Functional: No

Inverse: none

Domain: *itil:Metric*

Range: *itil:AnalyticalMetric*

Subproperties: none

Property: hasApplication

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO), p. 340.

Description: (itil:hasApplication itil:ITService itil:Application) means that the *itil:ITService* uses the *itil:Application*.

Functional: No

Inverse: *itil:supportsITService*

Domain: *itil:ITService*

Range: *itil:Application*

Subproperties: none

Property: hasArtifacts

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:hasArtifacts wf:ArtifactsContainer wf:Artifact) means that the *wf:ArtifactsContainer* is composed of the *wf:Artifact*.

Functional: No

Inverse: *wf:inArtifactsContainer*

Domain: *wf:ArtifactsContainer*

Range: *wf:Artifact*

Subproperties: none

Property: hasAssociations

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:hasAssociations wf:AssociationTarget wf:Association) means that the *wf:AssociationTarget* is associated with the *wf:Association*.

Functional: No

Inverse: *wf:target*

Domain: *wf:AssociationTarget*

Range: *wf:Association*

Subproperties: none

Property: hasChangeRecord

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Change Record definition).

Description: (itil:hasChangeRecord itil:RFC itil:ChangeRecord) means that the *itil:ChangeRecord* contains the details of the change proposed in the *itil:RFC*.

Functional: Yes

Inverse: none

Domain: *itil:RFC*

Range: *itil:ChangeRecord*

Subproperties: none

Property: hasConfigurationRecord

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Configuration Record definition).

Description: (itil:hasConfigurationRecord itil:CI itil:ConfigurationRecord) means that the *itil:ConfigurationRecord* contains the details of the *itil:CI*.

Functional: No

Inverse: none

Domain: *itil:CI*

Range: *itil:ConfigurationRecord*

Subproperties: none

Property: hasCSFRelation

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 20 and p. 25.

Description: (itil:hasCSFRelation itil:Dashboard itil:CSFRelation) means that the *itil:Dashboard* has the *itil:CSFRelation*.

Functional: No

Inverse: none

Domain: *itil:Dashboard*

Range: *itil:CSFRelation*

Subproperties: none

Property: hasCustomerRelation

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO), p. 109-111.

Description: (itil:hasCustomerRelation itil:SLA itil:CustomerRelation) means that the *itil:SLA* has the *itil:CustomerRelation*, used to specify the responsibilities of the *itil:Customer*(s) in a specific *itil:SLA*.

Functional: No

Inverse: none

Domain: *itil:SLA*

Range: *itil:CustomerRelation*

Subproperties: none

Property: hasCustomerReq

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Service Level Requirement definition).

Description: (itil:hasCustomerReq itil:ITService itil:SLR) means that the *itil:SLR* is a customer requirement for an aspect of an *itil:ITService*.

Functional: No

Inverse: none

Domain: *itil:ITService*

Range: *itil:SLR*

Subproperties: none

Property: hasDirectionType

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf: hasDirectionType wf:Association wf:DirectionType) means that the directions of the *wf:Association* has the type *wf:DirectionType*.

Functional: Yes

Inverse: none

Domain: *wf:Association*

Range: *wf:DirectionType*

Subproperties: none

Property: hasEventCategoryCode

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: (itil:hasEventCategoryCode itil:Event itil:EventCategoryCode) means that the *itil:Event* has the class of *itil:EventCategoryCode*.

Functional: Yes

Inverse: none

Domain: *itil:Event*

Range: *itil:EventCategoryCode*

Subproperties: none

Property: hasEventLifecycle

Ontology: ITIL (itil:)

Source: see the class *itil:Event*.

Description: (itil:hasEventLifecycle itil:Event itil:Lifecycle) means that the *itil:Lifecycle* represents the lifecycle of the *itil:Event*.

Functional: Yes

Inverse: none

Domain: *itil:Event*

Range: *itil:Lifecycle*

Subproperties: none

Property: hasEventType

Ontology: ITIL (itil:)

Source: see the class *itil:EventType*.

Description: (itil:hasEventType itil:Event itil:EventType) means that the *itil:Event* has the type *itil:EventType*.

Functional: Yes

Inverse: none

Domain: *itil:Event*

Range: *itil:EventType*

Subproperties: none

Property: hasIncidentGroup

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: (itil:hasIncidentGroup itil:IncidentRecord itil:IncidentGroupType) means that the incident detailed in *itil:IncidentRecord* has been reported by a member of the *itil:IncidentGroupType*.

Functional: Yes

Inverse: none

Domain: *itil:IncidentRecord*

Range: *itil:IncidentGroupType*

Subproperties: none

Property: hasIncidentRecord

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Incident Record definition).

Description: (itil:hasIncidentRecord itil:Incident itil:IncidentRecord) means that the *itil:IncidentRecord* contains the details of the *itil:Incident*.

Functional: No

Inverse: none

Domain: *itil:Incident*

Range: *itil:IncidentRecord*

Subproperties: none

Property: hasIncidentResponsible

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: (itil:hasIncidentResponsible itil:IncidentRecord oc:Agent-Generic) means that the *oc:Agent-Generic* is the responsible of the incident detailed in *itil:IncidentRecord*.

Functional: Yes

Inverse: none

Domain: *itil:IncidentRecord*

Range: *oc:Agent-Generic*

Subproperties: none

Property: hasIncidentStatus

Ontology: ITIL (itil:)

Source: see the class *itil:IncidentStatusType*.

Description: (itil:hasIncidentStatus itil:IncidentRecord itil:IncidentStatusType) means that the incident detailed in *itil:IncidentRecord* has the status defined in *itil:IncidentStatusType*.

Functional: Yes

Inverse: none

Domain: *itil:IncidentRecord*

Range: *itil:IncidentStatusType*

Subproperties: none

Property: hasInterfaceRelation

Ontology: ITIL (itil:)

Source: see the class *itil:Process*.

Description: (itil:hasInterfaceRelation itil:Process itil:InterfaceRelation) means that the *itil:Process* has the *itil:InterfaceRelation*.

Functional: No

Inverse: none

Domain: *itil:Process*

Range: *itil:InterfaceRelation*

Subproperties: none

Property: hasInterfaceRelationType

Ontology: ITIL (itil:)

Source: see the class *itil:InterfaceRelationType*.

Description: (itil:hasInterfaceRelationType itil:InterfaceRelation itil:InterfaceRelationType) means that the *itil:InterfaceRelation* has the type *itil:InterfaceRelationType*.

Functional: Yes

Inverse: none

Domain: *itil:InterfaceRelation*

Range: *itil:InterfaceRelationType*

Subproperties: none

Property: hasITService

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Service Package definition)

Description: (itil:hasITService itil:ServicePackage itil:ITService) means that the *itil:ServicePackage* includes the *itil:ITService*.

Functional: No

Inverse: none

Domain: *itil:ServicePackage*

Range: *itil:ITService*

Subproperties: none

Property: hasITServiceProviderRelation

Ontology: ITIL (itil:)

Source: see the class *itil:SLA*.

Description: (itil:hasITServiceProviderRelation itil:SLA itil:ITServiceProviderRelation) means that the *itil:SLA* has the *itil:ITServiceProviderRelation*, used to specify the responsibilities of the *itil:ITServiceProvider* in a specific *itil:SLA*.

Functional: Yes

Inverse: none

Domain: *itil:SLA*

Range: *itil:ITServiceProviderRelation*

Subproperties: none

Property: hasManagedEventType

Ontology: ITIL (itil:)

Source: see the class *itil:ManagedEventType*.

Description: (itil:hasManagedEventType itil:Event itil:ManagedEventType) means that the *itil:Event* has the type *itil:ManagedEventType*.

Functional: Yes

Inverse: none

Domain: *itil:Event*

Range: *itil:ManagedEventType*

Subproperties: none

Property: hasMembers

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:hasMembers oc:Organization oc:Agent-Generic) means that the *oc:Agent-Generic* is a member of the *oc:Organization*; typically, membership eligibility is determined by the *oc:Organizaiton* and accepted with *oc:Agent-Generic*'s voluntary affiliation. The predicate *oc:hasMembers* relates a particular organization to the agents who are members of that organization. This predicate indicates 'generic' membership, although there may be specialized kinds of membership in the same organization.

Functional: No

Inverse: none

Domain: *oc:Organization*

Range: *oc:Agent-Generic*

Subproperties: none

Property: hasMetricType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO), p. 77; OGC. (2007). *ITIL Continual Service Improvement*. The Stationery Office (TSO), p. 72.

Description: (itil:hasMetricType itil:Metric itil:MetricType) means that the *itil:Metric* has the type *itil:MetricType*.

Functional: Yes

Inverse: none

Domain: *itil:Metric*

Range: *itil:MetricType*

Subproperties: none

Property: hasPerformanceLevel

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 24 and p. 29-30.

Description: (itil:hasPerformanceLevel itil:CSF itil:PerformanceLevel) means that the *itil:CSF* or the *itil:Outcome* has the level *itil:PerformanceLevel*. In an *itil:CSF*, to receive the performance level of 'High', all the associated *itil:KPI*(s) must have met or exceeded their *itil:Tolerance* acceptable values. When one of the associated *itil:KPI*(s) falls into an *itil:Tolerance* non-acceptable value, the *itil:CSF* performance level might

be ‘Medium’ or ‘Low’ depending on how the associated *itil:KPI* value fell within the specified *itil:Tolerance* range for it.

On the other hand, *itil:Outcome*(s) can be associated with a performance indicator (High, Medium or Low) that might reflect the likelihood of risk that the *itil:Outcome* will occur. In our modeling approach for ITSMSs, the risk level is derived from the mean average of the *itil:CSF* performance levels. Scoring for an *itil:Outcome* runs opposite to how the *itil:CSF*(s) are calculated. If an *itil:CSF* scores ‘Low’, meaning the likelihood of achieving that *itil:CSF* is low, then the *itil:Outcome* would score ‘High’. This means that the risk of the *itil:Outcome* occurring is high because the *itil:CSF* achievement was low.

Functional: Yes

Inverse: none

Domain:

itil:CSF

itil:Outcome

Range: *itil:PerformanceLevel*

Subproperties: none

Property: hasProblemRecord

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Problem Record definition).

Description: (itil:hasProblemRecord itil:Problem itil:ProblemRecord) means that the *itil:ProblemRecord* contains the details of the *itil:Problem*.

Functional: Yes

Inverse: none

Domain: *itil:Problem*

Range: *itil:ProblemRecord*

Subproperties: none

Property: hasProcess

Ontology: ITIL (itil:)

Source: see the class *itil:ServiceStage*.

Description: (itil:hasProcess itil:ServiceStage itil:Process) means that the *itil:ServiceStage* includes the *itil:Process*.

Functional: No

Inverse: *itil:inServiceStage*

Domain: *itil:ServiceStage*

Range: *itil:Process*

Subproperties:

(itil:hasStrategyProcess itil:ServiceStrategy itil:StrategyProcess) ->

inverse: (itil:inStrategyStage itil:StrategyProcess itil:ServiceStrategy)

(itil:hasDesignProcess itil:ServiceDesign itil:DesignProcess) ->

inverse: (itil:inDesignStage itil:DesignProcess itil:ServiceDesign)

(itil:hasTransitionProcess itil:ServiceTransition itil:TransitionProcess) ->

inverse: (itil:inTransitionStage itil:TransitionProcess itil:ServiceTransition)

(itil:hasOperationProcess itil:ServiceOperation itil:OperationProcess) ->

inverse: (itil:inOperationStage itil:OperationProcess itil:ServiceOperation)

(itil:hasCSIProcess itil:ContinualServiceImprovement itil:CSIProcess) ->

inverse: (itil:inCSIStage itil:CSIProcess itil:ContinualServiceImprovement)

Property: hasRoleRelation

Ontology: ITIL (itil:)

Source: see the class *oc:IntelligentAgent*.

Description: (itil:hasRoleRelation oc:IntelligentAgent itil:RoleRelation) means that the *oc:IntelligentAgent* is assigned with the *itil:RoleRelation*.

Functional: No

Inverse: none

Domain: *oc:IntelligentAgent*

Range: *itil:RoleRelation*

Subproperties: none

Property: hasScorecardType

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 25-28.

Description: (itil:hasScorecardType itil:CSFRelation itil:ScorecardType) means that the *itil:CSFRelation* has the type *itil:ScorecardType*.

Functional: Yes

Inverse: none

Domain: *itil:CSFRelation*

Range: *itil:ScorecardType*

Subproperties: none

Property: hasSequenceFlowConditionType

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:hasSequenceFlowConditionType wf:SequenceEdge wf:SequenceFlowConditionType) means that the *wf:SequenceEdge* has the type *wf:SequenceFlowConditionType*.

Functional: Yes

Inverse: none

Domain: *wf:SequenceEdge*

Range: *wf:SequenceFlowConditionType*

Subproperties: none

Property: hasServiceLifecycle

Ontology: ITIL (itil:)

Source: see the class *itil:ITService*.

Description: (itil:hasServiceLifecycle itil:ITService itil:ServiceLifecycle) means that the *itil:ITService* is managed according to the *itil:ServiceLifecycle*.

Functional: Yes

Inverse: *itil:inITService*

Domain: *itil:ITService*

Range: *itil:ServiceLifecycle*

Subproperties: none

Property: hasServicePortfolioType

Ontology: ITIL (itil:)

Source: see the class *itil:ServicePortfolioType*.

Description: (itil:hasServicePortfolioType itil:ServicePortfolio itil:ServicePortfolioType) means that the *itil:ServicePortfolio* has the type *itil:ServicePortfolioType*.

Functional: Yes

Inverse: none

Domain: *itil:ServicePortfolio*

Range: *itil:ServicePortfolioType*

Subproperties: none

Property: hasSLAIncidentResolution

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: (itil:hasSLAIncidentResolution itil:SLA itil:SLAIncidentResolution) means that the *itil:SLAIncidentResolution* contains the specification of the incident resolution times for the *itil:SLA*.

Functional: No

Inverse: none

Domain: *itil:SLA*

Range: *itil:SLAIncidentResolution*

Subproperties: none

Property: hasSLP

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Service Package definition); OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO), p. 209.

Description: (itil:hasSLP itil:ServicePackage itil:SLP) means that the *itil:ServicePackage* is composed of the *itil:SLP*.

Functional: No

Inverse: none

Domain: *itil:ServicePackage*

Range: *itil:SLP*

Subproperties: none

Property: hasStage

Ontology: ITIL (itil:)

Source: see the class *itil:Lifecycle*.

Description: (itil:hasStage itil:Lifecycle itil:Stage) means that the *itil:Lifecycle* is composed of the *itil:Stage*.

Functional: No

Inverse: *itil:inLifecycle*

Domain: *itil:Lifecycle*

Range: *itil:Stage*

Subproperties:

(itil:hasServiceStage itil:ServiceLifecycle itil:ServiceStage) ->

inverse: (itil:inServiceLifecycle itil:ServiceStage itil:ServiceLifecycle)

Property: hasSupportingService

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Core Service definition and Supporting Service definition).

Description: (itil:hasSupportingService itil:CoreService itil:SupportingService) means that the *itil:CoreService* includes the *itil:SupportingService* to enable or enhance the delivery of the service.

Functional: No

Inverse: none

Domain: *itil:CoreService*

Range: *itil:SupportingService*

Subproperties: none

Property: hasTechnicalManagementType

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Operation*. The Stationery Office (TSO), p. 222-223.

Description: (itil:hasTechnicalManagementType itil:Event itil:TechnicalManagementType) means that the *itil:Event* has the type of intervention specified in *itil:TechnicalManagementType*.

Functional: No

Inverse: none

Domain: *itil:Event*

Range: *itil:TechnicalManagementType*

Subproperties: none

Property: implementedByApplication

Ontology: ITIL (itil:)

Source: see the class *itil:Activity*.

Description: (itil:implementedByApplication itil:Activity itil:Application) means that the *itil:Activity* is implemented by the *itil:Application*.

Functional: Yes

Inverse: *itil:implementsActivity*

Domain: *itil:Activity*

Range: *itil:Application*

Subproperties: none

Property: implementsActivity

Ontology: ITIL (itil:)

Source: see the class *itil:Activity*.

Description: (itil:implementsActivity itil:Application itil:Activity) means that the *itil:Application* implements the *itil:Activity*.

Functional: Yes

Inverse: *itil:implementedByApplication*

Domain: *itil:Activity*

Range: *itil:Application*

Subproperties: none

Property: inActivityGroup

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:inActivityGroup wf:Activity wf:Group/wf:Lane) means that the *wf:Activity* is included in the *wf:Group*/*wf:Lane*.

Functional: No

Inverse: *wf:hasActivities*

Domain: *wf:Activity*

Range:

wf:Group

wf:Lane

Subproperties: none

Property: inArtifactsContainer

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:inArtifactsContainer wf:Artifact wf: ArtifactsContainer) means that the *wf:Artifact* is part of the *wf:ArtifactsContainer*.

Functional: Yes

Inverse: *wf:hasArtifacts*

Domain: *wf:Artifact*

Range: *wf:ArtifactsContainer*

Subproperties: none

Property: inBpmnDiagram

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:inBpmnDiagram wf:MessagingEdge/wf:Pool wf:BpmnDiagram) means that the *wf:MessagingEdge*/*wf:Pool* is part of the *wf:BpmnDiagram*.

Functional: Yes

Inverse: *wf:diagramComposedOf*

Domain:

wf:MessagingEdge

wf:Pool

Range: *wf:BpmnDiagram*

Subproperties: none

Property: includesMeasurement

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 22.

Description: (itil:includesMeasurement itil:Metric itil:Measurement) means that the *itil:Measurement* is used in the computation of the *itil:Metric*. The *itil:Metric* may not be clear understood purely by their names. Usually these require a small definition or explanation such that the *itil:Metric* is understood. For this reason, *itil:Metric*(s) and their associated *itil:Measurement*(s) (calculations) should be documented.

Functional: No

Inverse: none

Domain: *itil:Metric*

Range: *itil:Measurement*

Subproperties: none

Property: includesPBA

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO), p. 206.

Description: (itil:includesPBA itil:UP itil:PBA) means that the user profile *itil:UP* supports the pattern of business activity *itil:PBA*.

Functional: No

Inverse: none

Domain: *itil:UP*

Range: *itil:PBA*

Subproperties: none

Property: incomingEdges

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:incomingEdges wf:Vertex wf:SequenceEdge) means that the *wf:Vertex* is the target of the *wf:SequenceEdge*.

Functional: No

Inverse: *wf:vertexTarget*

Domain: *wf:Vertex*

Range: *wf:SequenceEdge*

Subproperties: none

Property: incomingMessages

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:incomingMessages wf:MessageVertex wf:MessagingEdge) means that the *wf:MessageVertex* is the target of the *wf:MessagingEdge*.

Functional: No

Inverse: *wf:messageVertexTarget*

Domain: *wf:MessageVertex*

Range: *wf:MessagingEdge*

Subproperties: none

Property: inEvent

Ontology: ITIL (itil:)

Source: see the class *oc:Event*.

Description: (itil:inEvent oc:Event1 oc:Event2) means that *oc:Event1* is a part, or subevent, of *oc:Event2*.

Functional: Yes

Inverse: *oc:subEvents*

Domain: *oc:Event*

Range: *oc:Event*

Subproperties: none

Property: inGraph

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:inGraph wf:SequenceEdge/wf:Vertex wf:Graph) means that the *wf:SequenceEdge*/*wf:Vertex* are part of the *wf:Graph*.

Functional: Yes

Inverse: *wf:graphComposedOf*

Domain:

wf:SequenceEdge

wf:Vertex

Range: *wf:Graph*

Subproperties: none

Property: inITService

Ontology: ITIL (itil:)

Source: see the class *itil:ServiceLifecycle*.

Description: (itil:inITService itil:ServiceLifecycle itil:ITService) means that the *itil:ServiceLifecycle* is used for the management of the *itil:ITService*.

Functional: No

Inverse: *itil:inITService*

Domain: *itil:ServiceLifecycle*

Range: *itil:ITService*

Subproperties: none

Property: inLifecycle

Ontology: ITIL (itil:)

Source: see the class *itil:Stage*.

Description: (itil:inLifecycle itil:Lifecycle itil:Stage) means that the *itil:Stage* is part of the *itil:Lifecycle*.

Functional: No

Inverse: *itil:hasStage*

Domain: *itil:Stage*

Range: *itil:Lifecycle*

Subproperties:

(itil:inServiceLifecycle itil:ServiceStage itil:ServiceLifecycle) ->

inverse: (itil:hasServiceStage itil:ServiceLifecycle itil:ServiceStage)

Property: inPool

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:inPool wf:Lane wf:Pool) means that the *wf:Lane* is part of the *wf:Pool*.

Functional: Yes

Inverse: *wf:composedOfLanes*

Domain: *wf:Lane*

Range: *wf:Pool*

Subproperties: none

Property: inServicePortfolio

Ontology: ITIL (itil:)

Source: see the class *itil:ServicePortfolio*.

Description: (itil:inServicePortfolio itil:ITService itil:ServicePortfolio) means that the *itil:ITService* is part of the *itil:ServicePortfolio*.

Functional: No

Inverse: *itil:detailsITService*

Domain: *itil:ITService*

Range: *itil:ServicePortfolio*

Subproperties: none

Property: inServiceStage

Ontology: ITIL (itil:)

Source: see the class *itil:Process*.

Description: (itil:inServiceStage itil:Process itil:ServiceStage) means that the *itil:Process* is part of the *itil:ServiceStage*.

Functional: No

Inverse: *itil:hasProcess*

Domain: *itil:Process*

Range: *itil:ServiceStage*

Subproperties:

(itil:inStrategyStage itil:StrategyProcess itil:ServiceStrategy) ->

inverse: (itil:hasStrategyProcess itil:ServiceStrategy itil:StrategyProcess)

(itil:inDesignStage itil:DesignProcess itil:ServiceDesign) ->

inverse: (itil:hasDesignProcess itil:ServiceDesign itil:DesignProcess)

(itil:inTransitionStage itil:TransitionProcess itil:ServiceTransition) ->

inverse: (itil:hasTransitionProcess itil:ServiceTransition itil:TransitionProcess)

(itil:inOperationStage itil:OperationProcess itil:ServiceOperation) ->

inverse: (itil:hasOperationProcess itil:ServiceOperation itil:OperationProcess)

(itil:inCSIStage itil:CSIProcess itil:ContinualServiceImprovement) ->

inverse: (itil:hasCSIProcess itil:ContinualServiceImprovement itil:CSIProcess)

Property: interfaceValue

Ontology: ITIL (itil:)

Source: see the class *itil:InterfaceRelation*.

Description: (itil:interfaceValue itil:InterfaceRelation itil:Process) means that the *itil:InterfaceRelation* represents the *itil:Process*.

Functional: Yes

Inverse: none

Domain: *itil:InterfaceRelation*

Range: *itil:Process*

Subproperties: none

Property: isFeedback

Ontology: ITIL (itil:)

Source: OGC. (2007). The Official Introduction to the ITIL Service Lifecycle. The Stationery Office (TSO). London, p. 21-22.

Description: (itil:isFeedback itil:ServiceStage1 itil:ServiceStage2) means that the *itil:ServiceStage1* is feedback of *itil:ServiceStage2*.

Functional: No

Inverse: *itil:receivesFeedback*

Domain: *itil:ServiceStage*

Range: *itil:ServiceStage*

Subproperties: none

Property: managesServicePortfolio

Ontology: ITIL (itil:)

Source: see the class *itil:ITService*.

Description: (itil:managesServicePortfolio itil:ITServiceProvider itil:ServicePortfolio) means that the *itil:ServicePortfolio* is managed by the *itil:ITServiceProvider*.

Functional: Yes

Inverse: none

Domain: *itil:ITServiceProvider*

Range: *itil:ServicePortfolio*

Subproperties: none

Property: managedByProcess

Ontology: ITIL (itil:)

Source: see the class *itil:Event*.

Description: (itil:managedByProcess itil:ManagedEvent itil:Process) means that the *itil:Event* is managed by the *itil:Process*.

Functional: No

Inverse: *itil:managesEvent*

Domain: *itil:Event*

Range: *itil:Process*

Subproperties: none

Property: managesCI

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Design*. The Stationery Office (TSO), p. 204.

Description: (itil:managesCI itil:ITService itil:CI) means that the *itil:CI* is necessary to support the provision of the *itil:ITService* to the business.

Functional: No

Inverse: none

Domain: itil:ITService

Range: itil:CI

Subproperties: none

Property: managesEvent

Ontology: ITIL (itil:)

Source: see the class *itil:Process*.

Description: (itil:managesEvent itil:Process itil:Event) means that the *itil:Process* is the responsible for managing the *itil:ManagedEvent*.

Functional: No

Inverse: *itil:managedByProcess*

Domain: *itil:Process*

Range: *itil:Event*

Subproperties: none

Property: measuredBy

Ontology: ITIL (itil:)

Source: see the class *itil:Process*.

Description: (itil:measuredBy itil:Process itil:Metric) means that the *itil:Process* is measured by the *itil:Metric*.

Functional: No

Inverse: *itil:measures*

Domain: *itil:Process*

Range: *itil:Metric*

Subproperties: none

Property: measuredByKPI

Ontology: ITIL (itil:)

Source: see the class *itil:CSF*.

Description: (itil:measuredByKPI itil:CSF itil:KPI) means that the *itil:KPI* is used to measure the achievement of the *itil:CSF*.

Functional: No

Inverse: none

Domain: *itil:CSF*

Range: *itil:KPI*

Subproperties: none

Property: measures

Ontology: ITIL (itil:)

Source: see the class *itil:Metric*.

Description: (itil:measures itil:Metric itil:Process) means that the *itil:Metric* is used to measure the *itil:Process*.

Functional: No

Inverse: *itil:measuredBy*

Domain: *itil:Metric*

Range: *itil:Process*

Subproperties: none

Property: meetsPBA

Ontology: ITIL (itil:)

Source: see the class *itil:SLP*.

Description: (itil:meetsPBA itil:SLP itil:PBA) means that the *itil:SLP* meets the *itil:PBA*.

Functional: Yes

Inverse: none

Domain: *itil:SLP*

Range: *itil:PBA*

Subproperties: none

Property: messageVertexSource

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:messageVertexSource wf:MessagingEdge wf:MessageVertex) means that the *wf:MessageVertex* is the source of the *wf:MessagingEdge*.

Functional: Yes

Inverse: *wf:outgoingMessages*

Domain: *wf:MessagingEdge*

Range: *wf:MessageVertex*

Subproperties: none

Property: messageVertexTarget

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:messageVertexTarget wf:MessagingEdge wf:MessageVertex) means that the *wf:MessageVertex* is the target of the *wf:MessagingEdge*.

Functional: Yes

Inverse: *wf:incomingMessages*

Domain: *wf:MessagingEdge*

Range: *wf:MessageVertex*

Subproperties: none

Property: outgoingEdges

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:outgoingEdges wf: Vertex wf:SequenceEdge) means that the *wf:Vertex* is the source of the *wf:SequenceEdge*.

Functional: No

Inverse: *wf:vertexSource*

Domain: *wf:Vertex*

Range: *wf:SequenceEdge*

Subproperties: none

Property: outgoingMessages

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:outgoingMessages wf:MessageVertex wf:MessagingEdge) means that the *wf:MessageVertex* is the source of the *wf:MessagingEdge*.

Functional: No

Inverse: *wf:messageVertexSource*

Domain: *wf:MessageVertex*

Range: *wf:MessagingEdge*

Subproperties: none

Property: performedBy

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:performedBy oc:Action oc:Agent-Generic) means that the *oc:Agent-Generic* deliberately does *oc:Action*. Note that an *oc:Action* can have multiple deliberate performers (*oc:Agent-Generic*(s)).

Functional: No

Inverse: none

Domain: *oc:Action*

Range: *oc:Agent-Generic*

Subproperties: none

Property: processOwner

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Process Owner definition).

Description: (itil:processOwner itil:Process itil:RoleType) means that the *itil:RoleType* is the owner of the *itil:Process*. The process owner is a role responsible for ensuring that a process is fit for purpose. The process owner's responsibilities include sponsorship, design, change management and continual improvement of the process and its metrics. This role is often assigned to the same person who carries out the process manager role, but the two roles may be separate in larger organizations.

Functional: Yes

Inverse: none

Domain: *itil:Process*

Range: *itil:RoleType*

Subproperties: none

Property: programCode

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:programCode oc:ComputerProgram-CW oc:ComputerCode) means that the code *oc:ComputerCode* is source or executable code for the program *oc:ComputerProgram-CW*.

Functional: Yes

Inverse: none

Domain: *oc:ComputerProgram-CW*

Range: *oc:ComputerCode*

Subproperties: none

Property: programSpecifications

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:programSpecifications oc:ComputerProgram-CW oc:ProgramSpecification) means that the *oc:ProgramSpecification* specifies how the *oc:ComputerProgram-CW* should behave.

Functional: No

Inverse: none

Domain: *oc:ComputerProgram-CW*

Range: *oc:ProgramSpecification*

Subproperties: none

Property: providerOfService

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:providerOfService oc:ServiceEvent oc:Agent-Generic) means that the *oc:ServiceEvent* is performed or provided by the *oc:Agent-Generic*. Typically, the *oc:Agent-Generic* acts in order to serve the *oc:recipientOfService* in *oc:ServiceEvent*.

Functional: Yes

Inverse: none

Domain: *oc:ServiceEvent*

Range: *oc:Agent-Generic*

Subproperties: none

Property: proposesChange

Ontology: ITIL (itil:)

Source: see the class *itil:RFC*.

Description: (itil:proposesChange itil:RFC itil:Change) means that the request for change documented in the *itil:RFC* propose the *itil:Change* in the service.

Functional: Yes

Inverse: none

Domain: *itil:RFC*

Range: *itil:Change*

Subproperties: none

Property: receivesFeedback

Ontology: ITIL (itil:)

Source: OGC. (2007). The Official Introduction to the ITIL Service Lifecycle. The Stationery Office (TSO). London, p. 21-22.

Description: (itil:receivesFeedback itil:ServiceStage1 itil:ServiceStage2) means that the *itil:ServiceStage1* receives feedback from *itil:ServiceStage2*.

Functional: No

Inverse: *itil:isFeedback*

Domain: *itil:ServiceStage*

Range: *itil:ServiceStage*

Subproperties: none

Property: recipientOfService

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:recipientOfService oc:ServiceEvent oc:Agent-Generic) means that the *oc:Agent-Generic* is a recipient of the *oc:ServiceEvent*. Thus, the service in question is done for or performed on the *oc:Agent-Generic*, and the *oc:Agent-Generic* is correspondingly affected by it.

Functional: No

Inverse: none

Domain: *oc:ServiceEvent*

Range: *oc:Agent-Generic*

Subproperties:

(itil:doneForCustomer itil:ITService itil:Customer)

Property: requiresOperationalMetric

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing, p. 22.

Description: (itil:requiresOperationalMetric itil:KPI itil:OperationalMetric) means that the *itil:OperationalMetric* is needed to compute the *itil:KPI*. The *itil:KPI*(s) are calculated or derived from one or more *itil:OperationalMetrics*. The results of these calculations are then compared to an *itil:Tolerance* range to identify whether those results fall within acceptable levels.

Functional: No

Inverse: none

Domain: *itil:KPI*

Range: *itil:OperationalMetric*

Subproperties: none

Property: responsibleFor

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: If (oc:responsibleFor oc:Agent-Generic oc:Situation) holds at time 't', this means that, sometime prior to time 't' the *oc:Agent-Generic* deliberately performed an action which was instrumental in bringing about the *oc:Situation* to the extent that, other things being equal, if the action had not been performed, the *oc:Situation* would not have come about. This sense of 'responsibility' is stronger than causal responsibility, i.e., it requires that the *oc:Agent-Generic* play more than an unwitting causal role in bringing about the *oc:Situation*. However, it is probably weaker than full-blown moral responsibility, since even though the *oc:Agent-Generic* intended to perform the action which brought about the oc:Situation, the *oc:Agent-Generic* might not have intended to bring about the *oc:Situation*.

Functional: No

Inverse: none

Domain: *oc:Agent-Generic*

Range: *oc:Situation*

Subproperties: none

Property: roleAction

Ontology: ITIL (itil:)

Source: see the class *itil:RoleRelation*.

Description: (itil:roleAction itil:RoleRelation oc:PurposefulAction) means that the *itil:RoleRelation* is participating in the *oc:PurposefulAction*.

Functional: Yes

Inverse: none

Domain: *itil:RoleRelation*

Range: *oc:PurposefulAction*

Subproperties: none

Property: roleCode

Ontology: ITIL (itil:)

Source: see the class *itil:RoleRelation*.

Description: (itil:roleCode itil:RoleRelation itil:RoleType) means that the *itil:RoleRelation* has the type *itil:RoleType*.

Functional: Yes

Inverse: none

Domain: *itil:RoleRelation*

Range: *itil:RoleType*

Subproperties: none

Property: roleRACI

Ontology: ITIL (itil:)

Source: see the class *itil:RoleRelation*.

Description: (itil:roleRACI itil:RoleRelation itil:RACICode) means that the *itil:RoleRelation* has the *itil:RACICode*.

Functional: No

Inverse: none

Domain: *itil:RoleRelation*

Range: *itil:RACICode*

Subproperties: none

Property: source

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:source wf:Association wf: Artifact) means that the *wf:Artifact* is the source of the *wf:Association*.

Functional: Yes

Inverse: *wf:hasArtifacts*

Domain: *wf:Association*

Range: *wf: Artifact*

Subproperties: none

Property: specifiesActivity

Ontology: ITIL (itil:)

Source: see the class *oc:Specification*.

Description: (itil:specifiesActivity oc:Specification itil:Activity) means that the *oc:Specification* provides the description of the *itil:Activity*.

Functional: No

Inverse: *itil:coordinatedBySpecification*

Domain: *oc:Specification*

Range: *itil:Activity*

Subproperties: none

Property: subEvents

Ontology: OpenCyc (oc:)

Source: OpenCyc Browser.

Description: (oc:subEvents oc:Event1 oc:Event2) means that *oc:Event2* is a part, or subevent, of *oc:Event1*. The *oc:Event*(s) can be decomposed into subevents temporally, spatially, and in other ways.The *oc:subEvents* property is the most general instance of *oc:SubEventPredicate*. This predicate relates a given *oc:Event* to the *oc:Event*(s) that are its parts.

Functional: No

Inverse: *itil:inEvent*

Domain: *oc:Event*

Range: *oc:Event*

Subproperties: none

Property: supportedByOLA

Ontology: ITIL (itil:)

Source: see the class *itil:SLA*.

Description: (itil:supportedByOLA itil:SLA itil:OLA) means that the *itil:SLA* is supported by the *itil:OLA* in order to meet the service agreements.

Functional: No

Inverse: none

Domain: *itil:SLA*

Range: *itil:OLA*

Subproperties: none

Property: supportedByUC

Ontology: ITIL (itil:)

Source: see the class *itil:SLA*.

Description: (itil:supportedByUC itil:SLA itil:UC) means that the *itil:SLA* is supported by the *itil:UC* in order to meet the service agreements.

Functional: No

Inverse: none

Domain: *itil:SLA*

Range: *itil:UC*

Subproperties: none

Property: supportsITService

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO), p. 340.

Description: (itil:supportsITService itil:Application itil:ITService) means that the *itil:Application* is software that underpin the *itil:ITService*.

Functional: No

Inverse: *itil:hasApplication*

Domain: *itil:Application*

Range: *itil:ITService*

Subproperties: none

Property: supportsPBA

Ontology: ITIL (itil:)

Source: OGC. (2007). *ITIL Service Strategy*. The Stationery Office (TSO), p. 204.

Description: (itil:supportsPBA itil:ITService itil:PBA) means that the *itil:ITService* supports the pattern of business activity *itil:PBA*.

Functional: No

Inverse: none

Domain: *itil:ITService*

Range: *itil:PBA*

Subproperties: none

Property: target

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:target wf:Association wf: AssociationTarget) means that the *wf:AssociationTarget* is the target of the *wf:Association*.

Functional: Yes

Inverse: *wf:hasAssociations*

Domain: *wf:Association*

Range: *wf: AssociationTarget*

Subproperties: none

Property: undertakesActivity

Ontology: ITIL (itil:)

Source: see the class *itil:Event*.

Description: (itil:undertakesActivity itil:Event itil:Activity) means that the *itil:Event* undertakes the tasks defined in the *itil:Activity* in order to manage the related event.

Functional: Yes

Inverse: none

Domain: *itil:Event*

Range: *itil:Activity*

Subproperties: none

Property: usedForNegotiation

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload. (Service Level Requirement definition).

Description: (itil:usedForNegotiation itil:SLR itil:ServiceLevelTarget) means that the *itil:SLR* is used to negotiate the agreed *itil:ServiceLevelTarget*.

Functional: No

Inverse: *itil:basedOnSLR*

Domain: *itil:SLR*

Range: *itil: ServiceLevelTarget*

Subproperties: none

Property: vertexSource

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:vertexSource wf:SequenceEdge wf:Vertex) means that the *wf:Vertex* is the source of the *wf:SequenceEdge*.

Functional: Yes

Inverse: *wf:outgoingEdges*

Domain: *wf:SequenceEdge*

Range: *wf:Vertex*

Subproperties: none

Property: vertexTarget

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: (wf:vertexTarget wf:SequenceEdge wf:Vertex) means that the *wf:Vertex* is the target of the *wf:SequenceEdge*.

Functional: Yes

Inverse: *wf:incomingEdges*

Domain: *wf:SequenceEdge*

Range: *wf:Vertex*

Subproperties: none

Datatype Properties

Property: adhoc

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The specific value that represents whether a particular *wf:Subprocess* is adhoc.

Functional: Yes

Domain: *wf:SubProcess*

Range: boolean

Property: agentDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *oc:Agent-Generic*.

Functional: Yes

Domain: *oc:Agent-Generic*

Range: String

Property: agentName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *oc:Agent-Generic*.

Functional: Yes

Domain: *oc:Agent-Generic*

Range: String

Property: agreementCustomer

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to represent the specific customer of the *itil:Agreement*.

Functional: No

Domain: *itil:Agreement*

Range: String

Property: agreementDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:Agreement*.

Functional: Yes

Domain: *itil:Agreement*

Range: String

Property: agreementITServiceProvider

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to represent the specific IT service provider of the *itil:Agreement*.

Functional: Yes

Domain: *itil:Agreement*

Range: string

Property: agreementName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:Agreement*.

Functional: Yes

Domain: *itil:Agreement*

Range: string

Property: agreementResponsibility

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned that represents a responsability of a specific *itil:Agreement*.

Functional: No

Domain: *itil:Agreement*

Range: string

Property: agreementService

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned that represents a managed service in a specific *itil:Agreement*.

Functional: No

Domain: *itil:Agreement*

Range: string

Property: agreementTarget

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned that represents a target in a specific *itil:Agreement*.

Functional: No

Domain: *itil:Agreement*

Range: string

Property: appDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:Application*.

Functional: Yes

Domain: *itil:Application*

Range: string

Property: appName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:Application*.

Functional: Yes

Domain: *itil:Application*

Range: string

Property: ciDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:CI*.

Functional: Yes

Domain: *itil:CI*

Range: string

Property: ciName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:CI*.

Functional: Yes

Domain: *itil:CI*

Range: string

Property: computerLanguage

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The characther string assigned to represent the computer language used in a specific *oc:ComputerCode*.

Functional: Yes

Domain: *oc:ComputerCode*

Range: string

Property: corePackage

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The specific value that represents whether a particular *itil:ServicePackage* is considered a core package.

Functional: Yes

Domain: *itil:ServicePackage*

Range: boolean

Property: customerResponsibility

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The characther string assigned to represent a responsibility of an *itil:Customer* in a specific *itil:SLA*.

Functional: No

Domain: *itil:CustomerRelation*

Range: string

Property: diagramAuthor

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The characther string assigned to represent the author of a *wf:BpmnDiagram*.

Functional: Yes

Domain: *wf:BpmnDiagram*

Range: string

Property: diagramTitle

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The characther string assigned to represent the title of a *wf:BpmnDiagram*.

Functional: Yes

Domain: *wf:BpmnDiagram*

Range: string

Property: elementID

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The character string assigned to identify a specific *wf:Identifiable*.

Functional: Yes

Domain: *wf:Identifiable*

Range: string

Property: incidentImpact

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload; Pilot project documentation.

Description: The integer value that represents the impact of a specific *itil:Incident*. The *itil:incidentImpact* is a measure of the effect of an *itil:Incident*, *itil:Problem* or *itil:Change* on business processes. The *itil:incidentImpact* is often based on how service levels will be affected. The *itil:incidentImpact* and *itil:incidentUrgency* are used to assign *itil:incidentPriority*. In our pilot project, the impact represents the number of users affected by the *itil:Incident*.

Functional: Yes

Domain: *itil:IncidentRecord*

Range: int

Property: incidentLevel

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The integer value that represents the level of importance of a specific *itil:Incident*. In our pilot project, the *itil:IncidentLevel* is calculated from the *itil:incidentUrgency* and the *itil:incidentGroupType* that reported the *itil:Incident* (*itil:hasIncidentGroup* property). The level codes range from 0 to 5 (5 is the highest level of importance).

Functional: Yes

Domain: *itil:IncidentRecord*

Range: int

Property: incidentPriority

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload; Pilot project documentation.

Description: The integer value that represents the priority of a specific *itil:Incident*. The *itil:incidentPriority* is a category used to identify the relative importance of an *itil:Incident*, *itil:Problem* or *itil:Change*. The *itil:incidentPriority* is based on *itil:incidentImpact* and *itil:incidentUrgency*, and is used to identify required times for actions to be taken. For example an *itil:SLA* for a specific *itil:Customer* may state that *itil:Incident*(s) with *itil:incidentPriority* equals to 10 must be resolved within 12 hours. In our pilot project, the priority codes range from 0 to 10 (10 is the highest priority). The *itil:incidentPriority* is calculated from *itil:incidentImpact* and *itil:incidentLevel*.

Functional: Yes

Domain: *itil:IncidentRecord*

Range: int

Property: incidentResolution

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe the resolution of a specific *itil:Incident*.

Functional: Yes

Domain: *itil:IncidentRecord*

Range: string

Property: incidentResolutionDatetime

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string representing a point in time that designates the ending of the period of resolution for the *itil:Incident*. This field is expressed using a compacted ISO notation YYYYMMDDhhmmss.sss where YYYY represents a year in values from 0000 to 9999, MM represents a month in values from 00 to 12, and DD represents a day in values from 00 to 31, hh represents an hour in values from 00 to 23, mm represents a minute in values from 00 to 59, and ss.sss represents the number of seconds and milliseconds in values from 00.000 to 59.999. Note that all character positions must be filled.

Functional: Yes

Domain: *itil:IncidentRecord*

Range: string

Property: incidentStartDateTime

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string representing a point in time that designates the beginning of the period of resolution for the *itil:Incident*. This field is expressed using a compacted ISO notation YYYYMMDDhhmmss.sss where YYYY represents a year in values from 0000 to 9999, MM represents a month in values from 00 to 12, and DD represents a day in values from 00 to 31, hh represents an hour in values from 00 to 23, mm represents a minute in values from 00 to 59, and ss.sss represents the number of seconds and milliseconds in values from 00.000 to 59.999. Note that all character positions must be filled.

Functional: Yes

Domain: *itil:IncidentRecord*

Range: string

Property: incidentUrgency

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload; Pilot project documentation.

Description: The integer value that represents the urgency of a specific *itil:Incident*. The *itil:incidentUrgency* is a measure of how long it will be until an *itil:Incident*, *itil:Problem* or *itil:Change* has a significant impact on the business. The *itil:incidentImpact* and *itil:incidentUrgency* are used to assign *itil:incidentPriority*.The urgency codes range from 0 to 5 (5 is the highest urgency). The *itil:incidentUrgency* is calculated from the *itil:serviceImportanceCode*.

Functional: Yes

Domain: *itil:IncidentRecord*

Range: int

Property: interfaceRelationDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:InterfaceRelation*.

Functional: Yes

Domain: *itil:InterfaceRelation*

Range: string

Property: internalProvider

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The specific value that represents whether an *itil:ITServiceProvider* is an internal service provider in the organization.

Functional: Yes

Domain: *itil:ITServiceProvider*

Range: boolean

Property: internalService

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The specific value that represents whether an *itil:ITService* is an internal service of the IT service provider.

Functional: Yes

Domain: *itil:ITService*

Range: boolean

Property: isDefault

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The specific value that represents whether a particular *wf:SequenceEdge* is considered the default edge.

Functional: Yes

Domain: *wf:SequenceEdge*

Range: boolean

Property: isTransaction

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The specific value that represents whether a particular *wf:Subprocess* represents a transaction.

Functional: Yes

Domain: *wf:SubProcess*

Range: boolean

Property: looping

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The specific value that represents whether a particular *wf:Activity* represents a loop.

Functional: Yes

Domain: *wf:Activity*

Range: boolean

Property: lifecycleDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:Lifecycle*.

Functional: Yes

Domain: *itil:Lifecycle*

Range: string

Property: lifecycleName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:Lifecycle*.

Functional: Yes

Domain: *itil:Lifecycle*

Range: string

Property: measureDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:Measurement*.

Functional: Yes

Domain: *itil:Measurement*

Range: String

Property: measureName
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The character string assigned to name a specific *itil:Measurement*.
Functional: Yes
Domain: *itil:Measurement*
Range: String

Property: metricDescription
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The character string assigned to describe a specific *itil:Metric*.
Functional: Yes
Domain: *itil:Metric*
Range: string

Property: metricName
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The character string assigned to name a specific *itil:Metric*.
Functional: Yes
Domain: *itil:Metric*
Range: string

Property: metricValue
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The numeric value assigned to represent the value of a specific *itil:Metric*.
Functional: Yes
Domain: *itil:Metric*
Range: float

Property: objectDocumentation

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The character string assigned that represents the documentation of a specific *wf:NamedBpmnObject*.

Functional: Yes

Domain: *wf:NamedBpmnObject*

Range: string

Property: objectName

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The character string assigned to name a specific *wf:NamedBpmnObject*.

Functional: Yes

Domain: *wf:NamedBpmnObject*

Range: string

Property: objectNcname

Ontology: Workflow (wf:)

Source: BPMN Modeler website: http://www.eclipse.org/bpmn/.

Description: The character string assigned that represents the nickname of a specific *wf:NamedBpmnObject*.

Functional: Yes

Domain: *wf:NamedBpmnObject*

Range: string

Property: packageDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:ServicePackage*.

Functional: Yes

Domain: *itil:ServicePackage*

Range: string

Property: packageName
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The character string assigned to name a specific *itil:ServicePackage*.
Functional: Yes
Domain: *itil:ServicePackage*
Range: string

Property: pbaDescription
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The character string assigned to describe a specific *itil:PBA*.
Functional: Yes
Domain: *itil:PBA*
Range: string

Property: pbaName
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The character string assigned to name a specific *itil:PBA*.
Functional: Yes
Domain: *itil:PBA*
Range: string

Property: portfolioDescription
Ontology: ITIL (itil:)
Source: Pilot project documentation.
Description: The character string assigned to describe a specific *itil:ServicePortfolio*.
Functional: Yes
Domain: *itil:ServicePortfolio*
Range: string

Property: portfolioName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:ServicePortfolio*.

Functional: Yes

Domain: *itil:ServicePortfolio*

Range: string

Property: preApprovedChange

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The specific value that represents whether an *itil:Change* is considered a standard change that requires a little effort to implement, carries a low level of risk, has pre-defined approval and does not require the intervention of the CAB (other changes require the approval of the CAB).

Functional: Yes

Domain: *itil:Change*

Range: boolean

Property: processChallenge

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the challenges for successful *itil:Process*.

Functional: No

Domain: *itil:Process*

Range: string

Property: processInput

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the input of an *itil:Process*.

Functional: No

Domain: *itil:Process*

Range: string

Property: processName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:Process*.

Functional: Yes

Domain: *itil:Process*

Range: string

Property: processObjective

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The characther string assigned to represent the objectives of a specific *itil:Process*. The *itil:processObjective* is the defined purpose or aim of an *itil:Process*, an *itil:Activity* or an *oc:Organisation* as a whole. The *itil:processObjective*(s) are usually expressed as measurable targets.

Functional: No

Domain: *itil:Process*

Range: string

Property: processOutput

Ontology: ITIL (itil:)

Source: itSMF International. (2007a). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the output of an *itil:Process*.

Functional: No

Domain: *itil:Process*

Range: string

Property: processRisk

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The characther string assigned to represent the risks that may be encountered with a specific *itil:Process*. An *itil:processRisk* is a possible *itil:Event* that could cause harm or loss, or affect the ability to achieve *itil:processObjective*(s). An *itil:processRisk* is measured by the probability of a threat, the vulnerability of the asset to that threat, and the *itil:incidentImpact* it would have if it occurred.

Functional: No

Domain: *itil:Process*

Range: string

Property: processScope

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *ITIL V3: Glossary of Terms and Definitions*. Version to Workload.

Description: The characther string assigned to represent the scope of a specific *itil:Process*. The *itil:processScope* is the boundary, or extent, to which an itil:Process, applies.

Functional: No

Domain: *itil:Process*

Range: string

Property: processTechnology

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the technology required to deliver and support a specific *itil:Process*. For example, data storage technology such as storage devices (disks, controllers, tapes, etc.) and *Storage Area Networks* (SANs), designed to attach computer storage devices.

Functional: No

Domain: *itil:Process*

Range: string

Property: processValueToBusiness

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the value of a specific *itil:Process*.

Functional: No

Domain: *itil:Process*

Range: string

Property: questionBeingAnswered

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing (Chapter 4 – Chapter 15).

Description: The character string assigned to represent the question that a specific *itil:KPI* is trying to answer.

Functional: Yes

Domain: *itil:KPI*

Range: string

Property: serviceImportanceCode

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The integer value that represents the importance code of a specific *itil:ITService*. In our pilot project, the importance codes range from 0 to 5 (5 is the highest importance).

Functional: Yes

Domain: *itil:ITService*

Range: int

Property: serviceProviderResponsibility

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent a responsibility of an *itil:ITServiceProvider* in a specific *itil:SLA*.

Functional: No

Domain: *itil:ITServiceProviderRelation*

Range: string

Property: serviceStageObjective

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the objective of a specific *itil:ServiceStage*.

Functional: No

Domain: *itil:ServiceStage*

Range: string

Property: serviceStageScope

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the scope of a specific *itil:ServiceStage*.

Functional: No

Domain: *itil:ServiceStage*

Range: string

Property: serviceStageValueToBusiness

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The characther string assigned to represent the value of a specific *itil:ServiceStage*.

Functional: No

Domain: *itil:ServiceStage*

Range: string

Property: serviceUsers

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The integer value that represents the number of users of a specific *itil:ITService*.

Functional: Yes

Domain: *itil:ITService*

Range: int

Property: situationDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *oc:Situation*.

Functional: Yes

Domain: *oc:Situation*

Range: string

Property: situationName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *oc:Situation*.

Functional: Yes

Domain: *oc:Situation*

Range: string

Property: slaIncidentPriority

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The integer value that represents the agreed priority of a specific *itil:SLA*. In our pilot project, the priority codes range from 0 to 10 (10 is the highest priority).

Functional: Yes

Domain: *itil:SLAIncidentResolution*

Range: int

Property: slaIncidentResolutionTime

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The integer value that represents the agreed resolution time for a specific priority (*itil:slaIncidentPriority*). The unit of measure is hours.

Functional: Yes

Domain: *itil:SLAIncidentResolution*

Range: int

Property: slpDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:SLP*.

Functional: Yes

Domain: *itil:SLP*

Range: string

Property: slpName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:SLP*.

Functional: Yes

Domain: *itil:SLP*

Range: string

Property: slrBusinessObjective

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The character string assigned that represents the business objective of a specific *itil:SLR*.

Functional: No

Domain: *itil:SLR*

Range: string

Property: slrDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:SLR*.

Functional: Yes

Domain: *itil:SLR*

Range: string

Property: slrName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:SLR*.

Functional: Yes

Domain: *itil:SLR*

Range: string

Property: slrResponsibility

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The character string assigned that represents the responsibility of a specific *itil:SLR*.

Functional: No

Domain: *itil:SLR*

Range: string

Property: slrTarget

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The character string assigned that represents the target of a specific *itil:SLR*.

Functional: No

Domain: *itil:SLR*

Range: string

Property: specDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *oc:Specification*.

Functional: Yes

Domain: *oc:Specification*

Range: string

Property: specName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *oc:Specification*.

Functional: Yes

Domain: *oc:Specification*

Range: string

Property: stageDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:Stage*.

Functional: Yes

Domain: *itil:Stage*

Range: string

Property: stageName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:Stage*.

Functional: Yes

Domain: *itil:Stage*

Range: string

Property: targetDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:ServiceLevelTarget*.

Functional: Yes

Domain: *itil:ServiceLevelTarget*

Range: string

Property: targetName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:ServiceLevelTarget*.

Functional: Yes

Domain: *itil:ServiceLevelTarget*

Range: string

Property: toleranceCode

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: The numeric value assigned to represent a specific tolerance of an *itil:KPI*.

Functional: Yes

Domain: *itil:Tolerance*

Range: float

Property: toleranceServiceTarget

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: The numeric value assigned to represent a specific service target tolerance (acceptable value) of an *itil:KPI*.

Functional: Yes

Domain: *itil:Tolerance*

Range: float

Property: toleranceWarningLevel

Ontology: ITIL (itil:)

Source: Steinberg, R.A. (2006). *Measuring ITIL: Measuring, Reporting and Modeling - the IT Service Management Metrics That Matter Most to IT Senior Executives*. Trafford Publishing.

Description: The numeric value assigned to represent a specific warning level tolerance (non-acceptable value) of an *itil:KPI*.

Functional: Yes

Domain: *itil:Tolerance*

Range: float

Property: upDescription

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to describe a specific *itil:UP*.

Functional: Yes

Domain: *itil:UP*

Range: string

Property: upName

Ontology: ITIL (itil:)

Source: Pilot project documentation.

Description: The character string assigned to name a specific *itil:UP*.

Functional: Yes

Domain: *itil:UP*

Range: string

Property: urgentChange

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The specific value that represents whether an *itil:Change* is considered urgent in order to restore a service after the identification of a problem. The *itil:Change* must be introduced as soon as possible to alleviate or avoid detrimental impact on the business.

Functional: Yes

Domain: *itil:Change*

Range: boolean

Property: visibleToCustomer

Ontology: ITIL (itil:)

Source: itSMF International. (2007). *Foundations of IT Service Management Based on ITIL V3.* Van Haren Publishing.

Description: The specific value that represents whether a particular *itil:ITService* represents a third-party service that is visible to the customers.

Functional: Yes

Domain: *itil:ITService*

Range: boolean

Appendix III

Glossary

AMIS ≡ Availability Management Information System

API ≡ Application Programming Interface

AST ≡ Agreed Service Time

B2B ≡ Business-to-Business

BCP ≡ Business Continuity Plan

BEDSL ≡ Business Entities Domain-Specific Language

BIA ≡ Business Impact Analysis

BPD ≡ Business Process Diagram

BPDM ≡ Business Process Definifion Metamodel

BPO ≡ Business Process Owner

BPSS ≡ Business Process Specification Schema

BPE ≡ Business Process Engineering

BPM ≡ Business Process Modeling/Management

BPMI ≡ Business Process Management Initiative

BPMN ≡ Business Process Model and Notation

BRM ≡ Business Relationship Manager

BSM ≡ Business Service Management

BWW ≡ Bunge-Wand-Weber

CAB ≡ Change Advisory Board

CARS ≡ Compliance, Audit, Risk and Security

CASE ≡ Computer-Aided Software Engineering

CAU ≡ Centro de Atención al Usuario

CCTA ≡ Central Computer of Telecommunications Agency

CEO ≡ Chief Executive Officer

CHA ≡ Chief Architect

CFIA ≡ Component Failure Impact Analysis

CI ≡ Configuration Item

CIM ≡ Computation Independent Model

CIO ≡ Chief Information Officer

CMDB ≡ Configuration Management Database

CMIS ≡ Capacity Management Information System

CMMI ≡ Capability Maturity Model Integration

CMS ≡ Configuration Management System

COBIT ≡ Control Objectives for Information and related Technology

CPA ≡ Collaboration Protocol Agreement

CSF ≡ Critical Success Factor

CSI ≡ Continual Service Improvement

CSIP ≡ Continual Service Improvement Plan

CWA ≡ Closed World Assumption

DL ≡ Description Logics

DSL ≡ Domain-Specific Language

DSM ≡ Domain-Specific Modeling

EDOC ≡ Enterprise Distributed Object Computing

EMF ≡ Eclipse Modeling Framework

EPC ≡ Event Process Chain

eTOM ≡ Enhanced Telecom Operations Map

GMF ≡ Graphical Modeling Framework

GMP ≡ Graphical Modeling Project

GPL ≡ General Purpose Language

HA≡ Head IT Administration

HD ≡ Head Development

HO ≡ Head Operations

ICTD ≡ Information and Communication Technology Department

IE ≡ Information Engineering

IEC ≡ International Electrotechnical Commision

IM ≡ Incident Management / Incident Manager

ISMS ≡ Information Security Management System

ISO ≡ International Organization for Standardization

IT ≡ Information Technology

ITIL ≡ Information Technology Infrastructure Library

ITIMF ≡ Information Technology Information Management Forum

ITSCM ≡ Information Technology Service Continuity Management

ITSM ≡ Information Technology Service Management

itSMF ≡ IT Service Management Forum

ITSMS ≡ Information Technology Service Management System

JMI ≡ Java Metadata Interface

KBSI ≡ Knowledge Based Systems Inc.

KPI ≡ Key Performance Indicator

LOS ≡ Line of Service

M2M ≡ Model-to-Model

M2T ≡ Model-to-Text

MAS ≡ Multi-Agent Systems

MDA ≡ Model-Driven Architecture

MDE ≡ Model-Driven Engineering

MDD ≡ Model-Driven Development

MIS ≡ Management Information Systems

MOF ≡ Meta Object Facility

MTBF ≡ Mean Time Between Failures

MTRS ≡ Mean Time to Restore Service

MTTR ≡ Mean Time To Repair

OASIS ≡ Organization for Advancement of Structured Information Standards

OCL ≡ Object Constraint Language

OE ≡ Ontology Engineering

OGC ≡ Office of Government Commerce

OLA ≡ Operational Level Agreement

OMG ≡ Object Management Group

OMT ≡ Object Modeling Technique

OO ≡ Object-Oriented

OWA ≡ Open World 0Assumption

OWL ≡ The Web Ontology Language

OWL-S ≡ The Web Ontology Language for Services

PBA ≡ Pattern of Business Activity

PDCA ≡ Plan–Do–Check–Act

PIM ≡ Platform Independent Model

PMBOK ≡ Project Management Body of Knowledge

PR ≡ Problem Record

PSM ≡ Platform Specific Model

QVT ≡ Query/View/Transformation

RA ≡ Risk Analysis

RACER ≡ Renamed Abox and Concept Expression Reasoner

RBSLM ≡ Rule-Based Service Level Management

RDF ≡ Resource Description Framework

RDF-S ≡ RDF Schema

REA ≡ Resource Event Agent

REFSENO ≡ Representation Formalism for Software Engineering Ontologies

RFC ≡ Request for Change

SACM ≡ Service Asset and Configuration Management

SAN ≡ Storage Area Network

SCA ≡ Sustained Competitive Advantage

SCD ≡ Supplier and Contract Database

SDP ≡ Software Development Process

SE ≡ Software Engineering

SIP ≡ Service Improvement Plan

SKMS ≡ Service Knowledge Management System

SLA ≡ Service Level Agreement

SLM ≡ Service Level Management

SLP ≡ Service Level Package

SLR ≡ Service Level Requirement

SMART ≡ Specific, Measurable, Appropriate, Realistic and Time-bound

SME ≡ Small and Medium-sized Enterprise

SMO ≡ Software Measurement Ontology

SOA ≡ Service Oriented Architecture

SPM ≡ ServicePortfolioManagement

SPO ≡ Service Provisioning Optimization

SQWRL ≡ Semantic Query-Enhanced Web Rule Language

SSU ≡ Shared Services Unit

SUMO ≡ Suggested Upper Merged Ontology

SuS ≡ System under Study

SWRL ≡ Semantic Web Rule Language

TSO ≡ The Stationery Office

UML ≡ Unified Modeling Language

UP ≡ User Profile

VCD ≡ Variable Cost Dynamics

W3C ≡ World Wide Web Consortium

WfMC ≡ Workflow Management Coalition

WSBPEL ≡ Web Services Business Process Execution Language

WSDL ≡ Web Services Description Language

XML ≡ Extensible Markup Language