



Interface Design for IT Service Management Practice

João Nabais, Alexandre Miguel Pinto
CISUC – University of Coimbra
3030-290 Coimbra, Portugal
jl nabais@student.dei.pt, ampinto@dei.uc.pt

António Cruz
SAPO – Portugal Telecom
1069-300 Lisboa, Portugal
antonio.j.cruz@telecom.pt

Jorge Cardoso
Karlsruhe Institute of Technology
76131 Karlsruhe, Germany
jorge.cardoso@kit.edu

Abstract—As the worldwide economy becomes increasingly service-based, companies have a growing need for the adoption of IT Service Management (ITSM) best practices, tools, and methodologies. The Information Technology Infrastructure Library (ITIL) is a set of best practices in ITSM and is now highly adopted by the industry. However, implementing ITIL is complex and costly, and enterprises that design, adopt and provide ITSM services, often end up with different analysis methods and designs for similar ITSM solutions.

Aiming at reducing costs and standardizing ITIL-based implementations we propose a methodology to build ITIL conformant interfaces for its processes and functions. This methodology aims for the standardization and partial automation of the construction of reusable software components that are ITIL conformant. It promotes software reuse for ITSM /ITIL solutions market, through the future development of ITSM interfaces.

Keywords—ITSM, ITIL, Service, Interface design

I. INTRODUCTION

IT Service Management (ITSM) has been evolving during the last decades. It started simply by taking advantage of new technologies for delivering applications as part of service offerings, supporting the business, but in the course of time it became clear that businesses needed a more encompassing and value creating approach. E.g., the notion of the “IT help desk” service emerged in order to deal with frequent user issues. In the last three decades a set of best-practices, processes and functions was compiled into what is known today as the IT Infrastructure Library (ITIL) [1,2]. ITIL appeared as an answer to the need of efficiency in IT service management, based on the service management know-how of the best and most successful organizations [1]. When properly implemented, ITIL allows organizations to provide services with greater efficiency, effectiveness, quality and cost reduction [3]. As a result, ITIL is the most adopted framework by worldwide organizations and it is still growing. By June of 2011 it was estimated that ITIL adoption had increased twenty percent per annum and the number of ITIL training attendees increased at a rate of thirty percent over the last decade [4]. ITIL provides a set of documented best ITSM practices, but only in a

descriptive form. Companies that want to develop ITIL best practices, either for internal use, or to be part of a service offering, have to do it from scratch, including the development of specific methodologies used to design, analyze and develop specific ITSM solutions.

A. Motivation

Problem: Some concerns arise when organizations want to fully, or partially, adopt ITIL best practices, not only because it comes with high hiring or certification costs, but also due to the time, effort and structural costs that ITSM implementations imply. We propose a solution to this problem, towards a cost reduction of ITIL practices implementation.

Contribution: Our work aims at the development of an interface design methodology for ITSM practices. This methodology can help ease companies’ workload throughout the analysis of ITIL best practices (processes and functions), the identification of ITSM process elements and of logic operations within such practices. This work ultimately contributes to the development of decoupled ITSM practices oriented web-services that can serve a variety of IT consumers.

Many companies buy full ITSM solutions from third party software vendors instead of consuming only the subset of web services, which correspond to the strictly desired ITSM functionalities they need. A wide range of entities will benefit with the successful realization of this work, mostly because we present a new way to ease ITIL/ITSM implementation allowing an easier, and more flexible way to get value, from an ITSM perspective.

We contribute, as well, to a normalization of the ITIL levels motivated by the study from November 2011 [4], by the AMP Group, which points that: “ITIL adoption levels are clearly different around the world”. The interfaces we developed, based in the methodology we also present, can serve as a basis for such normalization, as companies adopting them, fully or partially, could compare adoption levels between them with less effort, since the fundamental practices are at least similar.

Business Impact: With our approach, ITSM consumers can circumvent part of the efforts in time, resources and

financial costs, by adopting a set of normalized interfaces built with the methodology proposed.

ITSM is already a business on its own right. According to [14], “customers are outsourcing the delivery and support of their IT Services to Telecommunications Companies (TelCos), so TelCos are offering IT Service Management as a sellable service”. TelCos are already playing a big role in IT service management. They can make use of IT service components in order to manage their sellable services, so they can act as IT service providers as well as ITSM service providers. Also, exposing these services as SaaS offers through Web APIs would enable new business opportunities to service providers and solution integrators. This kind of offerings enables the rapid development of applications acting as clients of ITIL services, e.g., mobile apps that can easily integrate with Incident Management, Event Management, or Service Level Management services, delivering extra value to the end user. However, the IT service components can only be most efficiently used as building blocks of ITSM services if their behavior is unambiguously described. A fundamental mechanism for describing the behavior of these ITSM service components is the specification of their interfaces. Our present work serves precisely the purpose of providing a methodology for the specification of ITSM services, and in that sense it serve as a catalyst for ITSM services development.

Objectives:

In partnership with Portugal Telecom/SAPO, we aim at creating and developing a methodology for designing ITIL (or ITSM) practices, and model and build a set of Abstract IT service components [15] (ITIL Interfaces). Despite the fact that such Abstract IT service components are tailored for ITSM, we will specifically use them to map ITIL practices, and since such are specific instances of IT service components; they can have an associated Abstract IT service component (interface). The template we develop associated with each interface includes a process flow diagram schema of the respective component, a description of the component, a cross-functional flowchart diagram and the associated ITIL information object [12] (Transitions: Inputs & Outputs). By having enough information regarding transitions, processes and services, these templates allow the usage of a “Plug-and-Process” paradigm, which consists in “reusability, plug-replaceability [...] and extensibility” [10].

Another benefit of this work, in the field of IT Service Management, is a decrease of the difficulty in measuring the ITIL adoption levels. Despite not being the central concern of this work, it can help to lead to a standardization of ITIL adoption levels, easing the comparison between organization ITSM practices, since there is no need for surveys if they both use a similar set of interfaces.

B. Methodology

We describe our methodology as a sequence of steps that should be fulfilled in order to develop a set of artifacts that can be used to specify a set of interface logical operations.

Firstly, an ITSM practice process must be developed, through the analysis of ITSM practices and its elements, and it should be specified using a notation like, e.g., BPMN [16] or

EPC or UML diagrams [18], in order document the associated business process. After the process is specified, its elements must be analyzed in an operational-centric way to identify more granular logic activities within the ITSM process that will serve as a basis for the interface operations. After these “primitive” operations are defined, their behavior, as well as their inputs and outputs, must be determined and the data types needed to feed and store ITSM information must be defined. Lastly team revisions of such interfaces must be held and the identified improvement needs must be fulfilled.

C. Paper Structure

The rest of the paper is organized as follows. Section II presents the Background Notion including relevant concepts, as well as previous work developed within this scope. Section III presents in detail the ITSM Practices Interface Design, which is followed by Section IV describing the evaluation of the developed methodology. Conclusions of the work are presented with a critical review and identification of future work in Section V.

II. BACKGROUND NOTIONS

A. Components and Interfaces

Component: Component-based Software Engineering (CBSE) emerged, and it is described by Sommerville [5], as “the process of defining, implementing, and integrating or composing loosely coupled, independent components into systems.”.

Our work takes into consideration an essential point of CBSE: “there should be a clear separation between the component interface and its implementation. This means that one implementation of a component can be replaced by another, without changing other parts of the system” [5]. This is one of the reasons why we only provide a methodology for interface design: because some ITSM companies might want to develop, or change, specific component implementations, (e.g., in the Incident Management ITIL process, the incident matching activity implementation might differ between two distinct service providers).

Since we aim to provide a way to design interfaces (software components), we begin by clarifying exactly what do we understand by “component” in order to provide an overall view of where ITIL specific interfaces stand, at a scientific level. We considered the two definitions pointed out by Sommerville’s work.

Definition 1. “A software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard.” [6].

Definition 2. “A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.” [7].

Component as a Service: Sommerville states, regarding the above definitions, that “both of these definitions are based on the notion of a component as an element that is included in

a system, rather than a service that is referenced by the system". He also points that a "notion of a component as a service was developed" in response to problems such as the standards and protocols that "have hindered the uptake of CBSE", since they are "complex and difficult to understand". Therefore, if components are services, the reuse or integration of processes will be eased, since we do not need to have concerns, or constraints, regarding the connection of different components using different technologies (e.g., .NET and J2EE).

In light of the above, and since we need a definition of component more from a service perspective, we based our definition in the "critical characteristics" of reusable components, pointed out by Sommerville, and come up with the following definition:

Definition 3. A component is an independent and executable entity that is defined by interfaces, abstracting itself from source code, which could be referenced as an external service or included directly in a program.

Abstract IT Component: The components, or services, discussed above should be defined, described and specified as Abstract IT Components, which are defined as:

Definition 4. Abstract IT Components are templates used for specific instances of IT service components. [8].

Therefore, we adopt the methodology we present in the sequel, to define, describe and specify components and the related ITIL interfaces as Abstract IT Components.

ITIL Component: Fry [9] considers the 26 processes and 4 functions (Fig. 1) that one can choose in order to properly implement ITIL. These are dubbed in his work as "ITIL 2011 components". Fry categorizes such components into four distinct categories:

- **Action Components** - "require actions of operational nature to be performed as part of their normal functionality", (e.g. Service Desk, Incident Management, Problem Management, Event Management, etc.);
- **Influencing Components** - "modify and influence the way that action components perform their actions" (e.g. Service Level Management, Service Validation and testing, Service Catalogue Management, etc.);
- **Resourcing Components** - "ensure that the other components have the resources to meet their service commitments" (e.g. Capacity Management, Availability Management, Transition Planning and Support, etc.);
- **Underpinning Components** - "provide the underpinning facilities required by all components. Some of these components, such as financial management, may also serve other areas of IT." (e.g. Financial Management, IT Service Continuity Management, Strategy Generation, etc.).

Taking all the above into account, we come up with the following definition:

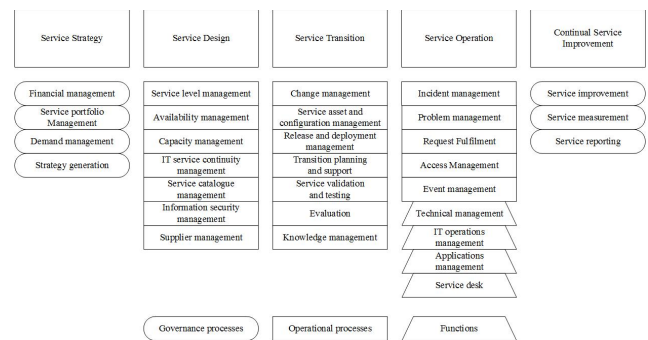


Figure 1. ITIL components

Definition 5. An ITIL component is a component that fulfills and materializes all needed functionalities related to an ITIL process or function.

ITSM Component: The notion, inspired by Fry's work, of "ITIL component", can be abstracted to "ITSM Component" and therefore an ITSM service can be considered a component that implements ITSM (not obligatorily ITIL). Such a component reflects a practice that some company adopts in order to manage specific IT services.

This definition (5) does not conflict with the one provided previously (3) for "component" since ITSM services (or components) can be defined by interfaces and could be referenced as external services or included directly into a program. It is important to define component at an ITSM level, since some companies are not interested in general best practices (ITIL), and therefore they look for ITSM practices of other companies in the same business sector. If such sector specific best practices have already been specified there is no reason to want to adopt a generic ITIL solution instead of a proven solution, within the same business scope, generating a win-win situation for both consumer and provider companies.

It is important to notice that an ITSM component can be composed by an individual ITSM function or process interface or by a set of ITSM interfaces, so the degree of atomicity can vary from component to component. For instance, Gartner¹ uses the term IT service support management (ITSSM) to refer to a specific set of ITSM practices.

Interface: As Sommerville states [5], "the services offered by a component are made available through an interface and all interactions are through that interface" and "an important part of any design process is the specification of the interfaces between the components in the design". Hence, we need to clarify what we do understand by "interface". In order to do so we considered the following definitions:

Definition 6. "An abstraction of the behavior of a component that consists of a subset of the interactions of that component together with a set of constraints describing when they may occur. The interface describes the behavior of a component that is obtained by considering only the interactions of that interface and by hiding all other interactions." [6].

¹ <http://www.gartner.com/it-glossary/itssm-tools-it-service-support-managementtools>

Definition 7. “A contract, in a form of a collection of operations definitions, which provides a mechanism for a clear separation between an external and internal view of a determined element and allows establishing “client-provider” relationship mediated by the notion of “contract”.” [11].

Definition 8. “Interface is the description of the signatures of a set of operations that are available to the service client for invocation.” [10].

In light of the above we consider an interface to be:

Definition 9. An abstraction and description of the behavior, more specifically operations (in the form of signatures) regarding a software component, providing a clear separation between external and internal view.

ITSM interfaces should provide information based on Fry’s [9] Activities, Transmissions, Work Instructions and Control and Quality metrics. Each ITIL interface is specified using a standard template for specific ITIL components (Abstract IT Component) and within it, information regarding the ITIL component transmissions (inputs and outputs), in other words, ITIL information objects.

ITIL Information Object: As Sommerville states, “The services offered by a component are made available through an interface and all interactions are through that interface. The component interface is expressed in terms of parameterized operations and its internal state is never exposed.”

Such interactions, or as Fry [9] defines it, such transmissions, between operations, or activities, can be performed in two ways, as input, or output. Therefore every Abstract IT Component has present, in its interface, such transmissions.

Kempter&Kempter[12] dub such input and output flows as ITIL Information Objects, posteriorly reused by Wang [13]. There is a necessity of including this definition, since “process levels are usually contemplated as a step of a separate project before getting involved in process internal parts in detail. Indeed, before being able to introduce detailed activities, should be clarified what outputs need to be produced by a process, and what inputs a process ought to expect prior processes”.

B. Coupling and Cohesion

When implementing interfaces that, ultimately, will result in services, certain service design principles must be kept in mind.

As in [10] we consider “two well-known software design guidelines: coupling and cohesion” since these “guarantee that services are self-contained, modular and able to support service composability”. ITIL (or ITSM) interfaces will represent actions (atomic or not), and interactions between them, as well as the order they follow, represented by business processes.

Coupling: In terms of service coupling “the objective is to minimize coupling, that is, to make (self-contained) business processes as independent as possible by not having any knowledge of, or relying on, any other business processes” [10]. This overlaps with the definition of discrete process used by Fry in his work [9]: “A discrete process is a stand-alone

process that can be completed in a linear fashion without impact from another process.” As explained by de Champeaux, Lea and Faure [18], and reused by Papazoglou and Yang [10]: “The central tactic” for low coupling “stems from the idea of abstract classes in object-oriented design where composite classes and actions minimize dependencies on irrelevant representational and computational details”.

Cohesion: In terms of cohesion, during the development of ITIL interfaces, we must create “strong, highly cohesive business processes, business processes whose services and service operations are strongly and genuinely related to one another.” [10]. Since ITIL publications already describe some processes of the components (services), operations (activities) and transmissions, such components and operations are already related to one another.

If there is not a specified ITSM process, like the ones “recommended” by ITIL publications, a custom ITSM process must be built.

III. ITSM PRACTICES INTERFACE DESIGN

As described in Section I.B, the interface design methodology is divided in the following set of steps:

- A) Build the ITSM Process;
- B) Specify the ITSM Process Flow;
- C) Identify Operations, Inputs, Outputs and Data Types;
- D) Discuss and Revise the Specification.

In the next sections, we explain in detail each of these steps.

A. Build the ITSM processes

When implementing processes specific to some ITSM practice there is a need of having a formalized process. In the case of ITIL some of the processes are already explicitly presented in the ITIL publications, such as Incident Management, Problem Management, Request Fulfillment, etc. However, if there is not a specific explicit process defined, the person developing the interface should use a methodology to build such process.

Some work on building an ITIL process is described by Fry [9]. Despite it being specific for ITIL processes, it can be used for general ITSM practices if the person building the interface already knows the behavior of the process of such practice.

First, Fry advises to answer five key questions in order to extrapolate the activities from an ITIL (ITSM) practice (Figure 2). Posteriorly, he provides a process describing a methodology in order to build an ITIL (or ITSM) process (Figure 3).

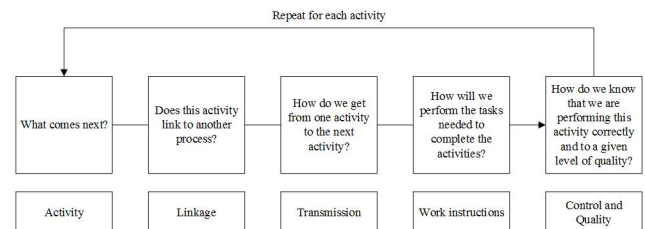


Figure 2. Fry's five key questions

B. Specify the ITSM Process Flow

When we have a diagram, provided by ITIL publications or as a result of Fry's process design methodology, we ought to document it. We can use a notation like BPMN [17], EPC or UML activity diagrams or another notation that allows the process to consist in "a number of tasks [activities] which need to be carried out and a set of conditions which determine the order of the tasks" [16] in order to represent such process. It should be noticed that when doing so, in the case of the ITIL processes present in the ITIL publications, we could go a little further and identify activities more granularly than those provided, through the reading of the process details. For instance,

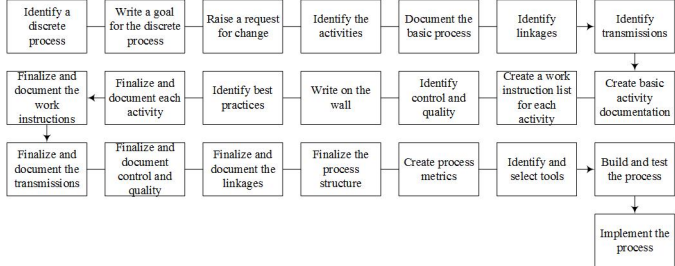


Figure 3. ITIL process to build an ITSM process

when assigning an incident a priority, to fetch the values of urgency and impact of an incident, so instead of only having an activity in the diagram named "Incident Prioritization" it can make sense to have another two "extra" activities named "Assess Impact" and "Assess Urgency". However, in this case, the person designing the process should carefully decide upon such cases it might make sense to leave the process "ITIL like" and increase the specification degree of activities at an operation centric phase (Section III.C).

C. Identify Operations, Inputs, Outputs and Data Types

In order to get an operation-centric view of ITSM activities within an ITSM process model, it is important to look at the activities present in the process diagram and think in which way can we subdivide them into logic operations, and how they are organized and ordered within a specific activity.

In order to derive operations, inputs, and outputs from the process diagram elements, it is important to take note of any artifacts that can be represented as variables. For instance, ITIL Incident Management and Problem Management processes have a section in each process that lists and covers almost every attribute for the main record types (Incident and Problem records). We should also analyze what needs to be inputted and outputted from the operations.

However not only the input/output data types and ITIL process records (such as Incidents, Problems, Changes, etc.) should be considered, some other data types might arise. For instance, during the reading and analysis of some ITIL process it is important to keep a record of actions made to that process (e.g. change of owner, categorization, etc.) and evince other data types that should be present (e.g. owner).

To help map and subdivide ITSM activities into operations, cross-functional diagrams should be developed. Such diagrams

link activities (or sub activities) to interface operations, their inputs and outputs, thus providing a global view of the relations between the elements that constitute an interface, which then can be used as a starting point for a revision and discussion that should take place with project stakeholders.

Figure 4 shows a partial example of a cross-functional diagram within the ITIL scope.

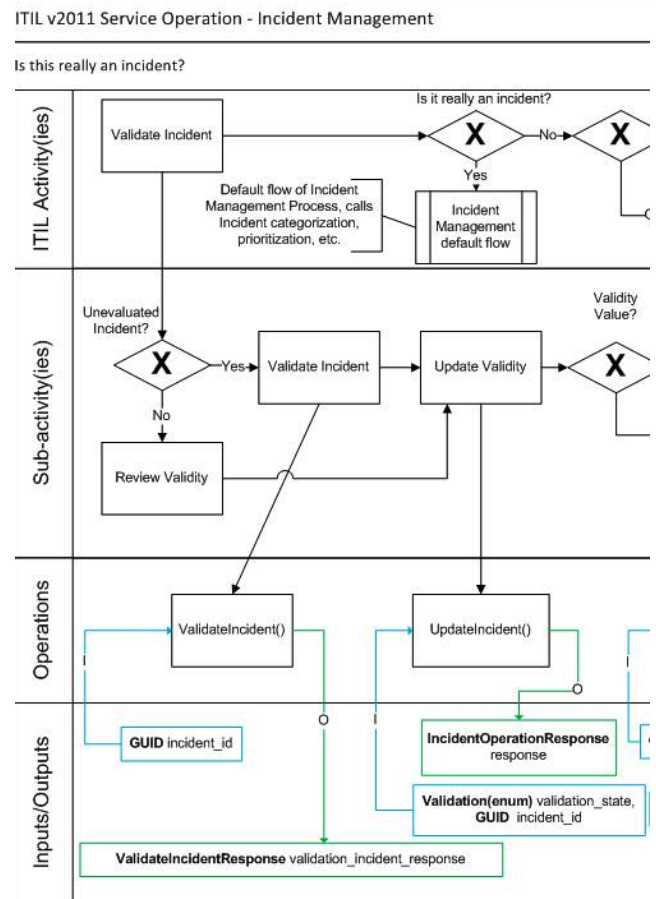


Figure 4. Portion of ITIL Incident Management process Cross Functional Diagram

Within SAPO, more specifically within the Service Delivery Broker (S.D.B.²) team, we identified a need to simultaneously define a generic ITIL conformant specification, covering the specific needs of each client (SAPO internal teams). The method found to deal with this need was to allow an extensibility model within the data types of each process.

Let us take a closer look at the ITIL Incident Management and Request Fulfillment processes as both have what is denominated Incident or Request Models (Incidents or Service Requests that are not new to the business, which occur frequently, and need to be dealt consistently in order to meet agreed Service Level Agreements [2]). Each model identified by a client has its own set of attributes. For instance, let us say that one of the request models is a model for a password reset request, which has an old password attribute, however this attribute is not needed in a model that deals with the register of

² <http://sdb.sapo.pt/en/index.html>

new users since a new user did not have a password before. Therefore a need for an extensibility model arises when designing ITSM interfaces. One solution to deal with this is to have an external specification (e.g., materialized in an XML³ or JSON⁴ file) related to each incident or service request where the user can define the “extra fields” needed for the different incidents or service requests types.

Some types of records in ITIL specifications have a solid base common to all of them. E.g., the ITIL Incident, the Service Request and the Request For Change all have, at least an ID, a summary, a categorization, an urgency, an impact, a priority and a description, so it makes sense to have a base record with all this information and make each of the ITIL specific data types inherit from such base record. Ultimately, the specification of such data types can be made via an ontology, e.g., with OWL (<http://www.w3.org/TR/owl2-overview/>).

D. Discuss and Revise the Specification

During the final period of this work, specifications for the Incident Management and Problem Management were developed (using the described methodology), discussed and reviewed with the SAPO S.D.B. Team. A specification was developed and it served as the basis for the Request Fulfillment internal implementation at the company. This implementation consists of an API to answer to the ITSM needs of the company, more specifically activities corresponding to the Request Fulfillment ITIL process, which will be used as part of S.D.B. service offering.

IV. EVALUATION

In addition to the reviews described in the previous section, the methodology was continually evaluated from a stakeholders and User eXperience (UX) point of view.

The methodology hereby presented was continuously evaluated against the requirements presented from all stakeholders, including the ITIL request models and specific requirements from each team within the company.

The user interfaces were also evaluated against requirements from the UX team within the company. Since the interfaces are going to be part of a solution (a web application), feedback from the UX team was needed, and every aspect that would structurally change or impact an interface, or its design, was discussed and, if needed, implemented.

V. CONCLUSIONS

We believe that this first scientific approach to ITSM interface design can take the development of ITSM interfaces (and the solutions they are a part of) to a whole new level. Not only from an organizational costs perspective, but we hope that this work can start a significant scientific discussion around ITSM interface design.

This work promotes the reuse of ITSM interfaces, contradicting the paradigm that each organizational entity should develop or buy its living ITSM solution silo. Reusable ITSM interfaces allow companies to improve and enhance the

functionality of their ITSM processes, since ITSM practices implementation can be shared with several consumers instead of one ITSM solution for each. Hence, the improvements identified for a party are identified to all improving the experience and quality of service of the ITSM solution overall.

Despite the fact that this work was, initially and in its essence, an exploratory academic project, it evolved to the point where this methodology is currently being used as the basis for the implementation of ITIL processes interfaces within the company we worked with, and will likely be used by a set of teams within the organization.

VI. ACKNOWLEDGEMENTS

We would like to thank to SAPO Labs⁵ and the CISUC IS Group⁶, whom supported this work, as well as the S.D.B. team that made each and every part of this work to be a reality.

REFERENCES

- [1] OGC - Office of Government Commerce, “The official introduction to the ITIL Service Lifecycle”, The Stationery Office, UK, 2007.
- [2] OGC - Office of Government Commerce, “ITIL Service Operation”, Stationery Office, UK, 2011.
- [3] R. Pereira and M. M. da Silva, “ITIL maturity model”, CISTI, pp. 1–6, 2010.
- [4] R. England, “Review of recent ITIL studies”, APM Group Ltd, 2011.
- [5] I. Sommerville, “Software engineering”, International Computer Science Series, Pearson Books, 2010.
- [6] G. T. Heineman and W. T. Councill, “Definition of a software component and its elements”, Chap. 1 in “Component-Based Software Engineering: Putting the Pieces Together”, Addison-Wesley, Boston, MA, 2001, pp. 5-20.
- [7] C. Szyperski, D. Gruntz and S. Murer, “Component software: beyond object oriented programming”, Component software series. ACM Press, 2002.
- [8] S. Dudek, F. Uebernickel and W. Brenner, “Explicating technological and organizational interfaces of modular IT service components to support the process of IT service composition.”, Proc. Intl. Workshop on Advances in IT Service Process Engineering, pp. 7–12, 2011.
- [9] M. Fry, “ITIL Lite - A road map to full or partial ITIL implementation”, The Stationery Office, UK, 2010.
- [10] M. Papazoglou and J. Yang, “Design methodology for web services and business processes”, Technologies for E-Services, LNCS, Springer Berlin Heidelberg, vol. 2444, pp. 175–233, 2002.
- [11] C. Videira A. Silva, “UML, Metodologias e Ferramentas CASE”, vol. 2, Edições Centro Atlântico, 2005
- [12] S. Kempter and A. Kempter, “ITIL implementation”, It Process Maps - <http://en.itprocessmaps.com>, 2008
- [13] J. Wang, “How to implement ITIL successfully?”, MSc Thesis, JÖNKÖPING INTERNATIONAL BUSINESS SCHOOL, 2010
- [14] J. Huang, “Should you be bi-lingual as an IT outsourcing service provider?”, eTOM and ITIL, BPTrends, January 2005
- [15] W. van der Aalst and K. M. van Hee, “Workflow Management: Models, Methods, and Systems”, MIT Press - Cooperative information systems, 2004
- [16] The Object Management Group, “Business Process Model and Notation”, <http://www.omg.org/spec/BPMN/2.0/PDF/>, 2011
- [17] D. de Champeaux and P. Faure, “Object-Oriented System Development”, 1993
- [18] M. Nüttgens, T. Feld, and V. Zimmermann. “Business Process Modeling with EPC and UML Transformation or Integration?.

³ <http://www.w3.org/XML/>

⁴ <http://www.json.org/>

⁵ <http://labs.sapo.pt/>

⁶ <https://www.cisuc.uc.pt/groups/show/is>

