Journal of Universal Computer Science, vol. 24, no. 7 (2018), 975-1011
submitted: 17/11/17, accepted: 10/7/18, appeared: 28/7/18 © J.UCS

Service-Driven Iterative Software Project Management with I-Tropos

Yves Wautelet
(KU Leuven, Brussels, Belgium
yves.wautelet@kuleuven.be)

Manuel Kolp
(Université catholique de Louvain, Louvain-la-Neuve, Belgium
manuel.kolp@uclouvain.be)

Loris Penserini
(TTP TECHNOLOGY srls, Pesaro-Urbino, Italy
elpense@gmail.com)

Abstract: The increased symbiotic relationships between society and Information and Communication Technology (ICT) pave the ways for a substantial alignment and rethinking of current software development methodologies. This paper presents the use and validation of a software analysis and project management (PM) framework for iterative software development within the Tropos method. This methodology is service-driven, its requirements models are founded on social-based modeling elements. The PM framework includes risk and quality management; it has been applied on multiple case studies and this paper presents a full experience report. The proposed methodology is aimed to provide a reference for practitioners willing to develop iteratively using Tropos.
Key Words: Service-oriented Development, Iterative Development, Software Life Cycle Management, Tropos, Unified Process, Project Management, Service Modeling
Category: D.2.1, D.2.9, H.1.0, J.4

1 Introduction

The intensive use of ICT in most of the crucial aspects of everyday life – i.e. software systems developed to support human activities and to cope with social problems – makes more evident to professionals that the software development has to deal with social aspects and human needs. Iterative development consists in the development of a series of releases having different scopes and increasing completeness. It allows to systematically collect users' opinions and desiderata on the software under development and target more complex and socio-technical systems. The iterative development is, consequently, being increasingly used by professionals notably through the application of the *Unified Process* (*UP*, see e.g. [IBM, 2007, Shuja and Krebs, 2007, Gibbs, 2006, Kruchten, 2003]) and the agile initiative. These types of processes provide benefits such as efficient Project

Management (PM), continuous organizational modeling and requirements acquisition, early implementation, continuous testing and modularity.

Classical software engineering methodologies usually support a waterfall *System Development Life Cycle* (*SDLC*) or advice their practitioners to proceed iteratively without truly offering a baseline to support that way of proceeding. Consequently, when applied, iterative development is simply performed in an "ad-hoc" manner (like for example in the eXtreme Programming's planning game) not suited for the development of huge user-intensive applications supporting complex business processes and organizations. This paper proposes a service-driven development process using coarse grained (i.e. high-level) and social-oriented requirements models, to drive the software development both in terms of PM and forward engineering (transformational) techniques. The process is called *Iterative-Tropos* (*I-Tropos*). The latter process was firstly detailed in the unpublished PhD thesis [Wautelet, 2008] in which it has been applied and validated on a production management system in the steel industry. The process skeleton (only) was presented in [Castro et al., 2009] with a set of other techniques to deal with complexity using the Tropos methodology [Castro et al., 2002, Mylopoulos et al., 2013, Castro et al., 2013, Yu et al., 2011, Bresciani et al., 2004, Penserini et al., 2007]. An extended version of steel industry case study results was published in [Wautelet and Kolp, 2011, Wautelet et al., 2011]. These results were preliminary and the process suffered of important software modeling issues in the context of large software developments. Indeed, the service layer was not considered so that the coarse grained representation (i.e. the top level view) was done using an i* Strategic Dependency diagram. This did not allow to easily represent several levels of granularity leading to poor requirements representation. These limitations are addressed using a service-engineering approach (see 4.2). The I-Tropos process as presented in this paper is by nature service driven; it is an evolution of the previous work validated with the *TransLogisTIC project* [Wautelet, 2012] which serves as a full experience report.

Figure 1 summarizes the position of I-Tropos and the PM framework with respect to the classical Tropos process. As evoked, the process focuses on using semantically rich conceptual models and elements (which are the "driving" models) independent (at analysis level) from specific development technologies such as object-oriented, agent-oriented, etc. The methodology aims to fill the iterative life cycle gap of Tropos through a phased template adapted from the one defined in the UP and a custom PM framework driven by high level entities (generalized into the name *services*[1]) allowing to model system requirements. The process can be said to be service-driven both in terms of PM and transformation (see Section 4). I-Tropos introduces the Strategic Services Model (SSM); the latter is

[1] Note that we do not necessarily target building a *Service-Oriented Architecture* (*SOA*), the service concept is, in this context, a structuring concept for an IT system offer of functionalities.

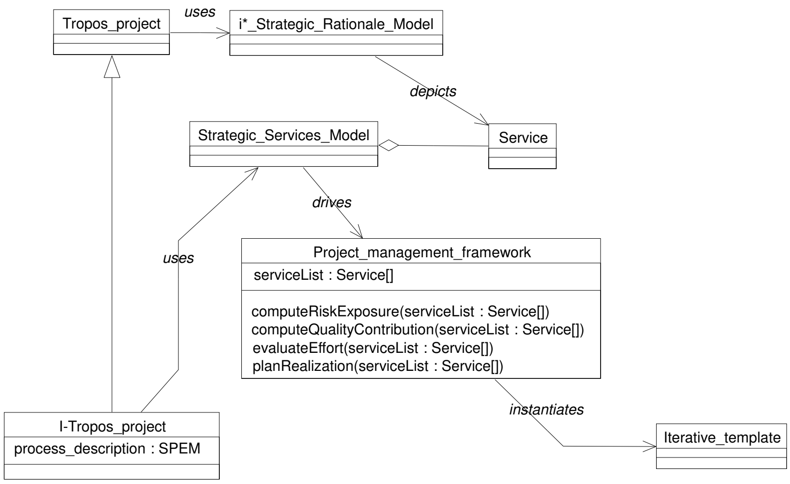

Figure 1: The Project Management Framework within Development Methods.

not used by the classical Tropos process. The SSM was introduced in [Wautelet and Kolp, 2016] that documents the I-Tropos transformation process, i.e. how to build a Multi-Agent System design and implementation out of a high level organizational analysis. The SSM allows to build a high level vision of the services to develop as well as to instantiate the PM framework. Further documentation of the services is made using the i* Strategic Rationale Model in the same fashion as in the classical Tropos process except that we develop one diagram per service in the system; this allows to better deal with scalability issues when developing large scaled applications in heterogeneous software ecosystems.

The PM framework (at least partially) covers risk, quality and time management aspects at the highest decision level, taking *Threats* and *Quality Expectations* (*QE*) evaluation directly into account for (iterative) development planning. The main purpose of the PM framework as presented here is thus to allow iterative development based on risk and quality evaluations. I-Tropos could be used in a pure waterfall fashion without the PM framework, we aim to isolate the PM aspects here to present them and validate their use.

The contribution of the paper is essentially methodological, i.e. a way to deal with an iterative life cycle in a (huge) software project requiring a high business and IT alignment (BITA). Indeed, taking back the organizational abilities of i*/Tropos (thus ensuring strong BITA), the scalability is improved with respect to Tropos by the service-based approach and the iterative life cycle. We point the PM framework as contribution but more particularly the way it is articulated. The values used to calibrate the framework must be seen as start values for a team willing to develop in I-Tropos and should be refined after each project in function of lessons learned and the teams specifics (see Section 9). Lots of parameters also need to be set-up by management and governance meetings in function of opinions and possibly measurements (this is discussed in Section 8).

Finally, the iterative template of the I-Tropos process can be compared to the one of the UP even if it is targeted to agent-based development. This is important in the sense that we do target an iterative development where most important development topics (i.e. services) for each iterations are set-up early on in the project and not on an iteration by iteration basis like it is in processes like SCRUM. Plan-driven methods like I-Tropos and RUP are adapted to projects with heavy processes where business continuity has to be ensured like in an industrial environment [del Nuevo et al., 2011]. It allows to combine the advantages of definite planning with the flexibility of introducing new requirements and aspects later on in the software development in an agile fashion.

We use the TransLogisTIC project as a real world running case study and as a full experience report. It concerns the development of a platform for e-collaboration between actors involved in *Outbound Logistics* (*OL*). OL is the part of the supply chain located downstream of the production line which is consequently focused on product delivery with a strong highlight on stocking and transportation. PM aspects of this software development project will be fully detailed in the paper.

The paper is structured as follows: Section 2 introduces the research method, Section 3 deals with the running case study, TransLogisTIC, a real world platform development project for OL that serves as an instantiation of our PM framework, Section 4 describes the model-driven PM approach we propose. Further refinements to i* models are presented and formalized in Section 5 while the I-Tropos' phases are developed in Section 6. Section 7 instantiates I-Tropos' PM framework to the TransLogisTIC project and discusses its results. Finally, Section 8 describes how and who is responsible for evaluating and deciding the value of the parameters required to set-up the PM framework of I-Tropos and Section 9 depicts the threats to validity. Section 10 depicts the related work and Section 11 concludes the paper and points to future work.

2 Research Method

This section deals with the research method by justifying how the whole software process - both in terms of SE and PM disciplines - has been build-up and validated. Basically, when compared to the Tropos process, there are two main add-ons within I-Tropos allowing monitored iterative development:

– **A generic SE process description** using semi-formal semantics where the process is divided into disciplines (partitioning activities under a common theme) and phases (which are groups of iterations with a major milestone) acting as a reference for practitioners;

- **A PM framework** allowing to plan the different project phases and iterations' content on the basis of threats and QE (those will be defined in Section 5).

To scope of this paper is the PM framework only.

2.1 Formalizing the SE Process (general context of the research)

The I-Tropos process has been build-up on the basis of previously validated methodologies that have been combined. Indeed, I-Tropos basically constitutes an enhanced version of the Tropos process where extensions are intended to furnish techniques and tools (driven by services) to deal with scalability and develop iteratively (see also Figure 1). It has thus been built on the basis of:

- the methodological foundations of the Tropos process allowing to define a series of disciplines wherein Tropos' activities are grouped. This represents the **vertical** dimension of the process;
- the iterative perspective of the UP. Indeed, the UP proposes a series of phases which are used and revised here within the context of Tropos. This represents the **horizontal** dimension of the process.

In order to express the process within commonly accepted semantics, we have used the *Software Process Engineering Meta-Model* (*SPEM*, [Anonymous, 2008]). Indeed, scientific literature on Tropos mostly focuses on describing the models to be produced but lacks to furnish a formal or semi-formal generic process specification on multiple levels. At the origin of I-Tropos, we made a systematic reading of [Castro et al., 2002, Bresciani et al., 2004, Penserini et al., 2007, Kolp et al., 2006] to produce a first version of the generic process. Then, this proposal has been revised based on 3 interviews with senior researchers having published specific contributions onto the Tropos process. Similarly, the architectural design and detailed design disciplines of I-Tropos have respectively been revised by 2 senior scientists having achieved subsequent contributions - within their PhD theses - at those specific levels within Tropos. The result was compiled into [Wautelet, 2008] and tailored to case studies for evaluation (see [Wautelet et al., 2011] for an evaluation of the first version of the process).

The process as described at that stage was nevertheless not suited for large software development because driven by goal models mixing various levels of abstraction leading to very complex structures that are hard to manage in broad development contexts. That is why this standard process has now been refined through the inclusion of a service-oriented view for driving the software project both in terms of project management (PM) and transformation. The refinement and the PM framework constitute the main contributions of the paper and is depicted in the following sections.

As evoked, each version of the generic process description is tailored to specific projects for validation so that further refinements have been reached through its application onto various case studies (see Section 2.2). Transformational techniques applied within our case studies (see section 4.1) are based on this generic description but since we only focus, in this paper, on the PM aspects so that we do not further document this (but it can be found in [Wautelet and Kolp, 2016]). The PM framework constituting the specific contribution of this paper and described in the coming sections could however be applied with other transformational techniques and would then require to be recalibrated (see section 2.2). After each case study has been performed we make a review of the successes and failures to improve the generic process description; the PM framework is exposed here in its current version and can be used as a standalone sub-process.

2.2 Project Management Framework

As evoked, the first step of the research consisted in formalizing and merging two validated software processes. However, a second add-on of I-Tropos – the one we focus on in this paper – is to furnish a PM framework to manage iterations content planning. This subsection discusses the methodological foundations used to build-it-up and calibrate it.

2.2.1 Towards an Adequate High Level View

As evoked, with respect to the previous versions of I-Tropos, our attempt to enhance the process for large scale development has lead to a major drawback: *finding adequate high level scope elements to "drive" the software project.* In other words, we need adequate building blocks to partition the software problem into manageable pieces that can be used as a "red thread" during the entire project. Those elements must together illustrate the software problem as a whole but without being overlapping. Moreover, they should allow dialogue with stakeholders so that they need to be located at a sufficiently high level of abstraction and allow (strategic) risk and quality analysis (see Section 4.2) in order to determine the element "criticability". Unfortunately, as such, the i* framework fails to furnish unambiguous scope elements, that is why we propose a meta-model with elements able to furnish an upper layer to i* models to enhance scalability through building blocks that can be used in an unambiguous manner; in I-Tropos these elements are called *services*.

2.2.2 PM Framework Calibration

The service elements evoked in the previous section are, in I-Tropos, prioritized for iterative planning. To this end, a method based on *threats* and *QE* (see Section 5 for definitions) is proposed. The method, however, requires validation but also calibration. The latter means that the method's figures need to be balanced or converted into industry-adopted measurements which requires estimations that are likely to evolve when researchers or "process engineers" gain experience. The PM framework was firstly calibrated on the basis of an ex-post quantitative and qualitative data analysis from 2 classical Tropos projects. Those projects where re-built with I-Tropos models and, based on the identified threats, QE and measured effort, default values were computed. Then, a preliminary project was conducted with the I-Tropos methodology, i.e. the development of a production system into the steel industry (see [Wautelet and Kolp, 2011, Wautelet et al., 2011]). This allowed to considerably refine the values precedently established. Finally, a second case study, the development of a collaborative platform for outbound logistics (OL) actors within the TransLogisTIC project, which is presented in this paper gave more accurate results with respect to planned and measured effort (see Section 7.4 for more information). The figures presented here thus represent a possible indication for development teams rather than a "scientific truth".

3 Case Study

We describe in this Section the TransLogisTIC project that will be used as a running case study through the paper. In 2008, the Belgian *Walloon Region* introduced the *Marshall Plan* (or more formally *Priority Actions for the future of Wallonia*), a vast economic investment aiming to inject funds in order to reinforce the attractiveness and competitiveness of Walloon's companies and raise employment rate. This plan notably integrates poles of competitiveness, and, among these poles, *Logistics in Wallonia* which is concerned with transportation and logistics. Into this latest pole, the TransLogisTIC project constitutes one of the projects conducted over the years. Within this project, we developed a collaborative platform for outbound logistics which is used as an experience report for the methodology proposed in this paper.

3.1 TransLogisTIC

Currently, the real-time visibility of information flows throughout the whole OL chain fails to ensure competitive integrated logistics. However, decisions at European level ask for the development of multimodal transportation, notably through Euro-corridors such as the "C" (Antwerp - Bale - Lyon) which would cross the Belgian Walloon area. For this reason, the TransLogisTIC project has been built up, with, as long-term main objective to develop combined, performing and complete transportations in Walloon Region with transport by rail particularly promoted in accordance with the European policy (see the Marco Polo program). The project has initially been planned on a 3 year basis with 14 million euro as overall budget and has involved several complementary actors including 10 private companies and 5 universities and research labs.

The analysis of the socio-economic context in which TransLogisTIC takes place has led to various concrete objectives, among which:

- The development of control systems tailored to the rail freight expectations;
- The development of real-time innovative localizing solutions covering the whole OL chain based on a new generation of positioning systems as well as on the optimal and extended use of existing infrastructures;
- The development of an online collaborative logistic platform allowing chargers, carriers, infrastructure managers and final clients (i.e. the major OL actors) to share information for a better optimization of the logistic chain.

The case study of the present paper **exclusively** focuses on this last point especially within the work carried out by the *Center for Supply Chain Management* (*CESCM*) at the *Louvain School of Management* (*LSM*) of the *Université catholique de Louvain* (*UCL*). This work concerns the development (i.e., analysis, design, implementation and test) of such a collaborative platform onto the Eclipse platform. As already said in the introduction section, the main contribution of this paper is the PM framework; therefore, we focus only on the software analysis and PM process and not on the design and implementation issues (which are covered in [Wautelet and Kolp, 2016]).

3.2 Collaborative Platform Development for Outbound Logistic Actors

This section briefly summarizes the OL processes. In order not to interfere with proprietary knowledge developed within the project, this view **is simplified**. Also, the way OL processes and user requirements information has been gathered is fully described in [Wautelet, 2012].
Outbound logistics is the process related to the movement and storage of products from the end of the production line to the end user. In the context of the developed platform we mostly focus on transportation. The actors of the supply chain play different roles in the outbound logistic flow. The producer will be a logistic client in its relationship with the supplier of raw materials, which will be considered as the shipper. The carrier will receive transportation orders from

the shipper and will deliver goods to the client, while relying on the infrastructure holder and manager. In its relation with the intermediary wholesaler, the producer will then play the role of the shipper and the wholesaler will be the client.

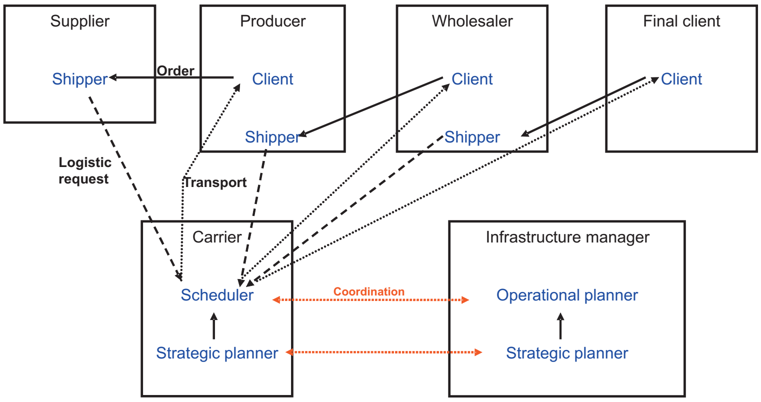

Figure 2: Material flows in the outbound logistics chain.

Figure 2 summarizes the material flows between the actors of the OL chain. The responsibilities corresponding to the different roles are:

– the *Shipper* receives an order from a client, does a logistic request to a carrier for the delivery of that order;

– the *Carrier* is refined into the following sub-actors:
  - the *strategic planner* decides on the transportation services that are offered on the long term, on the use of infrastructure, on the logistic resources to hold and on the client's acceptation conditions;
  - the *scheduler* orders transports to be realized, according to the strategic network and constraints, coordinates with the infrastructure manager and assigns logistic requests to those transports such that the delivery requirements are met;

– the *Infrastructure manager* holds the logistic infrastructure and coordinates with the carrier's scheduler to offer the network for the planned transports;

– the *Final client* books the transported merchandises.

The idea underlying the software development is to favour these actors' collaboration to obtain an optimized supply chain. Indeed, collaborative decision based on unified and centralized data will tend to avoid wastes in the global supply chain and enable to obtain the highest value that the chain can deliver at lowest cost (see [Pache and Spalanzani, 2007, Samii, 2004]). The collaborative application package to develop is thus composed of a multitude of aspects including the development of applications and databases to allow the effective collaboration and the connection to third party components providing well identified services.

4 Model-Driven Project Management

This section introduces the model-driven approach and introduces the opportunity of using services as scope elements for model-driven development with I-Tropos.

4.1 Background and Development Approach

Many modern software development methodologies are said to be *Model-Driven*. Model-driven engineering refers to the development process focusing on creating models - or abstractions - semantically closer to application domain concepts rather than computing (or algorithmic) ones. For instance, the UP is defined to be *use case driven* in the sense that the later engineering phases and disciplines are driven by the functional requirements and system behavior (encapsulated into the use cases) identified during early phases and disciplines (typically *Inception* and *Business Modeling/Requirements*). Moreover, effort estimation techniques such as *Use-Case Points* (cf. [Schneider and Winters, 1998]) directly take use cases and actors (which are high level model elements) as fundamentals for effort estimation. Within I-Tropos, we are also looking for an high-level model made of adequate scope elements to drive the whole software development. Worth noticing that for each iteration a higher/lower focus can be put on one or more of these elements in order to address the most critical parts of the project the earliest. More exhaustively, we are seeking elements allowing to divide the software problem into manageable pieces serving as red threads and scope for iterative planning and, finally, on the basis of which development effort can be estimated (the *services*, see Section 5.2).

Such entities, generally called *scope elements*, are part of a high-level requirements model of the methodology but can also be used as primary abstractions to drive the whole development process including PM disciplines in addition to just SE ones. In order to deploy this approach, these entities have to meet defined criteria, i.e. we need them to:

- be defined at a high abstraction level such as the organizational and strategic ones; typically a scope element should describe a conceptual process;

– be defined clearly enough to avoid multiple ways of representing a same reality (i.e. modeling the same with different elements of the model);

– not overlap with other scope elements (or make it explicit).

The first proposal of the paper is to map required elements characteristics into the i* framework [Yu et al., 2011, Dalpiaz et al., 2016] ones, a modeling framework proposing concepts such as *actor* (actors can be *agents*, *positions* or *roles*), as well as social dependencies among actors, including *goal*, *softgoal*, *task* and *resource* dependencies. This allows us to distinguish a set of entities (called *services*) suitable for managing SE activities through:

– **Transformation**: elements from one model are translated into elements of a different model at a more logical, architectural or platform-dependent level as details about the implementation are added[2]. Rich abstraction entities with their depending actors constitute a "high-level" platform-independent organizational analysis. Those models as a whole form a complete software requirement specification on which the design and implementation stages will be built/forward engineered whatever the underlying technology. Adding levels of refinement indeed allows to transform from a model to another, adding more details. Transformations are often applied to entire models, but can be applied to selections from models as well;

– **Process planning**: on the basis of a strategic risk and quality analysis, priorities among services are determined; the effort required for their realization is also estimated and this information feeds up (following a process depicted in the rest of this paper) an iterative development guidance.

Since the *service* elements are the building blocks of the highest level (i.e. including the elements of coarse grain only) model of the methodology and that they serve as scope elements both for *transformation* and *process planning*, I-Tropos can be said to be *Model-Driven.* This means that each lower level process element or each PM activity can be linked to a service element documented in the highest-level model (called the *Strategic Service Model*, see Section 5).

4.2 From i* Elements to Services

Following [Pastor et al., 2011], the i* framework can be evaluated on a series of nine features: *refinement*, *modularity*, *repeatability*, *complexity management*, *expressiveness*, *traceability*, *re-usability*, *scalability* and *domain applicability.* Those

[2] We again emphasize that the *transformation* within Tropos/I-Tropos is not the focus of this paper. We only present the transformation of a service to the i* *Strategic Rationale Model* and do not cover the design stage because we focus on the PM framework. The full transformation process is covered in [Wautelet and Kolp, 2016].

features are exhaustively assessed on the basis of a *not supported/not well supported/well supported* scale. Notably they highlighted what is clearly needed to extend the i* framework with mechanisms to manage granularity and refinement. Indeed, [Pastor et al., 2011] points out the lacks of mechanisms in i* for defining *granules* of information at different abstraction levels to structure, hierarchise or aggregate the semantics represented in the model. One of the flaws of i* is actually that all of the organizational modeling elements are represented on a unique abstraction level with poor hierarchy and composition/aggregation. Moreover, except for specifying abstract primitives as building blocks, analysts must be provided with guidelines to model a complete business setting through a set of organizational processes. These building entities could then be enriched into a set of more specific components that capture a certain organizational behavior. An high-level business view is then required such as the *Business Service Architecture* proposed in [Pastor et al., 2011] allowing software analysis focusing its activity on the *values* the Enterprise offers to the Customers. Those values are called services and used as basic granules of information that encapsulate i* process models. The services the enterprise offers are typically used as high-level scope elements while business processes fulfilling those services are then decomposed and refined. This approach allows us to combine the intentional and social characteristics of i* with the "traditional" business process modeling and determine the development strategy.

5 A Framework for Software Analysis and Process Planning

This Section depicts the building blocks of the software analysis framework used in I-Tropos including elements notably used as building blocks in the PM framework.

5.1 A Service-based Model

The meta-model presented in Figure 3 refines the proposition of [Pastor et al., 2011] by providing a view allowing to distinguish scope elements - the *services* - encapsulating sets of process elements. Services face *environmental factors*, i.e. *threats* and *quality expectations* (*QE*). The resulting diagram (model instance) will, in the development process, be used to define the development priority (in terms of PM, see section 7).

The left side of Figure 4 represents the services for the development of the collaborative platform for outbound logistics, their providing and consuming actors in the form of a dependency diagram. Services are represented as parallelograms, actors as circles. The model is inspired from i* in which a dependency describes an "agreement" (called *dependum*) between two actors: the *depender* and the *dependee*. The *depender* is the depending actor, and the *dependee*, the

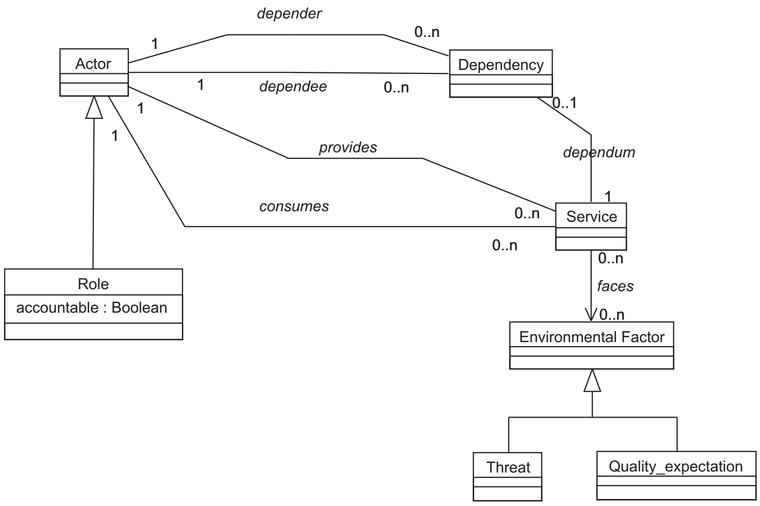

Figure 3: Scope Elements for Project Management with I-Tropos: A Meta-Model.

actor who is depended upon. The type of the dependency describes the nature of the agreement, in our case a *service*. Dependencies have the form *depender → dependum → dependee*. In our model, the *service consumer* (*SC*) is the depender and the *service provider* (*SP*) the dependee actor which means that the latter is the target of the dependency and the former its origin. The parallelogram represents the service as a black box encapsulating an i* *Strategic Rationale Diagram* (*SRD*) that can be of various complexity. An example is depicted in the next Section.

5.2 More Considerations about Services

Multiple definitions of the service concept can be found in literature with notably an impact onto the evaluation of their granularity. In the context of this paper, we rely to the conclusions of [Haesen et al., 2008] and define Services as "high-level" elements i.e. *coarse-grained granules of information that encapsulate an entire or a set of processes*. This view is in accordance with the one of [Ferrario et al., 2011, Ferrario and Guarino, 2008] which proposes an ontology for their proper representation. Within their conceptualization they notably distinguish the *Service Commitment* – prescriptive level – which is the level we refer to within the meta model presented in Figure 3 and the *Service Process* – design and implementation levels – referring to the operational level (useful in the transformation approach but not covered here). With respect to i*, operations on services are frequently implemented to encompass more functionality and operate on larger data sets when compared with traditional goal-based modeling

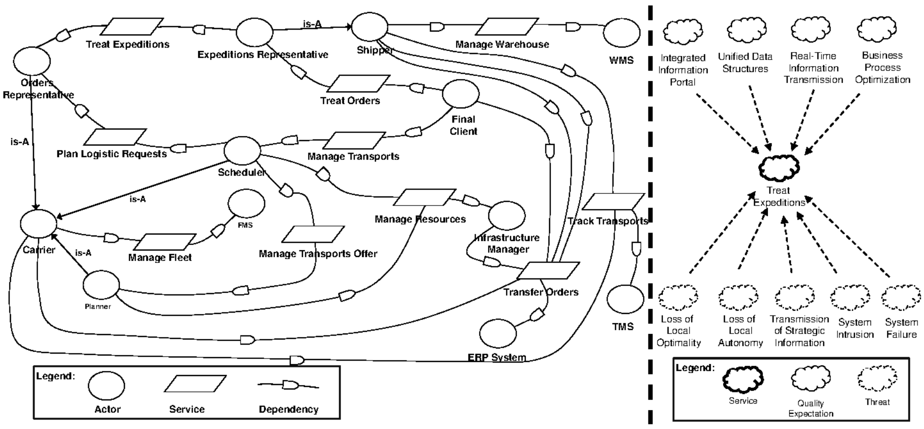

Figure 4: The Collaborative Application Package: Services as Scope Elements.

so that at Service Commitment level services can be understood as black boxes from which only their life cycle objectives are used. Note that, in our approach, services remain conceptual entities further described through a process perspective in an i* SRD (one per service). As already mentioned, the objective is not to make assumption on the implementation perspective, nor to build a SOA within the system design and implementation. The implementation of the i* models documenting a service is outside the scope of this paper; yet we based our approach on the building of a multi-agent system architecture in [Wautelet and Kolp, 2016].

$\langle s^t_j, pm_j \rangle$ is a service s_j, where s^t_j provides the details of the specification (in some language) of the service and pm_j is the process model of the service expressed here using the i* SRD.

We can take the example of the *Treat Expeditions* service. At service commitment (i.e. prescriptive) level, this service represents a value offered by the Carrier's `Order Representative` (depender) to the Shipper's `Expeditions Representative` (dependee). At service process level, this service includes all the elements required for settling a contract between a Shipper and a Carrier to transport goods, it is represented in an i* SRD (see Section 5.5).

Other approaches for dealing with modularity and manageability of i* models. [Franch, 2010] proposes to decompose i* models into SR modules; the suggested way of decomposing the problem is interesting but does lead to strongly coupled entities which is precisely what we want to avoid using a service-oriented approach. The services indeed are by nature loosely coupled and based on the values the enterprise offers.

5.3 Actors

Actors are intentional entities used to model people, organizations or software systems that are performers of some actions.

The *Role* concept inherits from *Actor* and, in the context of service modeling, it can consider *service provider* (*SP*) or *service consumer* (*SC*) as instances. The *Dependency* class materializes the dependency relationship between the service consumer and the service provider in the context of a service commitment (prescriptive level). The service consumer is the depender actor while the service provider is the dependee. More formally: An actor a_j can be defined as a tuple $\langle\{(rol_i, q_{rol_i}), \ldots, (rol_{i+m}, q_{rol_{i+m}})\}, Act\rangle_j$, where rol_i is a role the actor enacts to fulfill or consume services at quality level and cost q_{rol_i}. Played roles form a set AR of pairs, namely quality level and cost with values. Act is assumed to contain all additional properties of the actor necessary for its definition.

Format and content of Act will depend on the structure of the application domain. In practice, an actor usually plays several roles. In a more formal way we can say that: $\forall a_j \in A, \exists AR^{a_j} \subseteq R \times Q$ with $AR^{a_j} \neq \emptyset$ so that an actor role is defined as $a_k^{AR^{a_j}} \in AR^{a_j}$ and $a_k^{AR^{a_j}} = \langle a_j, rol_k, q_{rol_k}\rangle\ \forall k = i..m$ where A is the set of actors a_j and AR^{a_j} is the set of roles ($rol_k \in R$) that the actor a_j can enact along with their quality levels ($q_{rol_k} \in Q$).

With respect to the *Treat Expeditions* service, the `Carrier` plays the role of SP (because its `Order Representative` is providing it) and the `Shipper` plays the role of SC (because its `Expeditions Representative` is asking for it).

5.4 Environmental Factors

Services are subject to *Environmental Factors*: threats that can prevent taking benefits from the advantage/added value furnished by the service and QE can potentially reinforce competitive advantage of service adoption. Since the discussion takes place at the (coarse-grained) commitment level, elements are expressed in an aggregated manner in order to evaluate the alignment between the IT supported services and the organization's long terms positioning. A complementary ontological-basis (possibly in the form of various KPI, see [Horkoff et al., 2014]) can be used to qualify and quantify the threats and QE on a case by case basis. The purpose here is to express this in a generic way to illustrate how the software process is managed.

The right side of Figure 4 shows these Environmental Factors for the *Treat Expeditions* service of the collaborative platform to be developed. As mentioned in the legend, the upper side of the figure shows the QE. Indeed due to a lack of space we will, later in this paper, further document this service only.

The notions of threats and QE can be considered as a subgroup of the *situations* as defined in [Horkoff et al., 2014]. Situations encompass all the elements of a typical SWOT analysis while we do not consider the strengths and weaknesses of a sector because we focus on a particular product (the software to-be developed) and not an entire sector as a SWOT analysis does. The notion of *influence* defined in the same paper relates to what we describe in the threat and QE matrices (see section 7) because it studies the influence of the *situations* to product related elements (services in our case, goals in the case of [Horkoff et al., 2014]). Within the *influence*, [Horkoff et al., 2014] distinguish the logical and the probabilistic influences. The first one corresponds to our notion of weight and the second one corresponds our notion of impact (see sections 5.4.2 and 5.5).

5.4.1 Threats

A threat describes thus an event that can negatively affect the proper resolution of a service or that can be the result of the misuse of a service process both in terms of achievement and degradation of quality. The threat is expressed as an aggregate risk with a quantification of the negative impact and a likelihood of occurrence. The sum of this quantification for all the threats a particular service is exposed to is called the *Overall Risk Exposure* (see section 7). A threat is later refined into a series of goals and softgoals with respect to the transformation process (see Section 5.5).

5.4.2 Quality Expectations

QE must firstly be distinguished from traditional softgoals in the sense defined by [Liu and Yu, 2001, Castro et al., 2002, Chung et al., 2000, van Lamsweerde, 2009]. Indeed, following [Liu and Yu, 2001, Castro et al., 2002], "*a soft-goal is a condition or state of affairs in the world that the actor would like to achieve. But unlike a hard-goal, there are no clear-cut criteria for whether the condition is achieved, and it is up to the developer to judge whether a particular state of affairs in fact achieves sufficiently the stated soft-goal*" while [van Lamsweerde, 2009] highlights that "*soft-goals prescribe preferences among alternative system behaviours*". Since we address a higher-level business view of the system's services, we need an abstraction where we can specify the "quality" concerns of the organization in line with its long term strategy. In other words, QE are the stakeholders' desires to align the system-to-be with the competitive positioning defined for the long run. Softgoals refer to the actor-level concerns the system-to-be should have to cope with. In this sense, softgoals describe conditions, states or preferences of a system-to-be while QE are aimed to enhance the competitive advantage of the organization adopting the software system. Within our modeling approach, a QE is expressed and quantified in the form of a constraint/concern onto a particular service through a degree of excellence. The sum of this quantification for all the QE a particular service contributes to is called the *Overall*

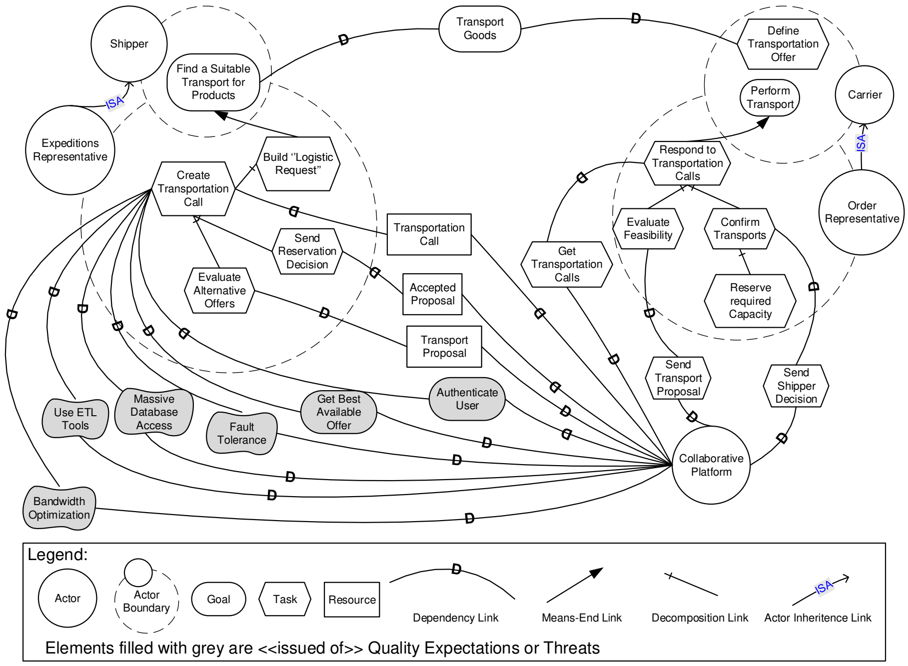

Figure 5: Strategic Rationale Diagram of the Treat Expeditions Service.

Quality Contribution (see section 7). A QE can be refined into a series of goals and softgoals in the transformation process (see Section 5.5).

5.5 Service Transformation

5.5.1 The Treat Expeditions Service as an i* Strategic Dependency Diagram

The transformation process starts from the commitment level where the software problem is represented into a set of services required by SC and provided by SP. The set of processes relating to the service fulfillment are depicted at process level in the form of i* goals, softgoals, tasks and resources. The SC and SP are forwarded into i* actors at process level.

The SRD for the *Treat Expeditions* service of our case study is depicted in Figure 5. The `Online Platform` is intended to support the `Expeditions Representative` (from the `Shipper`) and the `Orders Representative` (from the `Carrier`) in their activities related to the fulfillment of the service.

The `Shipper` actor wants to *Find a Suitable Transport for Products*, he is dependent of the `Carrier` actor to *Transport Goods* and the `Carrier` which *Defines*

Transportation Offer. To this end, the Shipper's `Expeditions Representative` actor achieves the Task *Create Transportation Call*. Shippers receiving orders from their clients will determine their day-to-day transportation needs. Those needs are defined as a *Logistic Request* which is a Resource required by the `Collaborative Platform`. Such requests identify the shipment, the origin and the destination, the requested pickup, delivery dates and other constraints specific to the shipment. Once the logistic requests have been identified for a scheduling horizon, shippers try to match those to a set of carriers, based on the process policies of the business unit. The Shipper's `Expeditions Representative` actor builds, for each request, one or several transportation calls, which are calls that will be addressed, using the `Collaborative Platform`, to the carriers to realizes a part of the journey of the shipment. A *Transport Proposal* resource is eventually addressed by a carrier; the *Evaluate Alternative Offers* and *Send Reservation Decision* tasks are aimed to treat these proposals and eventually transmit the *Accepted Proposal* to the `Collaborative Platform` in the form of a resource.

The `Carrier` actor is the one that performs the transport, this can be seen by the *Perform Transport* goal. To this end, the Carrier's `Order Representative` actor must execute the *Respond to Transportation Call* task. He then transmits its offer thanks to the `Collaborative Platform` actor through the *Send Transport Proposal* task. When the Shipper's `Expeditions Representative` actor has transmitted the *Accepted Proposal* resource, the `Collaborative Platform` actor decision is transmitted to the Carrier via the *Sends Shipper Decision* task.

5.5.2 Environmental Factors

QE and threats are then refined into a set of i* softgoals, goals and tasks at tactical level. Due to a lack of space and to keep the diagrams clear we only illustrate this process for one QE. The QE *Real Time Information Transmission* is supported by the SRD in the form of three i* softgoals, i.e. *Bandwidth Optimization*, *Fault Tolerance* and *Massive Database Access* but also on the i* task *Send Reservation Decision*. The QE *Unified Data Exchanges in Standardized Format* also supported by the SRD through the i* softgoal *Massive Database Access* (so same softgoals can refer to multiple QE) and another i* softgoal: *Use ETL Tools*. Let us also take the example of the Threat *System Intrusion* that notably refines in the SRD with the i* goal *Authenticate User*. Similarly, the Threat *Denial of Service* also refines into the i* softgoal *Fault Tolerance*. This transformation approach is partially illustrated in Figure 6.

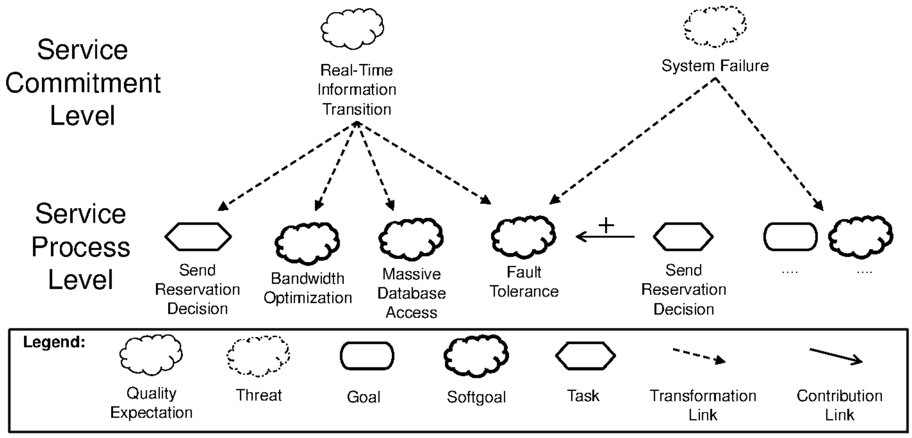

Figure 6: Transformation Process: Case Study.

6 Phases to be planned within I-Tropos

I-Tropos' phases are made upon a refinement of the UP phases as illustrated in Figure 7 (milestones expressed hereafter are based on metrics expressed in the UP, see [Kruchten, 2003]); each one is made of one or more iterations. Disciplines are gone through sequentially; as stressed before, the phases are separated by major milestones. Each of them has its own goal:

- The *Setting* phase is designed to identify and specify most of stakeholders' requirements, have a first approach of the environment scope, identify and evaluate project's threats and identify and evaluate QE;
- The *Blueprinting* phase is designed to produce a consistent architecture for the system on the basis of the identified requirements, eliminate most risky features in priority and evaluate first blueprints/prototypes to stakeholders;
- The *Building* phase is designed to build and evaluate each aspect of a working application and validate developments;
- The *Setuping* phase is designed to finalize production, train users and document the system.

Each phase is made of several iterations and it is up to the development team to determine the exact amount of iterations they want to include in the Blueprinting, Building phases and Setuping phases (typically the Setting phase is a unique iteration). Since the scope element is the service, the maximum iterations per phase would be the amount of services to be developed and this would the most align to the agile development fashion. It is then also possible to start a setuping phase for one or a few services while other services are still under the building phase in order to partially deploy the product. In the case study of this paper we have used 2 iterations in the Blueprinting phase. Also note that in function of the phase we are in, the effort spent on each discipline will of course be variable.

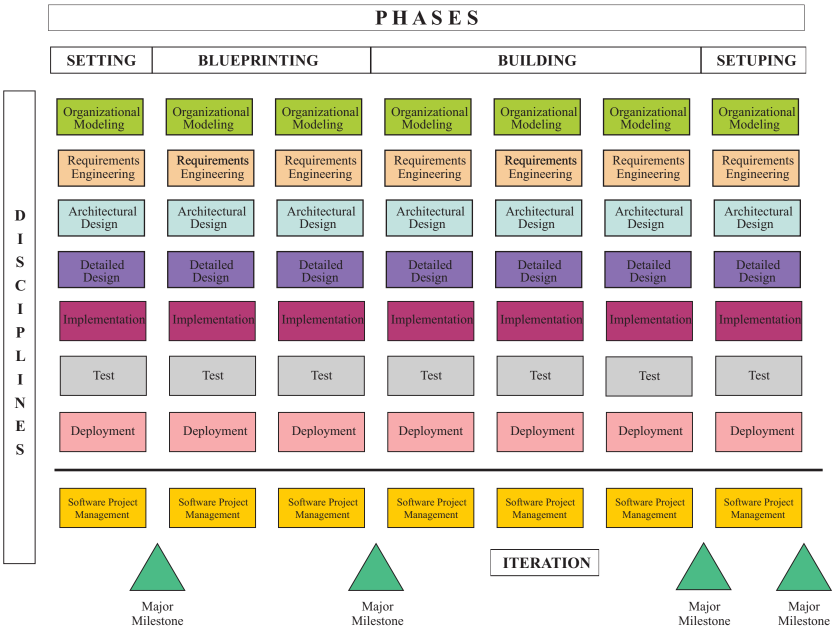

Figure 7: I-Tropos: Iterative Perspective.

7 Iterative Planning with I-Tropos

This section overviews the application of the I-Tropos process within its risk, quality and time management disciplines (in Figure 7, these disciplines are grouped into the Software Project Management box) for the TransLogisTIC project. The study presented in this paper reflects the work as it was performed during the setting phase, i.e. it is the managerial aspect of the iterative development life cycle leading to build a collaborative platform to support the outbound logistic actors' processes. Engineering disciplines within the software design and implementation are not covered to keep the focus on managerial aspects.

7.1 Project Management

The meta-model introduced in Figure 3 includes threat and QE elements identified, evaluated and balanced for the project planning. We do not represent all these elements for each service graphically as was done for the *Treat Expeditions* service in the right part of Figure 4 but list them and provide a matrix for threats/services as well as one for QE/services. This approach can be considered generic as it grounds on elements collected from every actor involved in the platform.

Each threat and QE is evaluated for each service on the basis of a *low*/*medium* /*high*/*none* scale. This allows to determine the overall risk exposure and overall quality contribution for each service. In other words, the service with the highest risk exposure introduces the highest "danger" to the project adequate realization so it should receive high priority. Similarly, the one with the highest overall quality contribution is the one adding highest value to the organization. A balance of these two elements allow to determine the "criticability" rank for each service (the highest, the most critical so the one that needs to be validated first). We basically define four values for the service's exposure both to threats and to QE as follows:

- **Low**: "L", in yellow, has a weight of 1;
- **Medium**: "M", in orange, has a weight of 2;
- **High**: "H", in red, has a weight of 4;
- **Non-existing**: an empty cell, this service is not concerned by this threat/QE, has a weight of 0.

The process of determining the threats and QE as well as assigning the values just mentioned is done by preparing and conducting a consultation process with stakeholders (see Section 8).

The *Overall Risk Exposure* of a service s is computed as follows:

$$OverallRiskExposure_s = \sum_i tw_i.re_i^s \qquad (1)$$

where tw_i is the weight of threat i and re_i^s is the exposure of the service s to the threat i. As seen hereover, re_i^s can thus take the values 0 (non-existing), 1 (low), 2 (medium) or 4 (high).

Similarly, the *Overall Quality Contribution* of a service s is computed as follows:

$$OverallQualityContribution_s = \sum_j qw_j.qe_j^s \qquad (2)$$

where qw_j is the weight of quality expectation j and qe_j^s is the exposure of the service s to the quality expectation j. As seen hereover, qe_j^s can thus take the values 0 (non-existing), 1 (low), 2 (medium) or 4 (high).

7.1.1 Risk Issues

The data produced by the stakeholder consultation along with the use of formula (1) can be organized in a threats/services matrix as presented in Figure 8 that allows rank services on the basis of their severity to risk exposure. The overall risk exposure of the *Manage Transports* service is, according to formula (1) computed as follows:

$$OverallRiskExposure_{ManageTransport} = 4.4 + 4.3 + 3.2 + 4.2 + 2.1 + 2.1 = 46$$

			SERVICES									
		Threat Weight (tw_i)	Treat Orders	Treat Expeditions	Plan Logistic Requests	Manage Transports Offer	Manage Transports	Manage Resources	Manage Warehouse	Track Transports	Manage Fleet	Transfer Orders
THREATS	Loss of Local Optimality	4		L		H	H	H	H	H	M	L
	Loss of Local Autonomy	3		L		H	H	H	M	H	H	L
	Transmission of Strategic Information	3	M	H	H	H	M	L	M	M	L	L
	System Intrusion	2	M	M	M	M	H	M	M	M	L	M
	Data Loss	2		L			L	L			L	L
	System Failure	2	L	L	L		L	M	M	M	M	
OverallRiskExposure			12	27	18	44	46	41	36	42	31	16

Figure 8: The Risk Exposure per Service Matrix.

7.1.2 Quality Issues

As above, the data gathered by the stakeholder consultation along with the use of formula (2) can be organized in a quality expectations/services matrix as presented in Figure 9 that allows to rank services on the basis of their values of *OverallQualityContribution*. The *Manage Transports* and *Manage Resources* services have an added value of 60, which are computed using formula (2).

QUALITY EXPECTATIONS	Quality Expectation Weight (qw$_i$)	Treat Orders	Treat Expeditions	Plan Logistic Requests	Manage Transports Offer	Manage Transports	Manage Resources	Manage Warehouse	Track Transports	Manage Fleet	Transfer Orders
		SERVICES									
Real-Time Information Transmission	5	H	M	H		H	H	M	H	M	M
Business Process Optimization	4	H	H	H	L	H	H	H	L		
Symetric Information	3	M	M	M	M	H	H				
Unified Data Structures	2	L	H	H	M	H	H	M	M	M	H
Integrated Information Portal	1	H	H	H	H	H	H	M	M	M	M
OverallQualityContribution		48	44	54	18	60	60	32	30	16	20

Figure 9: The Qualitative Contribution per Service Matrix.

7.2 Planning Realization

In the previous sections, we studied the threats and the QE' impact on the identified services. This section will use the *overall risk exposure* and *overall quality contribution* to determine a service priority. Indeed, facing an iterative life cycle we have to plan the technical realization of the services within the IT infrastructure. Services with the highest priority will firstly be designed, prototyped and tested during the Blueprinting phase and firstly be fully implemented during the Building phase. This allows exploring the trickiest issues first so that risks are taken earlier onto the project when corrective actions are the easiest to put into practice.

Computations are summarized in Figure 10 with respect to the following method:

- The $OverallRiskExposure_s$ is the exposure of a service s to risk;
- The $TotalRiskExposure$ is the sum of the $OverallRiskExposure$ of all the project's services;
- On the basis of the $OverallRiskExposure_s$, the $RelativeRiskExposure_s$ of a service s is computed using the formula: $RelativeRiskExposure_s = \frac{OverallRiskExposure_s}{TotalRiskExposure}$;
- The $OverallQualityContribution_s$ is the contribution of a service s to quality expectations;
- The *Total Quality Contribution* is the sum of the *Overall Quality Contribution* of all the project's services;

– On the basis of the $OverallQualityContribution_s$, the $RelativeQualityContribution_s$ is computed using the formula: $RelativeQualityContribution_s = \frac{OverallQualityContribution_s}{TotalQualityContribution}$;

– The *Priority Level* is computed by "balancing" the *Relative Risk Exposure* and the *Relative Quality Contribution* as follows: $PriorityLevel_s = k_1.RelativeRiskExposure_s + k_2.RelativeQualityContribution_s$ where k_1 and k_2 are weights to assign a different degree of importance. Specifically, within our case study, we use a repartition key of 75 percent for the relative risk exposure and 25 for the relative quality contribution (in other words, the risk component is set to a higher degree of importance). This repartition key is an approximation made on the basis of the previous projects[3] and discussed with stakeholders. These values can vary from project to project with respect to priorities and risks that the organization judges acceptable (see also Section 9 for more details);

– The initial list of services is ranked according to values of services' *Priority Level*, obtaining a new ordered list beginning with the service with the highest $PriorityLevel$. The numbers associated to services' positions in the new ordered list correspond to the column $ServicePriority$ of the table in Figure 10.

With respect to the priority list, the precedence constraints and the estimated service complexity, we determine the iteration plan in Figure 11.

7.3 Effort Estimation

Service complexity is an estimation of the amount of effort required for its full implementation. The I-Tropos methodology uses three categories of service complexity, i.e., Low/Medium/High, respectively with a value of 5/10/15. Those values are attributed to each service based on the study of the i* elements it regroups. In simple words, it is based on the complexity of the service process-level realization. Transforming the overall weight to person/months requires calibration on the basis of regression model using a large number of projects and remains an open issue. The proposed planning will be subject to modifications/reviews into the software project because of the process' iterative nature.

On the basis of the method depicted in Section 2.2.2 using the methodology overviewed in this paper, we have set the complexity of a service (*Service Complexity Weight* in Figure 10) to 0,8 person/months per unit. So that:

[3] The evaluation was based on an ex-post analysis of the elements responsible for delay in previous projects. We indeed compared the ex-ante evaluation with measured delays and made the repartition key vary until finding the best fitting compromise. We nevertheless do not consider this as a "scientific truth" but rather as a first basis to be discussed by the development team with project stakeholders on a case by case basis.

	Service	Overall Risk Exposure	Relative Risk Exposure	Overall Quality Contribution	Relative Quality Contribution	Priority Level	Service Priority	Service Complexity	Service Complexity Weight	Precedence
1	Treat Orders	12	0,038338658	48	0,12565445	0,060167606	9	L	5	S10
2	Treat Expeditions	27	0,086261981	44	0,115183246	0,093492297	6	L	5	
3	Plan Logistic Requests	18	0,057507987	54	0,141361257	0,078471305	8	M	10	
4	Manage Transports Offer	44	0,14057508	18	0,047120419	0,117211415	4	M	10	S5
5	Manage Transports	46	0,146964856	60	0,157068063	0,149490658	1	H	15	
6	Manage Resources	41	0,130990415	60	0,157068063	0,137509827	2	H	15	
7	Manage Warehouse	36	0,115015974	32	0,083769634	0,107204389	5	L	5	
8	Track Transports	42	0,134185304	30	0,078534031	0,120272485	3	H	15	
9	Manage Fleet	31	0,099041534	16	0,041884817	0,084752354	7	L	5	
10	Transfer Orders	16	0,051118211	20	0,052356021	0,051427663	10	L	5	
	Sum	313	1	382	1	1			90	

Figure 10: Service Prioritization.

	Service	Priority	Complexity	Blueprinting 1	Blueprinting 2	Building 1	Building 2	Building 3
1	Manage Transports	1	H	X		X		
2	Manage Resources	2	H	X		X		
3	Track Transports	3	H	X			X	
4	Manage Transports Offer	4	M		X		X	
5	Manage Warehouse	5	L		X		X	
6	Treat Expeditions	6	L		X			X
7	Manage Fleet	7	L		X			X
8	Plan Logistic Requests	8	M		X			X
9	Transfer Orders	10	L		X			X
10	Treat Orders	9	L		X			X
	Total Effort Weight		90	15	15	20	20	20

Figure 11: Iteration Plan.

- The Blueprinting 1 iteration should take approximatively 12 person/months;
- The Blueprinting 2 iteration should take approximatively 12 person/months;
- The Building 1 iteration should take approximatively 16 person/months;
- The Building 2 iteration should take approximatively 16 person/months;
- The Building 3 iteration should take approximatively 16 person/months;
- The total person/months of the project should be approximatively 72.

Note that the estimation has been done during the Setting phase; that phase has thus not been estimated since the estimation concerns activities of later phases and, at the time of the estimation, the cost and effort of the Setting phase were already known. The Setuping phase has also not been included in the estimation since that phase introduces serveral parameters that are external to the UCL's development team within our particular TransLogisTIC project. This situation is evoked into the next section.

The presented estimation needs further refinements and calibration but these conversion figures are taken as a first estimation basis; their performance is discussed in the next section.

7.4 Empirical Results

This section discusses some ex-post results and measures of the I-Tropos application to the TransLogisTIC case study.

7.4.1 Functional and Non-Functional Aspects

Inherently to the nature of iterative development, stakeholders have been involved during the whole development project. Major adjustments have been made on the basis of the prototypes presented to the involved actors which also implied delay with respect to the initial planning. This allowed us to evaluate the benefits of our approach on a daily basis even if this is not formalized here because we want to keep the focus on the PM framework.

The developed collaborative platform has later been reviewed using test scenarios developed on the basis of the project specifications but also further refined during the project and validated by partners. On the basis of the test set, a proof of concept/demonstration has been performed for the experts of the Waloon region and involved organizations (partners at technological and implementation levels) that sponsored the project showing that functional and non-functional specifications had been met. This shows that I-Tropos as a SE methodology was able to build upon Tropos and give accurate results. We do not enter into more details of the functional and non-functional offer of the platform at this level since the focus of this paper is to illustrate the iterative and PM aspects of I-Tropos. Further information about functional aspects of the e-collaborative platform are overviewed in [Wautelet, 2012].

7.4.2 Planning Accuracy and Measured Effort

Table 1 summarizes the results of the effort spent on each iteration[4]. The project involved – for the development of the collaborative platform – three full time employees during 30 months; therefore, the development took – from the Setting to the end of the Building phase (so Setuping is not included, see further in this section) – about 90 person/months. The Setting phase took approximately 3 months which is 9 person/months in total. Since the estimation takes place within the Setting phase and does only concern the coming phases[5], we have to substract these 9 person/months from the total of 90 to get the effort for the Blueprinting and Building phases only. We thus have 81 person/months for the Blueprinting and Building phases that must be compared to the estimation of 72 (so we had underestimated it by 9 man/month which is approximately 12,5%). We, of course, need to analyze these results with further details.

First of all, time was lost during the first iterations of Blueprinting and Building. This is not a hazard since most tricky issues were tackled during these iterations. However, one could wonder why we have spent so much time on Blueprinting1 and Building1. The 3 services tackled during Blueprinting1 (see Figure 11) induced to build up a common model for the whole logistic network; as different partners had different definitions of similar concepts and different views on the scope of such a model. Consequently, the modeling process required a lot of effort, e.g. many meetings with more actors than initially planned; this partly explains the underestimation. During the Building1 iteration, issues with one specific partner on the scope of each ones' part of the project led to significant delays.

The lesson learned from this project and the previous one is that progressing iteratively leads to sometimes unexpected modifications. This, of course, has a cost in terms of development time so that we envisage to further refine the effort estimation model to notably burden estimations with a factor linked to the complexity of interfacing of the model actors (using a GUI, EDI, etc.). The process thus needs to evolve thanks to the experience gained by researchers and process engineers with new projects.

Also, no figure is available for the Setuping phase because it has not been achieved by the same development team. Indeed, even if the Building phase produced a candidate release of the collaborative platform software package, the industrial partner who was in charge of performing the Setuping phase (the

[4] To be able to reason on the figures without introducing too many details we decided to keep the granularity at the month level. When the iteration ended before the 15th of the month we considered it finished the previous month and when it ended after we considered it latest until the end of the same month.

[5] Indeed, the estimation is based on a first software analysis so that it requires to perform modeling activities before being able to actually perform it. In I-Tropos, iterative planning and software estimation thus happen during the Setting phase.

Nb.	Phase	Person/Months Planned	Person/Months Measured	Δ
1	Setting	N/A	9	N/A
2	Blueprinting	24	30	+25%
2.1	Blueprinting1	12	18	+50%
2.2	Blueprinting2	12	12	0%
3	Building	48	51	+6,25%
3.1	Building1	16	19	+18,75%
3.2	Building2	16	16	0%
3.3	Building3	16	16	0%
2+3	$\sum$ Planned	72	81	+12,5%

Table 1: Iterations: Planned VS Measured.

same which induced delay in Building1) faced important financial problems. Another industrial partner was charged to continue developments but juridical issues slowed down the realization. Since the research team had delivered the expected results and financing was over, it was only marginally involved in the further developments. These considerations are nevertheless outside the scope of the content of this paper.

8 On the Evaluation and Decision of the Project Management Parameters

There is no unique pattern for conducting the process of evaluating and deciding about the *scope of the services*, the *threats*, the *QE*, the *weight of the threats and QE*, the *impact of the threats and QE onto particular services* as well as the *repartition key between threats and QE for iterative planning*. Because of the strategic nature (i.e. the long term impact of the choices made for these elements on the organizations position) of services, threats and QE, these elements should nevertheless be validated through a governance process supported by opinions and possibly measurements of the management-level. Services are indeed coarse grained elements having to be integrated in the organizations software ecosystem. Their development and deployment involve a huge amount of resources with a direct impact on the organizations future competitive position. This impact is, at least partially, identified through their relation with threats and QE. Because of this structural impact, the governance-level needs to be involved.

More specifically, we can here distinguish two types of choices that need to be made. The first one are structural IT choices and concern the services and their scope as well as the threats, the QE and their weight; these choices impact the organization long-term competitive position and are thus validated by the a governance board understanding the long-term strategy. The second one are project delivery related parameters and concern the impact of the threats and QE on the services and the balance between threats and QE but also the number of iterations to achieve; these choices impact the way that the project will be made by the project managers that understand the project specifics.

The general process to achieve decisions on these elements and parameters (in line with the procedure followed in the context of TranslogisTIC) is described here. Typically, a first set of discussions and interviews are conducted by the software modeling team onto the entire organization with representatives of all of the decisions levels (top managers, middle managers and operators). The primary aim is, of course, to understand the specific IT problem but, equally important, to understand the business environment in which the to-be system will be integrated. From this, the services can be scoped and modeled on the one side, a first list of threats and one of QE on the other side can be set-up. All of these elements first need to be discussed on the management-level meaning that the scoping, relevance and importance of the elements is discussed with managers and operators. A proposal with qualitative and possibly quantitative justifications is then presented to a governance board, in charge of reevaluating, modifying and validating the proposal. Specifically, within a formal governance board meeting, the evoked elements and values are discussed, new elements can be added and existing can be removed, values can be changed. Even if the discussions for evaluation should be made involving representatives of all managerial levels, the final decision should be made by a restricted IT governance committee (see [Weill and Ross, 2004] for the organization of such committees with respect to decisions on software applications).

On the other side, the values tracing the impact of threats and QE on services is the responsibility of the management team (generally under the validation of the CIO but not the entire governance board). The impact of the service is evaluated by business analysts, software designers and developers in function of available technical means. Several techniques can be set up and possibly crossed to come to an evaluation. We can take the example of the threat *Loss of Local Optimality* and the service *Track Transports* where the impact has been evaluated to *High*. Within the latter service, the Shipper can be able to see in real time where the transports of specific goods is located. With this information the Shipper can negotiate or impose some routes to the during the commercial transaction. This could lead to new constraints when managing the transports for the Carrier leading to substantial financial losses (typically a Carrier builds local optimums to minimize the transport cost with respect to the transportations he has to achieve for a time period). That is why this threat is evaluated with respect to High for the integration of the platform by all of the actors.

For some impact evaluation metrics can be used and estimated. Overall in the I-Tropos process - and in line with the agile principles – we always favor simple and fast estimations rather than heavy and complex measurements that would hamper the agility of the overall project.

9 Threats to the Validity

The project management framework as presented in this paper and pointed out as its main contribution is tailored to the use of I-Tropos. As such, its use with different transformation techniques (classical UML at design stage, other development languages) would induce to recalibrate the process leading to the need for a few test cases. A first threat to the validity of the PM framework is thus the generalization of its use outside the I-Tropos design and implementation stages as defined in [Wautelet and Kolp, 2016].

Similarly, another threat to the validity of the research is the accuracy of the model within other software development teams. The PM framework has indeed been applied on various real life projects but people involved in the modeling and management of these have always been the same so that their knowledge and experience of the whole I-Tropos process can have an influence on the results.

Finally, when computing the *Priority Level* for each service, the balance between k_1 and k_2 can modify their final rank; figures given in this paper only provide a guidance based on previous projects but precise values for each project are impossible to determine since there is no "one size fits all" solution. Generally speaking, the less risk the organization is willing to take with the system adoption, the highest k_1 must be (stronger focus on the risk component) and the highest the added value the organization is willing to have with the system adoption, the highest k_2 must be (stronger focus on the opportunity component).

10 Related Work

As evoked, the idea of developing iteratively is not new; it was already highlighted in the 80's by Barry Boehm (see [Boehm, 1988,Ruparelia, 2010]). The aim is not to make an inventory of iterative methods but to overview methods for planning and estimating iterations on the basis of analysis models (at best using social driven templates). Since I-Tropos is an agent-oriented development technique, we mostly focus on the methods for this software development paradigm and their relation to/definition of PM frameworks.

First of all, we have to mention the RUP/UML methodology [Shuja and Krebs, 2007,Anwar, 2014] which remains widely used nowadays in industry. With respect to the conditions evoked in this paper, the model for driving iterative development is the use-case one. No real structured method has been found in literature allowing to do so; only a few ad-hoc approaches have been found or non-detailed ways of proceeding (e.g. [Jalote, 2002, West, 2003, Sägesser et al., 2013]). The PM framework presented in this paper is nevertheless adaptable to the RUP context with use cases as scope elements instead of services but alignment of the modeling practices need to be considered and calibration is required.

Next to object-oriented development methods, numerous *Agent-Oriented Software Engineering* (*AOSE*) methodologies have been proposed in the past twenty years [Leitão and Karnouskos, 2015, Sturm and Shehory, 2014] and need to be updated and unified to nowadays software development standards [Dam and Winikoff, 2013] could possibly be used in an iterative perspective. Among them Tropos offers the most advanced social-based modeling features, that is why it has been chosen in the context of the present research. Gaia [Cernuzzi et al., 2014, Wooldridge et al., 2000], one of the most popular methodologies due to its simple and clear process as well as its neutrality with respect to implementation techniques or platforms has been used with an iterative SDLC [Sivakumar et al., 2013]. The main drawback of their proposal is that they decompose functionality on the basis of design models (called "parts") and not analysis ones so that it is impossible to plan on the basis of analysis elements within it. Moreover, the priority given to those functional parts is done in an *ad hoc* manner since no framework is given to evaluate the criticality of these "functional parts". There is also no real template provided for cutting out the project into phases so that each iteration can be considered as a "sub-waterfall project" with no incremental prototyping for testing as implied by the cut out between blueprinting and building phases in our framework. ADELFE [Mefteh et al., 2015, Bernon et al., 2002] claims to follow the RUP and uses use cases to depict (late) requirements so that at analysis stage the method is equivalent to traditional RUP/UML and the conclusions for the latter method are identical.

An iterative approach to goal-oriented knowledge acquisition is developed in [Pourshahid et al., 2014]. They start with identifying the main building blocks of the company strategy (called dimensions) that are later refined into the next iterations and associated to Key Performance Indicators (KPI). Their approach goes in many ways further than ours notably in the sense that managers can monitor the impact of decisions on the organization's goals and improve both decision models and business processes. Nevertheless, their method is applied on the entire company for global investment decision rather than at the level of a particular software development like we do. Also, the dimensions relate to a classical goal model not incorporating the benefits of a service-oriented approach as advocated in this paper.

Different sources highlight the importance of risk management at the center of the requirements engineering process (e.g. [Asnar et al., 2011, Lund et al., 2011, Cailliau and van Lamsweerde, 2013]). [Islam et al., 2014] suggests to incorporate risk management in the development of goal-oriented software and more particularly in the KAOS framework. Their approach is close to our approach with threats (that they call risk factors) seen as obstacles to the realization of goals; these obstacles are realized through risk events and goals can then be introduced to prevent these risk events. Their approach to risk management is very detailed and formal but exclusively oriented to operational risk management. We align to their transformation approach in the sense that we also suggest to transform threats into goals and softgoals but also manage risk through a higher (strategic) level perspective thanks to the service-oriented approach.

Agile methods such as *eXtreme Programming* (*XP*) [Vlaanderen et al., 2011, Lucia and Qusef, 2010], use *user stories* [Wautelet et al., 2014] to depict functionalities of the system-to-be; those are prioritized (and sorted) within the *planning game* [Beck and Fowler, 2000]. User stories are natural language sentences (informal templates are available) depicting a functionality the user is expecting; as pointed out in [Wautelet et al., 2014] no or poor formality is given for their definition and do not allow to represent the software problem in a coarse-grained (user stories are most often expressed as atomic functions) or social oriented manner. Even if the idea of prioritizing is present, the planning (and sorting) of user stories is realized in an ad-hoc manner. These methods are poorly scalable probably because of the operational perspective of US so that no formal PM framework can be built to sustain iterative planning.

Finally, [Wautelet et al., 2017] propose a process plug-in to integrate agent-based development in SCRUM. The process plug-in nevertheless focuses on the transformation approach and no framework is proposed to manage the iterations; in other words, the method is not plan driven and prioritizes user stories on the basis of business value after each sprint. The process plug-in nevertheless is based on a requirements model called the rationale tree that includes business and social behavior; goals could be used as scope elements in the PM framework depicted in this paper but we could then not deal with scalability and only tackle problems of limited size.

11 Conclusion

Developing a high level model based on services allows to scale the software system to-be around the long term values of the company developing the software product. These coarse grained building blocks encapsulate an entire business process thus process elements can be further studied within finer grained models. Moreover, service elements can also be used for studying the impact of environmental elements on parts of the system to-be: threats and QE can be evaluated for their impact/influence on the services being developed allowing to determine how critical the service development can or will be. That is why we have incorporated a service view within the I-Tropos process allowing to deal with the strategic impact of the system-to-be developed and deployed onto the organization adopting it. Service elements evaluation is notably used in the perspective of iterative development.

Information systems' iterative development has been widely used for the last ten years, notably through the RUP and agile methods. Iterative development is a building process used to deliver the functionality of a system in a series of releases of increasing completeness. Each release is developed in a specific, fixed time period called an iteration. It has become a best practice for building large-scale user-intensive systems due to its potential benefits mainly resolving major risks before making large investments, enabling early user feedback, testing and integrating modules and components in a continuous way, and focusing on project short-term objective milestones. However, in terms of PM, despite the plethora of frameworks that exist for the traditional (waterfall or sequential) system development, there is a lack of established frameworks for iterative planning and effort estimation. The paper has proposed a social-driven framework to cope with the above limitations, applied and validated it on a case study as well as presented the experience gained.

This paper indeed proposes to use a building process driven by service elements used in an i* fashion to drive the iterative software development in terms of PM. This PM framework covers risk, quality and time issues at the highest decision level taking threats and QE's evaluation directly into account for iterative development planning.

With respect to the Tropos, the current (upgraded) version of the I-Tropos process, this new increment allows to:

- deliver increasingly the functionality and quality of a system in a series of incremental releases;
- reduce the risk profile with continuous integration throughout the project;
- manage functional and non-functionals requirements using a service approach;
- allow managing of time, change, quality and risk issues.

The process remains nevertheless not a "one size fits all" approach in the sense that different development teams with various experience and using a different design and implementation approaches need to calibrate the process' figures to their own characteristics. The process will thus continuously gain maturity with its use. Also, it has been built and calibrated without considering code reuse or fitting to specific multi-agent architectural styles and design patterns [Kolp et al., 2011, Kolp et al., 2008]. Including such styles and patterns would have an influence on the effort required to implement the MAS and thus require new calibration. Finally, let us highlight that a CASE-Tool called DesCartes Architect [Anonymous, 2018] incorporates a module to support the PM framework presented in this paper in its classical version; the incorporation in the cloud version of the same tool is under development.

References

[Anonymous, 2008] Anonymous (2008). Software & systems process engineering metamodel specification. version 2.0. Technical report, Object Management Group.

[Anonymous, 2018] Anonymous (2018). Descartes architect: Design case tool for agent-oriented repositories, techniques, environments and systems. `http://www.isys.ucl.ac.be/descartes/`.

[Anwar, 2014] Anwar, A. (2014). A review of rup (rational unified process). *International Journal of Software Engineering (IJSE)*, 5(2):12–19.

[Asnar et al., 2011] Asnar, Y., Giorgini, P., and Mylopoulos, J. (2011). Goal-driven risk assessment in requirements engineering. *Requir. Eng.*, 16(2):101–116.

[Beck and Fowler, 2000] Beck, K. and Fowler, M. (2000). *Planning Extreme Programming*. Addison Wesley.

[Bernon et al., 2002] Bernon, C., Gleizes, M. P., Picard, G., and Glize, P. (2002). The adelfe methodology for an intranet system design. In Giorgini, P., Lespérance, Y., Wagner, G., and Yu, E. S. K., editors, *AOIS@CAiSE*, volume 57 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Boehm, 1988] Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72.

[Bresciani et al., 2004] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

[Cailliau and van Lamsweerde, 2013] Cailliau, A. and van Lamsweerde, A. (2013). Assessing requirements-related risks through probabilistic goals and obstacles. *Requir. Eng.*, 18(2):129–146.

[Castro et al., 2009] Castro, J., Kolp, M., Liu, L., and Perini, A. (2009). Dealing with complexity using conceptual models based on tropos. In Borgida, A., Chaudhri, V. K., Giorgini, P., and Yu, E. S. K., editors, *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*, pages 335–362. Springer.

[Castro et al., 2002] Castro, J., Kolp, M., and Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: the tropos project. *Inf. Syst.*, 27(6):365–389.

[Castro et al., 2013] Castro, J., Kolp, M., and Mylopoulos, J. (2013). A requirements-driven development methodology. In *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*, pages 265–280.

[Cernuzzi et al., 2014] Cernuzzi, L., Molesini, A., and Omicini, A. (2014). The gaia methodology process. In *Handbook on Agent-Oriented Design Processes*, pages 141–172. Springer.

[Chung et al., 2000] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J. (2000). *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishing.

[Dalpiaz et al., 2016] Dalpiaz, F., Franch, X., and Horkoff, J. (2016). istar 2.0 language guide. *CoRR*, abs/1605.07767.

[Dam and Winikoff, 2013] Dam, H. K. and Winikoff, M. (2013). Towards a next-generation AOSE methodology. *Sci. Comput. Program.*, 78(6):684–694.

[del Nuevo et al., 2011] del Nuevo, E., Piattini, M., and Pino, F. J. (2011). Scrum-based methodology for distributed software development. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, pages 66–74. IEEE.

[Ferrario and Guarino, 2008] Ferrario, R. and Guarino, N. (2008). Towards an ontological foundation for services science. In Domingue, J., Fensel, D., and Traverso, P., editors, *FIS*, volume 5468 of *Lecture Notes in Computer Science*, pages 152–169. Springer.

[Ferrario et al., 2011] Ferrario, R., Guarino, N., Janiesch, C., Kiemes, T., Oberle, D., and Probst, F. (2011). Towards an ontological foundation of services science: The general service model. In *10. Internationale Tagung Wirtschaftsinformatik, Zürich, 16.-18. Februar 2011*, page 47.

[Franch, 2010] Franch, X. (2010). Incorporating modules into the *i** framework. In Pernici, B., editor, *Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings*, volume 6051 of *Lecture Notes in Computer Science*, pages 439–454. Springer.

[Gibbs, 2006] Gibbs, R. D. (2006). *Project Management with the IBM®Rational Unified Process®: Lessons From The Trenches*. IBM Press.

[Haesen et al., 2008] Haesen, R., Snoeck, M., Lemahieu, W., and Poelmans, S. (2008). On the definition of service granularity and its architectural impact. In Bellahsene, Z. and Léonard, M., editors, *CAiSE*, volume 5074 of *Lecture Notes in Computer Science*, pages 375–389. Springer.

[Horkoff et al., 2014] Horkoff, J., Barone, D., Jiang, L., Yu, E. S. K., Amyot, D., Borgida, A., and Mylopoulos, J. (2014). Strategic business modeling: representation and reasoning. *Software and System Modeling*, 13(3):1015–1041.

[IBM, 2007] IBM (2007). *The Rational Unified Process, Version 7.0.1.*

[Islam et al., 2014] Islam, S., Mouratidis, H., and Weippl, E. R. (2014). An empirical study on the implementation and evaluation of a goal-driven software development risk management model. *Information & Software Technology*, 56(2):117–133.

[Jalote, 2002] Jalote, P. (2002). *Software project management in practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Kolp et al., 2008] Kolp, M., Faulkner, S., and Wautelet, Y. (2008). Social structure based design patterns for agent-oriented software engineering. *International Journal of Intelligent Information Technologies*, 4(2):1–23.

[Kolp et al., 2006] Kolp, M., Giorgini, P., and Mylopoulos, J. (2006). Multi-agent architectures as organizational structures. *Autonomous Agents and Multi-Agent Systems*, 13(1):3–25.

[Kolp et al., 2011] Kolp, M., Wautelet, Y., and Faulkner, S. (2011). Social-centric design of multi-agent architectures. In Yu, E., Giorgini, P., Maiden, N., and Mylopoulos, J., editors, *Social Modeling for Requirements Engineering*. MIT Press.

[Kruchten, 2003] Kruchten, P. (2003). *The Rational Unified Process : An Introduction*. Addison-Wesley.

[Leitão and Karnouskos, 2015] Leitão, P. and Karnouskos, S. (2015). *Industrial Agents: Emerging Applications of Software Agents in Industry*. Morgan Kaufmann.

[Liu and Yu, 2001] Liu, L. and Yu, E. (2001). From requirements to architectural design ? using goals and scenarios. In *Proceedings of the 1st International Workshop From Software Requirements to Architectures. Toronto, Canada.*, pages 22–30.

[Lucia and Qusef, 2010] Lucia, A. and Qusef, A. (2010). Requirements engineering in agile software development. *Journal of Emerging Technologies in Web Intelligence*, 2(3).

[Lund et al., 2011] Lund, M. S., Solhaug, B., and Stølen, K. (2011). *Model-Driven Risk Analysis - The CORAS Approach*. Springer.

[Mefteh et al., 2015] Mefteh, W., Migeon, F., Gleizes, M. P., and Gargouri, F. (2015). Simulation based design for adaptive multi-agent systems with the ADELFE methodology. *IJATS*, 7(1):1–16.

[Mylopoulos et al., 2013] Mylopoulos, J., Castro, J., and Kolp, M. (2013). The evolution of tropos. In *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*, pages 281–287.

[Pache and Spalanzani, 2007] Pache, G. and Spalanzani, A. (2007). *La gestion des chaînes logistiques multi-acteurs : perspectives stratégiques*. Presses Universitaires de Grenoble.

[Pastor et al., 2011] Pastor, O., Estrada, H., and Martínez, A. (2011). The strengths and weaknesses of the i* framework: an experimental evaluation. *in Giorgini P., Maiden N., Mylopoulos J., Eric Yu editors, Social Modeling for Requirements Engineering, in Cooperative Information Systems series, MIT Press*, pages 607–643.

[Penserini et al., 2007] Penserini, L., Perini, A., Susi, A., and Mylopoulos, J. (2007). High variability design for software agents: Extending tropos. *TAAS*, 2(4):27.

[Pourshahid et al., 2014] Pourshahid, A., Johari, I., Richards, G., Amyot, D., and Akhigbe, O. (2014). A goal-oriented, business intelligence-supported decision-making methodology. *Decision Analytics*, 1:9.

[Ruparelia, 2010] Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3):8–13.

[Sägesser et al., 2013] Sägesser, K., Joseph, B., and Grau, R. (2013). Introducing an iterative life-cycle model at credit suisse IT switzerland. *IEEE Software*, 30(2):68–73.

[Samii, 2004] Samii, A. K. (2004). *Stratégie Logistique, Supply Chain Management : Fondements - Méthodes - Applications*. Dunod.

[Schneider and Winters, 1998] Schneider, G. and Winters, J. (1998). *Applying Use Cases - A Practical Guide*. Addison Wesley.

[Shuja and Krebs, 2007] Shuja, A. and Krebs, J. (2007). *Ibm®; rational unified process®; reference and certification guide: solution designer*. IBM Press, first edition.

[Sivakumar et al., 2013] Sivakumar, N., Vivekanandan, K., and Gayatri, S. (2013). Evaluating the lifecycle coverage of gaia methodology. *International Journal of Computer Technology and Applications*, 4(1):145.

[Sturm and Shehory, 2014] Sturm, A. and Shehory, O. (2014). The landscape of agent-oriented methodologies. In *Agent-Oriented Software Engineering*, pages 137–154. Springer.

[van Lamsweerde, 2009] van Lamsweerde, A. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley.

[Vlaanderen et al., 2011] Vlaanderen, K., Jansen, S., Brinkkemper, S., and Jaspers, E. (2011). The agile requirements refinery: Applying scrum principles to software product management. *Information and Software Technology*, 53(1):58 – 70.

[Wautelet, 2008] Wautelet, Y. (2008). *A Goal-Driven Project Management Framework for Multi-Agent Software Development: The Case of I-Tropos*. PhD thesis, Université catholique de Louvain.

[Wautelet, 2012] Wautelet, Y. (2012). Representing, modeling and engineering a collaborative supply chain management platform. *International Journal of Information Systems and Supply Chain Management*, 5(3):1–19.

[Wautelet et al., 2017] Wautelet, Y., Heng, S., Kiv, S., and Kolp, M. (2017). User-story driven development of multi-agent systems: A process fragment for agile methods. *Computer Languages, Systems & Structures*, 50:159–176.

[Wautelet et al., 2014] Wautelet, Y., Heng, S., Kolp, M., and Mirbel, I. (2014). Unifying and extending user story models. In Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., and Horkoff, J., editors, *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, volume 8484 of *Lecture Notes in Computer Science*, pages 211–225. Springer.

[Wautelet and Kolp, 2011] Wautelet, Y. and Kolp, M. (2011). Goal driven iterative software project management. In Cuaresma, M. J. E., Shishkov, B., and Cordeiro, J., editors, *ICSOFT (2)*, pages 44–53. SciTePress.

[Wautelet and Kolp, 2016] Wautelet, Y. and Kolp, M. (2016). Business and model-driven development of BDI multi-agent systems. *Neurocomputing*, 182:304–321.

[Wautelet et al., 2011] Wautelet, Y., Kolp, M., and Poelmans, S. (2011). Requirements-driven iterative project planning. In Cuaresma, M. J. E., Cordeiro, J., and Shishkov, B., editors, *Software and Data Technologies*, volume 303 of *Communications in Computer and Information Science*, pages 121–135. Springer.

[Weill and Ross, 2004] Weill, P. and Ross, J. W. (2004). *IT governance: How top performers manage IT decision rights for superior results.* Harvard Business Press.

[West, 2003] West, D. (2003). Planning a project with the ibm rational unified process. Technical report, IBM, New York.

[Wooldridge et al., 2000] Wooldridge, M., Jennings, N., and Kinny, D. (2000). The gaia methodology for agent-oriented analysis and design. *In Autonomous Agents and Multi-Agent Systems*, 3:285–312.

[Yu et al., 2011] Yu, E., Giorgini, P., Maiden, N., and Mylopoulos, J. (2011). *Social Modeling for Requirements Engineering.* MIT Press.