UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES)

Dottorato di Ricerca in

INFORMATION AND COMUNCATION ENGINEERING FOR PERVASIVE INTELLIGENT ENVIRONMENTS

Tematica:

Metodologie dell'Ingegneria del Software per lo Sviluppo di Sistemi Software Distribuiti e Complessi

CICLO

XXIX

REQUIREMENTS ENGINEERING FOR COMPLEX SYSTEMS

Settore Scientifico Disciplinare ING-INF/05

Coordinatore: Ch.mo Prof. Felice Crupi

Firma [illegible]

Relatore: Ch.mo Prof. Domenico Saccà

Firma [illegible]

Correlatori: Ch.mo Prof. Angelo Furfaro

Firma [illegible]

Ch.mo Prof. Alfredo Garro

Firma [illegible]

Dottoranda: Dott.ssa Teresa Gallo

Firma [illegible]

Abstract (English)

Requirements Engineering (RE) is a part of Software Engineering and, in general, of System Engineering. RE aims to help in building software which satisfies the user needs, eliciting, documenting, validating and maintaining the requirements that a software has to adequately satisfy.

During its 30 years of RE history, its importance has been perceived with various degrees, from being the most important activity, well formalized and defined in big complete documents which were the bible of the software project, to the opposite side where it has been reduced to just an informal activity, volatile, never formalized, not at all maintained, because ever changing.

The need for well managing requirements is extremely important, mainly for complex systems which involve great investments of resources and/or cannot be easily substituted.

A system can be complex because it is realized by the collaboration of a numerous and heterogeneous set of stakeholders, as for example in a big industrial research project, often co-funded with public resources, where usually many partners, with different backgrounds and languages must cooperate for reaching the project goals.

Furthermore, a system can be complex because it constitutes the IT system of an Enterprise, which has been grown, along the time, by adding many pieces of software, integrated in many and different ways; the IT system is often distributed, ubiquitously interoperates on many computers, and behaves as a whole big system, though developed by many software providers, at different times, with different technologies and tools.

The complexity of these systems is highly considered for several critical industrial domains where features of real-time and fault-tolerance are vital, such as automotive, railway, avionics, satellite, health care and energy; in these domains a great variety of systems are usually designed and developed by organizing and integrating existing components that pool their resources and capabilities to create a new system which is able to offer more functionalities and performances than those offered by the simple sum of its components.

Typically, the design and management of such systems, best known as System of Systems (SoS), have properties not immediately defined, derived and easily analyzed starting from the properties of their stand-alone parts. For these reasons, SoS requires suitably engineered methods, tools and techniques, for managing requirements and any other construction process phase, with the aim to minimize whichever risk of fail.

However, every complex IT system, even though it does not deal with a critical domain, but it supports the core business of an enterprise, must be well governed to avoid the risk of becoming rapidly inadequate to its role. This risk becomes high when many uncontrolled IT developments, aimed at supporting requirements changes, accumulate. In fact, as the complexity grows up, the IT system might become too expensive to maintain and then it should be retired and substituted after some too short time, often with big and underestimated difficulties.

For these reasons, complex systems must be governed during their evolution, both from the point of view of 'which application is where and why', and from the point of view of the supported requirements, that is 'which need is supported by each application and for whom'. This governance would facilitate the knowledge, the management, the essentialness and the maintenance of the complex systems, by allowing efficient support and a long-lasting system, with the consequence of minimizing waste of costs and inadequacy of the support for core business of the enterprise.

This work addresses mainly the issue of governing systems which are complex because either they are the result of the collaboration of many different stakeholders (e.g. are big co-funded R&D projects) or they are Enterprise Information Systems (EIS) (e.g. IT system of medium/large enterprises).

In this direction, a new goal-oriented requirements methodology, named GOReM, was defined which has specific features useful for the addressed issues. In addition a new approach, ResDevOps, has been conceived, that allows to refine the government of the requirements of an EIS which is continuously improved, and which increases and evolves along the time.

The thesis presents the framework of state of the art in which these activities found their collocation, together with a set of case studies which were developed inside some real projects, mainly big projects of R&D which have seen involved the University of Calabria, but also some cases in real industrial projects.

The main results were published and were included in international conference proceedings and a manuscript is in press on an international journal.

Arcavacata di Rende (CS), July 2017 *Teresa Gallo*

Abstract (Italian)

L'Ingegneria dei Requisiti (RE) è una parte della Ingegneria del Software e, più in generale, della Ingegneria dei Sistemi. RE si propone di supportare la costruzione di un software che risponda alle esigenze degli utenti. RE si riferisce alle attività di elicitare, documentare, convalidare e mantenere i requisiti degli utenti, che il software deve soddisfare adeguatamente.

La disciplina RE è passata attraverso una storia di 30 anni, dall'essere l'attività più importante, ben formalizzata e definita in grandi documenti completi che diventavano la 'bibbia' del progetto software, alla situazione opposta di una attività informale, volatile, non formalizzata, non tracciata, perchè i requisiti sono in continua evoluzione.

La necessità di una buona gestione dei requisiti è estremamente importante soprattutto per i sistemi complessi che hanno implicato grandi investimenti di risorse o non possono essere facilmente sostituiti.

Un sistema può essere complesso perchè risultato dalla collaborazione di numerosi e diversi stakeholder, come ad esempio in un grande progetto di ricerca industriale, spesso cofinanziato con risorse pubbliche, dove di solito molti partner, con diverse provenienze e linguaggi devono cooperare per raggiungere i risultati del progetto.

Inoltre, un sistema può essere complesso perchè costituisce il sistema informativo di un'impresa, che si è evoluto nel tempo, con l'aggiunta di molti pezzi di software, integrati in molti e diversi modi; il sistema IT è spesso distribuito, interagisce ubiquitariamente su molti computer, e si comporta come un grande sistema complesso, anche se sviluppato da molti fornitori di software, in tempi diversi, con differenti tecnologie e strumenti.

La complessità di questi sistemi è considerata importante da gestire per diversi domini industriali critici in cui le caratteristiche di tempo reale e tolleranza ai guasti sono vitali, quali il settore automobilistico, ferroviario, avionico, satellitare, la sanità e l'energia. In questi ambiti i sistemi sono generalmente progettati e sviluppati attraverso l'organizzazione e l'integrazione di componenti esistenti che mettono in comune le loro risorse e capacità per creare un nuovo sistema che sia in grado di offrire maggiori funzionalità rispetto

a quelle offerte dalla semplice somma delle sue componenti. Tipicamente, la progettazione e la gestione di tali sistemi, meglio noti come sistemi di sistemi (SoS), hanno proprietà che non possono essere immediatamente definite, derivate e facilmente analizzate partendo dalle proprietà delle loro parti viste stand-alone, ma richiedono adeguati metodi ingegnerizzati, strumenti e tecniche di controllo non solo dei requisiti ma di tutte le fasi della realizzazione del sistema complessivo che mirano a ridurre al minimo qualsiasi rischio di fallire.

Tuttavia, ogni sistema IT complesso, poichè supporta il 'core business' di un'impresa, dove tra l'altro, il processo di evoluzione non è chiaramente definito come un insieme di sottosistemi ben distinti e integrati, deve comunque essere ben governato per evitare il rischio di diventare rapidamente un insufficiente supporto al business, troppo costoso da mantenere e quindi destinato ad essere ritirato e sostituito, con grandi e spesso sottovalutate difficoltà, dopo un troppo breve tempo di utilizzo. Questo rischio diventa alto quando, per supportare i requisiti in continua evoluzione, si accumulano, in modo incontrollato, modifiche successive.

I sistemi complessi devono essere governati nella loro evoluzione, sia dal punto di vista di "quale applicazione si trova, dove e perché ", che dal punto di vista dei requisiti supportati, cioè "quale necessità è supportata da ciascuna applicazione e per chi" . Questo governo facilita la conoscenza, la gestione, l'essenzialità e la manutenzione dei sistemi complessi, permettendo un supporto efficiente e un sistema di lunga durata, con la conseguenza di ridurre al minimo lo spreco di costi e la progressiva inadeguatezza del supporto per il business dell'impresa.

Questo lavoro si occupa dunque principalmente del problema di governare sistemi che sono complessi perché o sono il risultato della collaborazione di molti soggetti diversi (e.g., sono grandi progetti cofinanziati di R&S) o sono sistemi informativi aziendali (EIS) (e.g., il sistema IT di una impresa).

In questa direzione è stata definita una nuova metodologia di requisiti orientata agli obiettivi, di nome GOReM, con caratteristiche specifiche utili per le problematiche affrontate. Inoltre è stato definito un nuovo approccio, ResDevOps, per il governo dei requisiti e dei bisogni di un moderno EIS che è in continuo miglioramento e crescita in funzionalità e che si evolve nel tempo.

La tesi presenta lo stato dell'arte in cui tali attività hanno trovato la loro collocazione insieme ad una serie di casi di studio che sono stati sviluppati all'interno di alcuni progetti reali, principalmente grandi progetti di R&S, che hanno coinvolto l'Università della Calabria, ma anche alcuni casi in una industria privata.

I principali risultati sono stati pubblicati e inclusi in atti di convegni internazionali e un manoscritto è attualmente in corso di stampa su una rivista internazionale.

Arcavacata di Rende (CS), Luglio 2017 *Teresa Gallo*

Acknowledgments

Questo lavoro di Tesi ha visto il fondamentale sostegno e contributo di molte persone che mi hanno supportato durante questi tre anni di dottorato.

Prima di tutto voglio ringraziare il mio Relatore, il Prof. Domenico Saccà, mio maestro di lunga data nonché autorevole consigliere nelle scelte professionali della mia lunga esperienza lavorativa. Lo ringrazio per aver accettato di farmi da supervisore in questa meravigliosa avventura che è stata questo corso di dottorato.

Il mio sentito grazie ai miei Correlatori. In particolare, alla pazienza, dedizione, preparazione e impegno del Prof. Angelo Furfaro, che mi ha seguita ed indirizzata e a cui devo davvero tanto. Senza di lui non so se e come sarei riuscita in un percorso che considero onorevole e certamente affascinante e proficuo per la mia crescita professionale. Ringrazio inoltre il Prof. Alfredo Garro, che ha contribuito come meglio non poteva, considerati i suoi numerosi importanti impegni che lo hanno tenuto lontano ed occupato, portando lustro internazionale all'Università della Calabria.

Tutti e tre i miei supervisori hanno sempre dimostrato simpatia ed affetto nei miei confronti, ricambiate al centuplo dal profondo del mio cuore.

Non posso non citare le care amiche Ing. Sabrina Graziano e Ing. Simona Citrigno, fantastiche professioniste che mi hanno accolto all'inizio di questa esperienza, hanno condiviso con me conoscenze e impostazioni pregresse e con cui ho fruttuosamente collaborato in ICT-SUD, condiviso e scambiato idee, seguito lavori di tesi e che mi hanno coinvolto, per volontà del mio tutor, su attività di progetti di ricerca industriale che hanno poi segnato il mio percorso di dottorato. Con esse ho condiviso fatiche e gioie nella stesura di alcune pubblicazioni, oltre che con il Ph.D. Andrea Tundis che anche ringrazio per le interessanti discussioni, la disponibilità ed il sostegno.

Voglio inoltre ringraziare tutti i colleghi di dottorato, di solo qualche decina di anni più giovani di me, che mi hanno sempre considerata una di loro. Nutro nei loro confronti un affetto immenso. Sono tutti motivati, brillanti e hanno grande dedizione per la Ricerca, quella con la 'R' maiuscola che caratterizza l'Università della Calabria tutta ed il DIMES in particolare. Auguro a tutti loro la meritata brillante carriera e che possano realizzare tutti i loro sogni.

Io ho realizzato il mio sogno, spero non l'ultimo, con questo corso di dottorato che mi onoro di aver seguito e grazie al quale posso a buon ragione dire: non si finisce mai di imparare, i sogni si possono realizzare a qualunque età e mai dire mai!

Ringrazio inoltre la mia amica e compagna professionale Dott.ssa Antonella Mecchia che ha riletto pazientemente questo lavoro in inglese. I numerosi typos sono comumque ascrivibili a me e se non sono troppi è merito suo.

Grazie a tutti coloro che avranno la pazienza di leggere questa tesi e vorranno darmi i loro feedback in qualunque momento.

Luglio 2017 *Teresa Gallo*

Contents

1 Introduction 1
1.1 Reference context, Motivations and Objectives 1
1.2 Main Results 3
1.3 Thesis Overview 3
1.4 Publications 5

2 Background 7
2.1 Life Cycle models 7
2.2 Requirements Engineering 15
2.3 Customer vs. Provider issues 18
2.4 IT Business Management 19

3 State of the art solutions and current trends 21
3.1 RUP, agile methods and AUP 24
3.1.1 Rational Uniform Process 24
3.1.2 Agile Uniform Process 24
3.1.3 Lean Agile methods 25
3.1.4 eXtreme Programming (XP) 27
3.1.5 Scrum 28
3.1.6 Kanban 29
3.2 DevOps 30
3.3 RE: Goal oriented methodologies 32
3.4 Agile based Fameworks 35
3.4.1 Disciplined Agile Delivery 35
3.4.2 Scalable Agile Framework 35
3.5 IT business strategies 37
3.5.1 Agile Lifecycle Management (ALM) 38

4 Requirements Engineering: GOReM 45
4.1 Introduction to GOReM: yet another requirements methodology 46
4.2 GOReM overview 48

4.3 GOReM: Lesson learned . . . 51
4.4 Case Studies . . . 56
4.4.1 Case Stydy: InMoto - Tourism . . . 57
4.4.2 Case Study: Cyber Security - Systemic Risk in Bank . . . 62
4.4.3 Case Study: Cyber Security - Compliance analysis . . . 75
4.4.4 Case Study: Cyber Security - Systemic Risk in Payment On Line . . . 93

5 Long-lasting EIS with a new approach: ResDevOps . . . 105
5.1 Introduction to the problem statements . . . 105
5.2 ResDevOps = ResDevs+DevOps . . . 107
5.2.1 ResDevOps Context Modeling . . . 110
5.2.2 Case Studies . . . 113
5.2.3 Conclusion and future work . . . 121
5.3 ResDevs might extend some pre-existing frameworks . . . 123
5.3.1 Extending ResDevOps . . . 123
5.3.2 ResDevs inside SAFe and DAD . . . 125
5.3.3 ResDevOps with Atlassian tools . . . 125

6 Conclusions . . . 137
6.1 Main contributions . . . 137
6.2 Future Work . . . 139

References . . . 141

List of Tables

4.1 THE CONTEXT MODEL - MAIN CONCEPTS 54
4.2 THE CONTEXT MODEL - RULES AND REGULATIONS . . . 55
4.3 THE SCENARIO MODEL - ROLES 55
4.4 THE SCENARIO MODEL - SWOT ANALYSIS 55
4.5 THE SCENARIO MODEL - RULES AND REGULATIONS . . . 56
4.6 THE APPLICATION MODEL - MAIN CONCEPTS 57
4.7 SCENARIO MODEL FOR RISK TREATMENT: ROLE AND GOAL DESCRIPTION 70
4.8 SCENARIO MODEL FOR RISK TREATMENT: RULES AND REGULATIONS DIAGRAM 71
4.9 CONTEXT MODEL: SecaaS CONTEXT RULES AND REGULATIONS DIAGRAM 80
4.10 SIEM SCENARIO MODEL: ROLES 82
4.11 SWOT ANALYSIS 84
4.12 SIEM SCENARIO: RULES AND REGULATIONS DIAGRAM 85
4.13 PHASES, ACTIVITIES AND WORK PRODUCTS OF RAMSoS 94
4.14 GOReM EXTENSIONS THROUGH THE RAMSoS METHOD 95
4.15 STAKEHOLDERS, ROLES, GOALS AND DEPENDENCIES . 99
4.16 STAKEHOLDERS, ROLES, GOALS AND DEPENDENCIES . 104

List of Figures

2.1 Waterfall model 11
2.2 Spiral model 12

3.1 Rational Uniform Process (RUP) model 25
3.2 Agile Uniform Process (AUP) model 26
3.3 Kanban board built with Atlassian tools 30
3.4 The DevOps loop 32
3.5 An example of sheet for documenting features 33
3.6 DAD basics framework 36
3.7 3 levels SAFe 37
3.8 4 levels SAFe 38
3.9 ALM three sectors 39
3.10 ALM Governance 40
3.11 ALM Development 41
3.12 ALM Operations 42

4.1 GOReM meta model 49
4.2 GOReM Methodology 51
4.3 Tourism in mobility Context Model: Stakeholder Diagram 58
4.4 Tourism in mobility Context Model: Softgoals Diagram 59
4.5 Tourism in mobility Context Model: Softgoals Dependency Diagram 60
4.6 Tourism in Mobility Brokerage Scenario Model: Role Diagram . 61
4.7 Tourism in Mobility Brokerage Scenario Model: Role Diagram . 62
4.8 Tourism in Mobility Brokerage SCENARIO: Roles Dependency Diagram 63
4.9 Event Tourism Application Model: Use Case Diagram 64
4.10 Event Tourism Application Model: Main process 65
4.11 Event Tourism Application Model: Event packaging process 65
4.12 Context Model: The Banking Business Continuity's Softgoals and Dependencies Diagram 67

4.13 Scenario Model for Risk treatment: SoftGoals-Roles-Goals Diagram ... 69
4.14 Application Model for Cyber Systemic Risk: Actor Diagram ... 72
4.15 Application Model for Cyber Systemic Risk: Main Use Case Diagram ... 73
4.16 Application Model for Cyber Systemic Risk: Sistemically important processes Management Process ... 74
4.17 Application Model for Cyber Systemic Risk: Incident Management Process ... 74
4.18 Context Model: SecaaS Stakeholder Diagram ... 77
4.19 Context Model: SecaaS Softgoal Dependency Diagram ... 77
4.20 SIEM Scenario Model: Role Diagram ... 81
4.21 SIEM Scenario Model: Goal Diagram ... 83
4.22 Compliance Analysis Application Model 1: Roles, Actors and (Main) Use Case Diagram ... 86
4.23 Compliance Analysis Application Model 2: Actors and related Role Diagram ... 89
4.24 Compliance Analysis Application Model 2: Main Use Case Diagram ... 89
4.25 Compliance Analysis Application Model 2: Process Diagram ... 90
4.26 PEO Context Model: Stakeholders, Softgoals and Dependency Diagram ... 97
4.27 PEO Scenario Model: Stakeholders, Roles and Goals ... 98
4.28 PEO Notification SMS Application Model: MAIN Use Cases ... 98
4.29 Low Risk of the PEO service ... 100
4.30 High Risk of the PEO service ... 101
4.31 High Risk of the PEO service ... 102
4.32 High Risk of the PEO service ... 103

5.1 ResDevOps interoperation schema ... 109
5.2 Context Model: ResDevOps Stakeholder diagram ... 111
5.3 Context Model: ResDevOps Dependency diagram ... 112
5.4 Context Model for the EIS of the customer: Stakeholder Diagram ... 115
5.5 Context Model for the EIS of the customer: Dependency Diagram ... 116
5.6 Scenario Model for Overall Enterprise Organizational Communication Management: Role Diagram ... 117
5.7 Application Model: Application for issue management – Role to Actor mapping and Issue Tracker Use Case ... 118
5.8 EIS requirements model with GOReM ... 118
5.9 New Application Model related to an Application for asset management: Actors involved by Roles ... 120
5.10 New Application Model related to an Application for asset management:Main use cases ... 121

5.11 New Application Model related to an Application for asset management:The main process 121
5.12 The initial workflow implemented as part of Help Desk tool for the new application model 122
5.13 EIS requirements model and the Service Desk Tool 122
5.14 The SAFe House of Lean 126
5.15 Scenario Model for EIS Communication Management: Roles Diagram 128
5.16 GOReM in JIRA: Projects and issues, modeling the context, the scenario and some applications 130
5.17 GOReM in JIRA: Context Model: EIS for the customer and its Stakeholders issues 130
5.18 GOReM in JIRA: Context Model: EIS for the customer and its Sofgoals issues 131
5.19 GOReM in JIRA: Context Model: Stakeholder Dependency Diagram 131
5.20 GOReM in JIRA: Context Model: Stakeholder and its linked issues 131
5.21 GOReM in JIRA: Scenario Model: EIS Communication and its linked issues 132
5.22 GOReM in JIRA: Application Model: Asset Management and its linked issues 132
5.23 GOReM in JIRA: Application Model: Asset Management Process 133
5.24 Asset Management JIRA SW Projects 133
5.25 Asset Request related to the application model 'Asset Management' 133
5.26 Asset Request JIRA workflow 134

1 Introduction

IT system, application, software, IT product, are often used as synonyms, while in specific context they have different meanings. Generally speaking, the distinction is between software development context, *software or application*, which must be installed on a computer to run and, technological context, *IT system or IT product*, which is strictly composed by specialized hardware together with its software. So, a computer with its operating system, is an IT system in the same way as an IT system is an aircraft if it is able to fly or a ship if it is able to sail or a washing machine if it is able to wash, while a text editor, a report maker, an issue tracker, an operating system are examples of software or application.

In this work it is often used the term "system" or "IT system" even though it is addressed mainly the software part of it, but it is not forgotten that software needs adequate hardware to work.

Another distinction is between "software" and "application". "Software" is best used when we refer to the software development lifecycle (SDLC) where the focus is on the production process of the software, while "application" refers mostly to a software program, result of an SDLC, which was delivered in some operational environment, where the application runs and should be managed by an application lifecycle model (ALM).

1.1 Reference context, Motivations and Objectives

By **Complex System**, as it will be explained later, it is possible to refer to a system whose realization is the result of a project which implies the collaboration of numerous partners and stakeholders, each one having many different expectations.

However, by complex system it is mainly meant the result of the evolution along the time of the IT system which runs inside an Enterprise. Whichever Enterprise has some IT system which support its business. From simple standalone computers with ERP system (e.g, an electronic sheet) to advanced robotic systems for system automation. Current IT systems are very often distributed, are the results of many integrations among different systems, they inter-operate to exchange data using many different and/or sophisticated ways. The result is a complex system, generally known as Enterprise Information System (EIS), where boundaries among the different compound systems are difficult to find over the time. Complexity comes from integration or interoperability, functionalities addition, new software introduction, which have transformed the EIS along the years to a difficult to control, ubiquitous, pervasive IT system. Main issues to be managed are:

1. Verticalization, as EIS support the most part of the enterprise business's work.
2. Distribution, as the communication's evolution (smartphones/tablets/WiFi tools/cloud services) has delocalized software system, employers and jobs.
3. Pervasivity, as EIS is intelligent and often invisibly used (sensors, applet, software agents, cameras, ...).
4. Evolution, as an EIS is improbably completely substituted as a whole. Instead, parts of it have been enhanced in performance/functionalities/used technologies.
5. Cooperation among providers and the customer, as EIS subparts can be developed/enhanced by different subjects.

In particular, Complex Systems' evolution has to be mastered with the aim not only to avoid collapsing of the EIS but also to ensure long life to the EIS, as changing it as a whole is a risky and expensive operation. On the other hand, EIS must be maintained efficient, safe, with a clear design of its evolution along the time, up to date in functionalities/technologies/performances and with affordable costs. This is very important, because the business of the Enterprise completely depends on how well these topics are mastered.

Nowadays, where application and, more generally, system providers are looking for maintaining or improving their business in a worldwide widespread competition context, it becomes strategic to reserve big attention to 'customer satisfaction'. On the other hand, customers need to care about their vital IT system which supports all their Enterprise businesses until becoming itself a business. So, every Enterprise has a core business supported by its internal, strategic and continuously evolving IT business, aiming to maintain the suitability of its complex IT system.

Mastering complex systems deals with:

1. Software Development: Tools, Methodologies, Software Development Life Cycle (SDLC) or Software Production Processes
2. Research: Industrial Research projects, funded by Public Authorities and/or by private investments
3. Software Project Management: Project Management methodologies and tools, Software Development Program and Portfolio Management
4. Application Lyfecycle Management (ALM), which adds Governance and Maintenance to the product development

5. Scaled Agile Frameworks, IT Business Management and Enterprise Information System (EIS) Management.

This work addresses some aspects of those large issues. In particular:

1. Requirements Engineering and its central role in complex systems,
2. From Research to Operation environments processes,
3. EIS and IT business management

To this aim, this work uses important state of the art concepts as *Software life cycles*, *Agile methodologies and DevOps approach*, *Requirements Engineering* and in general, *project/program/portfolio and IT business management.*

1.2 Main Results

Starting from the research objectives above described, the main contributions resulting from the research activity presented in this thesis concerns the definition of:

1. a goal oriented methodology, called GOReM [23], which has been used in modeling requirements in different applicative domains [29, 26, 30] inside big industrial research project, where the University of Calabria has been an important research partner (e.g., DICET-InMoto and Cyber Security) [28, 25],

2. a methodological process, called ResDevOps for supporting long-lasting EIS [27]

3. an applicative framework, defined with widespread commercial tool, Atlassian JIRA, for a suitable management of the IT business inside an Enterprise, supporting EIS Management.

These activities allowed also, to deal with a variety of interesting application domains as services in cloud, cyber security, compliance analysis, systemic risk, simulation, tourism in mobility and asset management.

1.3 Thesis Overview

This thesis is organized as follows. In Chapter 2 a background of the main Software Engineering concepts, are presented. In particular, involved issues in life cycle models from Waterfall model to Software Development Lifecycle Management (SDLM), are discussed in Section 2.1, Requirements Engineering is focused in Section 2.2, customer/provider satisfaction, is faced in Section 2.3, and IT Business Management project with program and portfolio management issues, are discussed in Section 2.4.

Chapter 3 deals with available state of the art solutions and current trends. In particular, in Section 3.1 the widespread RUP (Rational Uniform Process) development process and the main known agile methods are introduced; in Section 3.2 the DevOps philosophy which is currently welcomed by customer and pursued by many agile method supporters, is shown. The most famous goal-oriented methodologies and approaches are presented as a good trend for Requirements Engineering (RE) in Section 3.3. Some agile based frameworks adopted and chased by famous branded enterprises, as SAFe (Scaled Agile Framework) and DAD (Disciplined Agile Delivery) are discussed in Section 3.4. Finally, useful current trend for IT strategy and management are presented in Section 3.5 with focus on Application Lyfecycle Management (ALM).

Chapter 4 presents motivations and the definition of a new goal oriented methodology for RE, devised at the University of Calabria, named GOReM. In particular, in Section 4.1 the motivations for a yet another methodology are presented. The main concepts behind GOReM, are introduced in section 4.2. GOReM has been applied in some big research industrial projects and lessons learned from these experiences and how these have been receipted in the methodology, are presented in section 4.3. In section 4.4, the main case studies in modeling different domains from Tourism to Cloud, Cyber Security and Systemic Risk, as guided by some real needs of big research projects, are presented. The combination of GOReM with a methodology, named RAMSoS, which allows to add simulation for System of Systems, is the last case study of the section. These case studies were presented in international conferences related to the faced domain.

Chapter 5 is devoted to present how GOReM has been also important in defining a new Software Engineering approach, named ResDevOps, which was presented at the 24th International Conference on Requirements Engineering (RE) in September 2016. This new approach aims to having long-lasting EIS. Section 5.1 introduces the problem statement. Section 5.2 describes the approach as a combination of two parts: ResDevs and DevOps. The same GOReM is used to model the context in which ResDevOps operates and some case studies, both from the university and the industry, are discussed, together with the further work which would be interesting to do. Section 5.3 is devoted to the description of how some issues outlined by ResDevOps have been managed by recent frameworks for IT business management in an Enterprise, and how the approach might be inserted in those existing frameworks to be extended/enriched. Lastly, an implementation of the approach for an industrial case study with Atlassian JIRA is shown.

In Chapter 6 the contributions of this thesis are summarized and ongoing and future works delineated.

1.4 Publications

Contents of this thesis were published and presented in international Conferences as well as in the Journal Concurrency and Computation: Practice and Experience where is in press:

- S. Citrigno, A. Furfaro, T. Gallo, A. Garro, S. Graziano, D. Saccá, "Mastering concept exploration in large industrial research projects", Proceedings of the INCOSE Italian Chapter Conference on Systems Engineering (CIISE), Rome Italy, November 2014

- A. Furfaro, T. Gallo, A. Garro, D. Saccá, A. Tundis, "Requirements specification of a Cloud Service for Cyber Security Compliance Analysis", in Proc. of the 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech'16), Morocco, IEEE, 2016.

- A. Furfaro, T. Gallo, D. Saccá, "Modeling cyber systemic risk for the business continuity plan of a bank", in Proc. of the Int. Cross Domain Conference and Workshop (CD-ARES 2016), Salzburg, Austria, Aug. 31-Sep. 2 2016.

- A. Furfaro, T. Gallo, A. Garro, D. Saccá, A. Tundis, "ResDevOps: A Software Engineering Framework for Achieving Long-lasting Complex Systems", in Proc. 24th IEEE International Requirements Engineering Conference, Bijing, China, Sept 12-16 2016

- S. Citrigno, A. Furfaro, T. Gallo, A. Garro, S. Graziano, D. Saccá, A. Tundis. "Extending GOReM through the RAMSoS method for supporting modeling and virtual evaluation of the Systemic Risk", Proceedings of the INCOSE Italia Conference on Systems Engineering (CIISE), Turin, Italy, November 14-15, 2016

- A. Furfaro, T. Gallo, A. Garro, D. Saccá, A. Tundis, "Cybersecurity Compliance Analysis as a Service: Requirements Specification and Application Scenarios", CloudTech Special issue in Concurrency and Computation: Practice and Experience (2017 in press)

- A. Tundis, S. Citrigno, T. Gallo, A. Garro, S. Graziano, D. Saccá, M. Mühlhäuser, "Systemic Risk Modeling and Evaluation through Simulation and Bayesian Networks", ARES'17, Reggio Calabria, Italy, August 29-September 01, 2017 (accepted).

2 Background

This chapter is devoted at framing the context of Software Engineering concepts behind this thesis work.

Software life cycle models historically began to engineer the steps used to produce software and they evolved to engineer specific phases, as the Requirements Engineering phase where this work is focused.

Software is developed for a customer and is produced by software providers (i.e., professionals, enterprises, or the IT internal department of the customer). Customers and providers have different goals.

Customers want to have a quality software, as cheap as possible, but which well satisfies their functional and non-functional needs.

Providers, on the other hand, want to increase the number of customers and, as consequence, their gains. To reach this objective they must build a good reputation in satisfying customers' requirements.

Nowadays, a suitable IT Management is one of the most important business of whichever Enterprise, because its Information System (EIS) supports, in a vital way, its core business.

2.1 Life Cycle models

Life cycle models, in this context, relate to software and to the usual activity phases involved in the process to produce software and then to provide applications, such as

- establishing business context,
- defining project management,
- eliciting requirements,
- designing solutions,
- deciding tools and the software factory to be used
- establishing quality parameters and/or acceptance tests,

- preparing contracts where responsibilities, times, costs, milestones for delivering, installing and monitoring, developing software, managing releases and configurations and other stuff, are as clear as possible.

Along the time there have been various software life cycles: from the famous *waterfall model* till the *agile software development methodologies*, passing from *prototyping* and *spiral models*.

The concept of life cycle or software production process was born together with the Software Engineering (SE) concept that aimed at transforming software development from a craft activity to an engineering one. This necessity was born as long as the developed software transited from single standalone programs to IT systems composed by many components. Then, SE discipline was born between years sixty to seventy [1, 2, 3], for both the necessity to organize always more complex, pervasive and critical programs (i.e. from structured to object-oriented programming with growing (cyber) security attention) and the necessity to integrate different pieces of, even pre-existing, programs, developed by many different providers, with different user typology, requirements, programming languages, locations, ... (i.e. from component-based design to distributed programming)[4, 5].

Nowadays Software Engineering is related to every aspects of the software development; it is an important research theme and a central topic in the industrial sector, even though the idea to have a well engineered process for building software, similarly to the civil and mechanics disciplines, is not a reality, yet. In my opinion this is a goal far away to be reached: software development is, and will be in the long term, a creative activity, where methodologies and tools surely will improve their support in giving a direction and in improving communication among customers and providers, among providers, among developer teams, among all involved stakeholders, with the aim to reach software solutions with an acceptable satisfaction of the users' needs.

The different life cycle models can fall in more then one of the following typologies, which in the comings and goings of the times appear under a different name, although with improved features: waterfall, iterative, incremental and evolutionary.

1. Waterfall models were the first attempt to define a process in software development. They consist in performing sequentially, the phases of analysis, design, development, testing and maintenance of the software. They differ for the names of the phases, but also for, at least, the number of the phases, by extending both the begin with the phases of project planning and contract definition, and the end,with the phases of installation in operation, monitoring, correction evolutionary and retirement of the software. Although most of the literature declares it is a finished model, many others say it is always alive and used in many realities. Both point of views have their reasons because the waterfall model is used for very small software development and it is well alive inside other modern mod-

els, even if it has a different name and a specific aim. The first Waterfall model was introduced in 1970 [2]. Figure 2.1 shows graphically the process. It is worth noting that this model is designed such that it is not possible to move onto the next phase of process until the preceding phase is complete. From this derive the name ”waterfall”, even though some iterations between phases have been always outlined.

The whole process of software development, according to the Waterfall model, starts by well understanding the requirements and needs of the customer. Once this step has been completed, the analysis and design phases can start.

Analysis and design are the most intensive steps and involve the top engineers, who ideate a design that would perfectly meet all user requirements and that is robust enough for implementation. Once the design is ready, coding starts. Separate teams can focus on a small part of the entire software project and all these parts will be put together in an integration phase that eventually follows.

Once the software is ready, the testing and debugging phases start. Here, every designed feature of the software are tested and bugs, if any, are corrected. From the testing phase, it may be necessary to go back to verify the design phase. This is followed by the actual acceptance phase of the application, where the resulting program is delivered to the customer. If the acceptance phase fails, revision of the requirements phase might be necessary.

Many criticisms have been reserved to the risks behind this model, because of the risky delays in discovering issues in each phase. This is especially true in the Requirements phase, where misunderstandings are very risky to be discovered after too long time and after that much money has been already spent.

2. Iterative models allow to coming back to a previous phase for reworking after that, in the current phase, it has been discovered that some issue needs to be improved and clarified. Often these models have adopted the strategy to intentionally produce some draft results in a phase, with the aim to better work using the draft results of a successive phase. So, for example, the analysis phase produces successive drafts, each of which benefits from the results of the draft design following phase. Same thing happens after a first attempt to develop the program, that might require to go back to the phases of design or even analysis, to better approximate the best expected result. And so on, until the final delivery of the software is ready for the customer.
3. Incremental models plan to develop the application as a set of subsequent self-contained pieces. For each subpart the waterfall or the incremental models can be adopted. In addition, incremental models are also those models that start from a complete requirements analysis phase and, even, a complete design phase, before planning the next phases as autonomous subparts. Each subpart plans some interfaces to communicate each other,

but maintaining their autonomy. Often, an integration phase allows to have a single application to be delivered to the customer.

4. Evolutionary models see the application as for the incremental models, but they differ because they use to add a whole subpart in another subpart that needs the previous one to correctly work. In other words, they build the software by growing and evolving the subparts that have been previously developed.
5. Prototype models are evolutionary for nature and then completely different from the linear waterfall models. They involve the creation of *prototypes* of the final software, continuously improved and delivered to the customer, allowing to get user feedbacks used to plan new development efforts, until a final application that satisfy the customer needs, is reached. Thus, the process starts by developing basic functionalities and user interfaces, as understood by some preliminary meetings with the customer. By successive demos of the evolving prototype, based on the customer feedbacks,the process reaches the final state when the customer says it is what he wants.
   In other words, according to the customer feedbacks, the prototype is reworked and it keeps improving through better designing and coding, until it is transformed into the program able to satisfy all customer needs. This is a kind of interactive design and the end user is involved in every stage of development. Every evolving prototype goes through testing and debugging phases, including the final program, before the actual deployment.
   These models have the intrinsic problem known as "spaghetti house", because at the end of the development project, it is not clear neither requirements nor design and integrations of the final delivered application, with heavy consequences on the maintenance phase and in the future improvement of the application itself.
6. Spiral models combine the features of the prototype model and the waterfall model. In fact, they include the phases of the waterfall model, while removing almost every possible/known risk factors from it [6], and include the prototype approach as a good means for solving any issues resulting from the risk analysis. Figure 2.2 shows the representation of a spiral model that iterates planning, requirements, design and implementation on definite steps until it reaches the final application. Each iteration has a risk analysis, a prototyping and, possibly a simulation activity. Actually the risk analysis is the most important part of spiral model. In this phase, all possible alternatives, which can help in identifying risks and its solutions in the current phase, are evaluated with the help of prototype and simulation. For example, if risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed with the available data and find out a possible solution in order to deal with the potential changes in the requirements. The same applies to design and development phases, which via a testing phase, allows a customer evaluation and an identification of problems or errors to be solved in a next iteration.

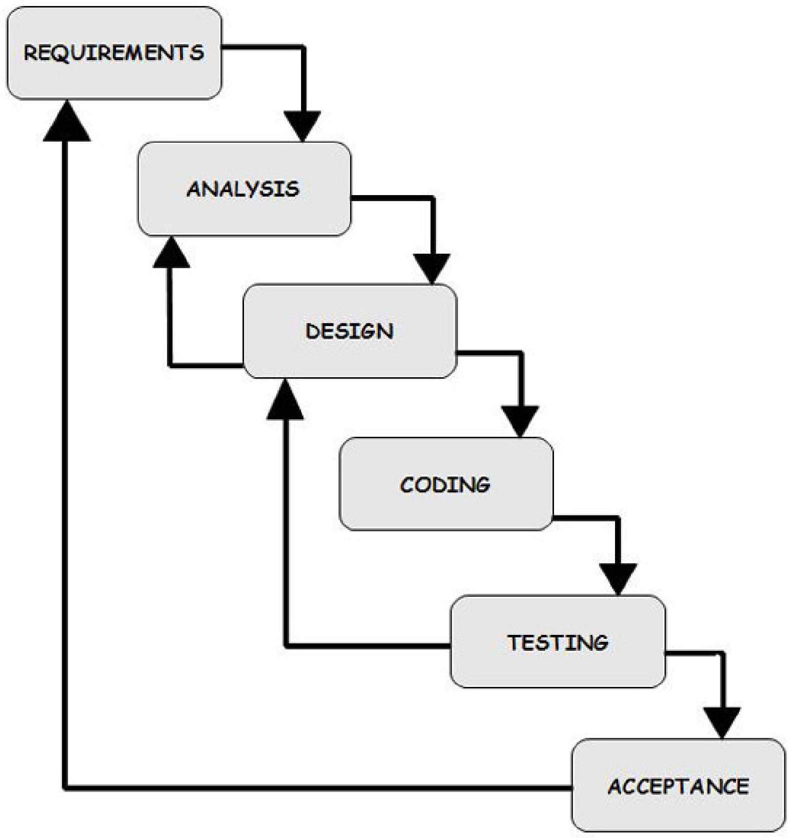


Fig. 2.1. Waterfall model

All life cycle models have in common most of the activities to be done. The differences are in the followed processes and in the pre-conditions to be verified before that those activities might be performed. Indeed, if in the years seventy, requirements, design and development were the only main activities, nowadays instead, a lot of activities have to be considered and managed. In the following, a brief description of the most usual activities:

- idea defining, scope understanding and bounding inside the preexisting IT system: this deals with the necessity, in the very initial approach to the potential customer needs, to (i) well define the idea and its usefulness and feasibility, (ii) understand what is the concern of the development project that it is going to be established, in terms of what is inside the scope and what is outside of the scope, (iii) establish the hw/sw subparts of

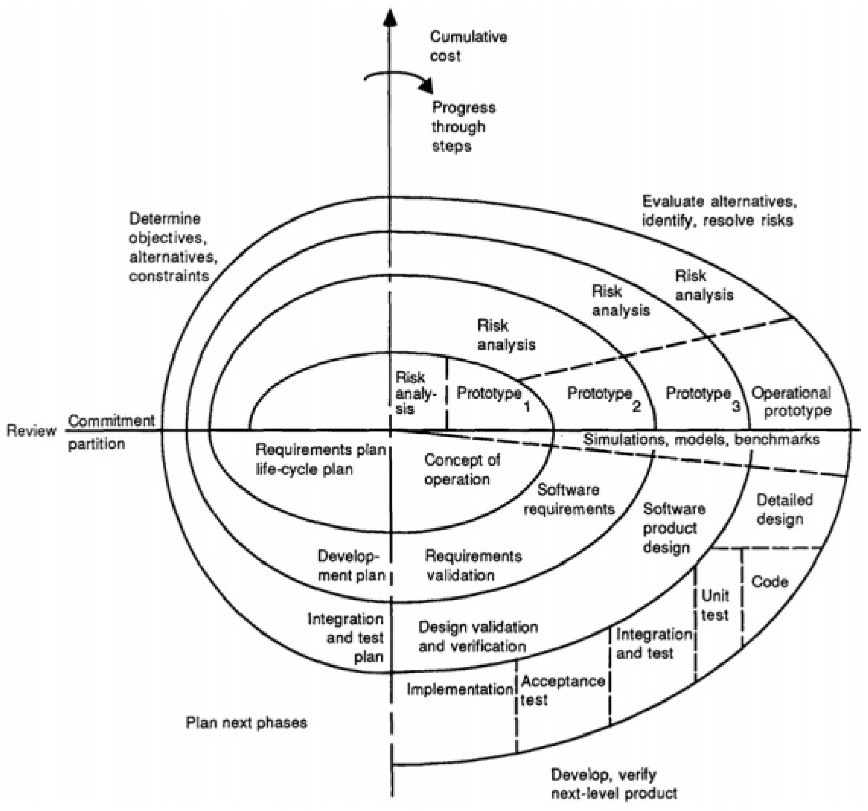


Fig. 2.2. Spiral model

the preexisting IT system involved in the project for integrations and/or changes.

- process defining, that is deciding the workflow to be followed during the project, including life cycle but also project plan, milestones and points of control and verification with the customer, process of the payments, responsibilities, as well as roles and people involved as interface between customer and provider.
- contract signing. Unfortunately, a contract must be signed before the requirements is well defined. To this end, the process in the previous activity, where some verification points should allow to guarantee both customer and provider interests, is very important.
- requirements elicitation and analysis are the major activities both expensive and strategic. Every life cycle deals in a different way with requirements, which are imprecise, subject to misunderstanding and evolving in nature. Requirements engineering shows the current issues and it is described in the next section.

- design solutions. Often analysis and design are considered a single activity whose aim is to analyze the requirements and to architect a solution in terms of algorithms or functionalities, hardware and basic software, performance and possible issues or risks
- environments setting, deals with the important activity to define the software factory and its environments for development, test and simulation, as well as to establish the production environments, that is the hw and sw where the resulting software must operate. Each environment must be precisely characterized by a location, a hw and sw, including description of networks communication and reachability, names, version and architecture.
- project management and team set up deal with the activity to establish management process, team composition and, for each person, its function and responsibility. This activity depends on the life cycle decided to be adopted both on the provider side and on the customer side.
- development or coding activity, which might involve many teams and different programming languages
- test, verification and validation, deal with the activities to test the software for bugs but also to verify if the development functionality conforms with the design and the established requirements, as well as the validation of software by the end-users/customer.
- version, configuration and release management, deal with the life cycle and usually divides the software to be developed in subparts. Each subpart, during its life, is provided with an identifier for its actual version. This version will run inside a configuration of a set of different subparts with their-own version, and it is identified by a unique identifier. In addition, a release of the software, has an identifier and it is done by a configuration where a subset of the overall software subpart, could be installed for a verification and even a validation in production.
- delivering is the activity to freeze a release, ready to be installed in the final customer environment
- operation, is the activity to install a delivered release of the software
- monitoring is the activity to observe and log the behavior of the installed release during the usage in the production environment. Bugs and observations of the end user are monitored and reported to the provider
- maintenance is the activity which usually deals with the bugs fixing, even though there exists the evolutionary maintenance, dealing also with minor reworking.
- remaking or innovation, should deal with the customer necessity to maintain updated the software by many point of views: legislative, security, performance, continuous improvement. Life cycle usually does not contemplate this activity, because it could be included, in some way, in the maintenance.
- retirement is the final action of an application, when it might be retired for being substituted. This activity is very difficult in the today's IT systems

because they are complex, ubiquitous and often not well known to have a clear process for a painless retirement of a specific application.

All those activities complicate a lot the Software Development lifecycle (SDLC), normally covering the software lifecycle and the management activities related to the development of the software until the delivery to the customer. Also the life cycle model is more complicated with respect to the waterfall mode. The spiral model in 2.2 gives an idea of the complexity of the process. This fact together with the complexity of the software to be developed, arose the need and the success of current lean and agile development approaches.

Thus, in the field of Software Engineering, lean agile methodologies [13] developed as a reaction to these development processes when they started to be seen too much bureaucratic in their aim to reach the engineering methods. Instead, new agile methods attempt a useful compromise between absence of process and too much process, providing that a just enough process is a reasonable payoff. The lean methods instead are related to the production just in time, without predicting what might be needed in the future, because changes may arise, destroying our efforts.

It is a common idea that lean agile methods are less document-oriented, and are rather code-oriented, as they state that the principal part of documentation is source code [16].

However, this is the consequence that, first of all, lean agile methods are adaptive rather than predictive [32]. Traditional engineering methods tend to deeply planning the software process and plans are very long in time. This could work well until things do not change. So their nature is to avoid changes. The lean agile methods, however, welcome changes, and agile processes adapt themselves to the changes.

In addition, these methods, better known as only agile methods, are people-oriented rather than process-oriented. The goal of traditional engineering methods is to define a process that will work well if it is properly used. Agile methods assert that no process will ever exactly be used by the development team, so the role of a process is to be adaptable and to support the development team in their work and not to predict or impose what the development team must do next.

However, agile methods do not concentrate on software development process, which is indeed fast and customer oriented.

Activities as governing the overall development process from the inception of the idea of a development project, until the deployment, maintenance in operation, update and retirement of the application (i.e. the result of the development process), and even the way as the application is inserted and maintained into an overall Information System (IS), are not treated at all in the common Software Development Life Cycle (SDLC). An answer to this issue is the so called Application Lifecycle Model (ALM) [31], considered the current interesting trend inside the agile enterprise. Chapter3 will explain ALM concern.

2.2 Requirements Engineering

The term Requirements Engineering was probably born in 1979 in a TRW technical report [14] but did not come into general use until the 1990s with the publication of the IEEE Software Requirements Engineering (2nd Edition) and the establishment of a conference series on requirements engineering that has evolved into the current International Requirements Engineering Conference (RE).

Requirements can be defined as prerequisites, conditions or capabilities needed by users (individuals or systems) to solve a problem or achieve an objective. Requirements thus specify desired objectives. In computer science, they describe functions and features of an IT System. The discipline of requirements engineering (RE) aims at increasing the quality of system development by providing systematic procedures for collecting, structuring, and documenting requirements.

According to Sommerville [21] the requirements engineering process aims to produce an agreed requirements document that specifies a system satisfying stakeholder requirements.

Sommerville distinguish four main activities in the requirements engineering process: feasibility study, requirements elicitation and analysis, requirements specification, requirements validation.

1. Feasibility study aims to estimating if the user needs might be satisfied, considering the existing technology, the cost-effective from a business point of view and from the budget available. This activity should be quick and cheap and should address the decision to go ahead or to stop the process.
2. Requirements elicitation and analysis aim at deriving the system requirements through looking at existing systems, interviews with stakeholders, workshops with different class of stakeholders with which discussing contradictions emerged among requirements, eventual development of some prototypes, and whichever activities that could help to understand the system to be specified.
3. Requirements specification is the activity of traducing the information derived from the analysis into a document that constitutes the set of requirements. The document should specify both requirements expressed at an abstract level for end-users and customers, and requirements at a deeper detailed description, namely the functionalities that the system should provide.
4. Requirements validation is the checking that the specified requirements actually define the whole system that the customer wants. This activity should discover the mistakes and correct them.

Starting from 1970's, it was recognized [14] that the high cost and poor quality of software development was a critical issue, especially on large high-technology programs of the US Department of Defence (DoD). Techniques for software development were not keeping pace with the increase in system complexity. Software Engineering, as an emerging discipline, was focusing on the more visible activities of software construction and test. However, the major cause of inadequate software, was recognized to be poor requirements definition and design with a documented useful description of the system developed and of the process used to develop it.

Yet in 1987 in [15] is recognized that the hardest part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements. This is the main cause of a wrong resulting system. No other part is as difficult to rectify later.

So until recent years, deciding precisely what to build and documenting the results was the goal of the fundamental requirements phase of software development. For many developers of large, complex software systems, requirements are their biggest software engineering problem.

Unfortunately, there had been for long time considerable disagreement on how to solve the problem.

Sommerville in [21] outlines that the activities in this requirements process are not carried out in a strict sequence. Requirements analysis continues during specification and new requirements could come during the process.

The issues of requirements process are concentrated during analysis, because:

- Stakeholders don't know what they really want.

  Stakeholders express requirements in their own languages and culture.
- Different stakeholders may have conflicting requirements and used terms.
- New stakeholders may emerge and the business environment may change.

The result of a requirements process usually has been a requirements document summarizing the objectives of the system. Other important outcomes [16] are a system test plan (since the requirements define the conditions against which the system will have to be tested) and a development plan.

Traditionally, a requirements document was a single, sequential text, but the term nowadays covers modern, more flexible formats, often appropriate for the current trendy agile methods, such as a Web site, a wiki or a cloud based collaborative document.

While Sommerville observes that in agile methods, such as eXtreme Programming (XP) and Scrum, requirements are developed incrementally according to user priorities and the elicitation of requirements comes from users who are part of the development team, Meyer in [16] is much more critical and says that the agile school rejects the idea of upfront requirements activities. The rejection is common to all agile variants (e.g. XP, Scrum, ...), because agilists view requirements documents as a form of "waste", for two reasons:

1. The **waste criticism**: a requirements document for agilists is not a useful deliverable, since it will not be part of what is given over to the customer.
2. The **change criticism**: agile correctly emphasizes on changes. It is not possible to try to freeze the requirements at the beginning of the project because the customers would change their wishes as they start seeing versions of the system, which will suggest them new ideas.

Concerning the waste criticism, Mayer affirms that it does not justify throwing away the notion of upfront written requirements, since requirements often provide a good basis for writing system documentation. In addition, software is an engineering artifact and there is no justification for renouncing the basic engineering technique of specifying what you are going to do, in writing and at the appropriate level of detail, before you do it.

In sum Meyer said that there is a middle ground between one extreme, absurdly bureaucratic of the traditional software development models, and the other, absurdly informal methods of the agile world. And he adds that comments are not strong enough. So starting any significant software project (anything beyond a couple of months and a couple of developers) without taking the time to write some basic document defining the core requirements is professional malpractice.

Concerning the Change criticism, Mayer says that no serious software engineering text advocates freezing requirements at the beginning.

The requirements document is just one of the artifacts of software development. Not only code modules and regression tests but also documentation, architecture descriptions, development plans, test plans and schedules have to be considered. In other words, Mayer says that requirements are software. Like other components of the software, requirements should be treated as an asset which can change and in practice should be put under the control of configuration management tools.

This work agrees with most of the ideas of Mayer expressed in [16], and thus the reference is often omitted.

The last International Requirements Engineering (RE) 2016, was focusing on 'Delivering Value through Better Requirements'. It refers to the need for requirements engineers to identify and communicate stakeholders' values and to constantly prioritize their work so as to deliver the most value to project stakeholders and product development teams.

Jan Bosch (Chalmers University Technology, Sweden) in its keynote in the RE1́6 remembered these famous people thinking: *If I'd asked my customers what they wanted, they'd have said a faster horse* (Henry T. Ford)

The critical failing of user interviews is that you're asking people to either remember past use or speculate on future use of a system (Jakob Nielsen)

[The assumption that a] reasonably well-defined set of requirements exists, if only we take the time to understand them, is wrong. (Dean Leffingwell)

Customers don't know what's possible. Most have no idea about the enabling technologies involved. (Marty Cagan)

You can't just ask customers what they want and then try to give that to them. By the time you get it built, they'll want something new. (Steve Jobs)

Underlying Insight of Jan Bosch:

1. Customers don't know what they want
2. You need to show it to them
3. And then measure their behavior
4. Or, if you must, talk to them

It is in these sentences the transition from waterfall model, where requirements engineering is presented as the first phase of the development process, which must be completely well defined before to pass to the development phase and then to show the results to the customer, passing from the iterative, incremental, spiral model, where prototypes would anticipate the feedback from the end user or customer, until the lean agile models, where a continuous delivery should allow to better and faster provide solutions to the customer needs, and any upfront big requirements activity is forbidden.

However the history of the software engineering doesn't finish with the philosophy of the continuous delivery, typical of the agile methodologies. In fact, the current trend in the software production understands that adding to the agile philosophy a better definition of the requirements improves the value of delivered software.

Concluding, as Mayer says "Much of software engineering is about building systems right; requirements are about building the right system"

Thus, the requirement engineerings is still in an alive evolution.

2.3 Customer vs. Provider issues

Provider is who has to develop some software for a customer who pays the provider for this activity. Providers want to gain and customer wants to have a chip software but which highly fits their needs.

In order to reach its goals, the provider has to offer quality software, by spending less possible and mustn't lose the trust of its customers. A provider must care to its reputation and expertise in building the right software for its customers.

On the other hand, to reach its goals, the customer has to find good providers with positive reputations, reliable and not too expensive. A Customer is always looking for a good trade off between costs and reliability of its providers.

Thus customers and providers have different goals and need opportune tools, strategies and methodologies to succeed.

Especially for IT complex system, which constitute the Enterprise Information System (EIS) of medium -large enterprise, the customer, that is the enterprise, must govern the complexity of its EIS which is ubiquitously invading all the business and the professional life of the enterprise and its people.

Very often, such an enterprise has many software providers which have to cooperate in building, updating and integrating new software to the whole EIS.

This introduces to the issue related to the management of what is known as "co-sourcing". It is defined, in the online BusinessDictionary (www.business dictionary.com), as *The combining of services from within and outside a business to achieve the same goal. Outsourcing can fill in gaps in internal expertise and save businesses time, money, and effort in recruiting additional staff. For example, an organization might develop software partly in-house and outsource part of the development to an outside organization.*"

However, co-sourcing implies organization and tools for supporting contracts, subcontracts as well as coordinated outsourcing [18] activities, developments and deliveries of the software.

Interesting is what Deloitte LLP in[18] says about outsourcing: *"Making thoughtful decisions on whether or not to outsource, including what elements of the enterprise should be retained or included in the initiative, and setting expectations with clear requirements can set the stage for the duration of the relationship. This includes defining the terms of the relationship as strategic, value-based or transactional, and incorporating the appropriate value measures into vendor selection criteria and SLAs. It also includes evaluating service delivery models and locations, agreeing upon audit rights, and determining how to manage risks, including those around regulations, legislation, cyber security, and data privacy".*

2.4 IT Business Management

Both customers and provider have its IT division which support ubiquitously all its business. The IT is a Business division which needs to be well managed. In fact, nowadays, it governs the Enterprise with its core business. A good management of the IT complex system [19] is a key factor to succeed in the core business.

In particular, project management, program management and portfolio management of software projects must be carefully supported, monitored and under the control of the enterprise top managers.

Specifically, in this context: *Project Management* refers to the supervision and coordination of a set of activities carried out to produce a software product or service. This includes ranking activities, managing resources involved (both humans and costs), and to reach objectives successfully, by keeping them on budget and within the time planned.

Program Management is the management of inter-related software projects that should be done following a specific road-map for reaching an overall objective. The scope of a program is likely to be wider than a project. The goal to which the program was set up is reached when all the projects are successfully completed.

Portfolio Management is a set of software projects or programs, grouped to ensure effective management aimed to strategic business objectives. Very often each division in an enterprise has its own portfolio, even thought this couldn't be the best organization to create value addition for the involved stakeholders. According to Project Management Institute (PMI) [20], organizations that manage projects in portfolios have a better Return of Investment (ROI).

However, all these concepts should allow to supervise the whole EIS from many point of view:

- costs,
- effectiveness in supporting the Enterprise businesses,
- supported and unsupported needs,
- state of its innovation and margins for its improvements,
- comprehensive list of all the subsystems composing the EIS, with their architectures, boundaries inside the EIS, in term of its gateways to other subsystems, the environments where they run (i.e. hardware, software, location), their providers and corresponding contracts
- model of the overall covered requirements with links to the related systems
- plans of developments, maintenances and evolutions
- set of tools fo supporting the supervision

This work tries to move some steps ahead in this uneasy ecosystem.

3

State of the art solutions and current trends

Historically, the activity of developing software started with projects aimed to develop simple software programs that became standalone applications helping the user in some of its activities.

For a simple project with a tiny standalone application as result, there was very simple software process to follow: Understand the requirements, think to a flowchart for answering to user needs, start to develop the program with the assembler language first and then with a high level language, starting from the instruction 'begin' and finishing with the instruction 'end'.

As long as the developed programs became more complex, with functions, modules, components, classes, distributions, services, web applications, and so on until cloud-based developments and distributed common services, the necessity for an engineered discipline of how to develop the software (i.e. Software Engineering) has been evolving from waterfall models to the current lean agile approaches.

The motivations for this evolution are interesting: at the beginning, simple programs needed a simple process where requirements, design, develop, test and delivery were the main activities to be followed in sequence, as in the initial waterfall model. Along the time, the software developed became always more complex, with many subparts to be integrated in a whole application. Thus, the need for a better software development process, which should have had to engineer each phase of the process, became necessary. In this time, it happened that the waterfall model was, on one hand extended to cover also, management activities as project plan, costs, resources and, on the other hand, operational activities as quality assurance and some planned deliveries of the application. At that time, each phase became always more formal and big upfront activities for documents writing for establishing requirements' details, design and final test plans, were introduced in the software development processes. All these documents had to be accepted by the customer before starting the development phases.

At some time it was clear that this kind of sequential steps were inadequate, because of the final dissatisfaction of the customer who had payed for having an application not suitable for its needs. Thus, the software development lifecycle (SDLC) started to evolve through introducing iterations and increments, with the aim to divide software projects long-lasting in smaller sub-projects, each one aimed to provide partial results (i.e. prototypes), to be possibly evaluated by the customer who may decide to going ahead or to modifying or even stopping the project, while avoiding friction with the provider and waste of time and money.

The major issue was the contract between the customer and the provider, because the customer wanted to know the whole project cost to be put in the contract while the provider wanted to well evaluated the amount of needed work to be done for the project, avoiding underestimated efforts. This issue brought to big analysis activity for establishing the requirements to be satisfied by the developed software and the contract with the customer started to put specific milestones where the contract could either be stopped or go ahead.

However, the customer was often not satisfied and the provider did not stay almost never in the planned efforts.

Agile methodologies [22] have understood that the customer doesn't know at contract level what their needs actually are, so one of the best known lean agile principle is both to decide what to do as later as possible, because conditions will change, and to accept any requirement change asked from the customer. Hence, agile methods start develop from some nebulous but basically central requirements, called 'user stories' and try to develop and deliver to the customer a partial application to be evaluated by the final users in its production environment. Customer could ask for whichever change. As an example, in Scrum method, a product owner defines user stories and a development team, supported by a Scrum master, chooses a small set of user stories to be developed and delivered to the customer. Feedbacks are welcomed and new or modified user stories are added to the project. This chain continues until a final state, with a final software delivery, is reached.

Agile seems marvelous, as it substitutes the possibility to shout only one arrow in a shooting gallery, where the risk to fail the center is high, with the possibility to bring the arrow with an hand and go ahead until reaching the center of the target, and, indeed, this case is very difficult to fail! However,

- this requires a big change in the type of contract between provider and customer, as it is not possible to guess at contract time how long the whole software process takes. So it is necessary that the customer, either does not fix the elapsed time for the whole project but decides to pay for a team working for a established period of time (e.g. a year or some months, which, also, has an easy to calculate cost) or to establish a precise set of user stories to be developed by the provider as subpart of the whole application and, after having tested the suitability with its needs, then decides if to order the development of additional user stories, including those coming from required changes, or to stop working with this provider. In any case, the approach to the contract and the culture of the customer

as well as that of the provider, must completely change with respect to the past.
- these agile methods have a high risk to lose the all requirements which have been satisfied by the final application, as user stories don't substitute at all a sound requirements analysis phase, because are fragmented, subject to changes often not well maintained, and so on. Very often the only available documentation is the code of the implemented methods.

Concluding, when applied in both customer side and provider side, agile methods have demonstrated their effectiveness in having a customer and provider satisfaction, but they require a big cultural change.

In addition, this thesis work claims that every software project, typically done by different teams and providers, has as its context the EIS of the customer and affects the EIS with extensions, integrations and generally evolutions, often not governed at all. This determines a not controlled state of what the EIS is: what needs it supports, the knowledge of its specific set of applications actually running and, for each one, where it is running (i.e. location, hw/sw resources), the specific needs that are satisfied, for whom it is used, and so on. This chaotic state of the EIS doesn't allow to govern innovation and the improvement to the support of the core business of the Enterprise, with unthinkable and unmeasurable consequences.

Application Lifecycle Model (ALM) is going in the direction to govern at least the set of applications that coexist in the IT department of an enterprise, but this is not sufficient because the interdependencies, distributions and locations among applications are not considered at all. A new approach, called ResDevOps, is presented in the next chapter, useful in governing the whole EIS. ResDevOps doesn't see the EIS as set of standalone subsystems, but as big complex system (both hw and sw) that evolves along the time and needs to live as long as possible because it is costly and vital for the Enterprise itself.

This chapter presents in Section 3.1 the most widespread development process introduced by IBM, named RUP (Rational Uniform Process), then the main known agile methods and how IBM has been conforming them to the Agile trend with the AUP (Agile Uniform Process); then the DevOps philosophy which is currently welcomed by customer and pursued by many agile method supporters, is shown in Section 3.2. In Section 3.3 the most famous goal-oriented methodologies and approaches are presented as the current trend for requirements engineering; some frameworks adopted and chased by famous branded enterprises, as SAFe (Scaled Agile Framework) and DAD (Disciplined Agile Delivery) are discussed in Section 3.4. Finally, main concepts and useful approaches as Application Lifecycle Management (ALM) for managing IT business strategies, are presented in Section 3.5.

3.1 RUP, agile methods and AUP

This section describes the available and most used solutions currently applied in the Industry for governing the software development process.

3.1.1 Rational Uniform Process

(RUP) model [7] is one of the most famous and applied model, devised in IBM, for developing application and especially complex applications. In fact, it has been used in Component-based applications and Service Oriented Architecture (SOA) applications. After the success of the agile methodologies, which are adaptive rather than predictive as explained in the previous chapter, RUP is yet the most applied process for IT systems predictive in nature, as is in System Engineering for critical IT system and in general systems of systems (SoS) that need to assemble predefined building blocks (e.g. aircraft systems, naval systems, and , in general, whichever real time control system). Figure 3.1 shows this model. It is an iteration model, divided in 4 phases: Inception, where mainly business is modeled and, through an initial requirements engineering and a small analysis and design activities, as well as a software factory with its environments initial configuration, a project plan and then a contract can be established. After this Inception phase, the Elaboration phase might start. Usually RUP suggests two iterations in the Elaboration phase, where requirements gathering, analysis and design are mainly performed, some implementation activities might be done, usually as proof of concepts, and testing and deployment activities are established as well. Project management and environment configuration activities, already started in the Inception phase, will continue along all the RUP phases. The third phase is Construction, usually divided in three or more iterations, where implementation,test, deployment as well as configuration and change management, are the most important activities, while requirements, analysis and design should be limited activities. The last phase is called Transition and consist in the deployment, installation and operation in the target environment of the customer. Usually, two iterations would be enough to complete this phase. RUP has been very popular and many industries are using it. With the crown popularity and success of the agile methodologies, it has been substituted by AUP (Agile Uniform Process)

3.1.2 Agile Uniform Process

(AUP) [8], is an iterative and incremental process, created by Scott Ambler working for IBM, that combined the Rational Unified Process (RUP) with agile modeling [9]. AUP tried to include the principles described in the Agile manifesto [10] that is to give value to: "Individuals and interactions over processes and tools, Working software over comprehensive documentation,

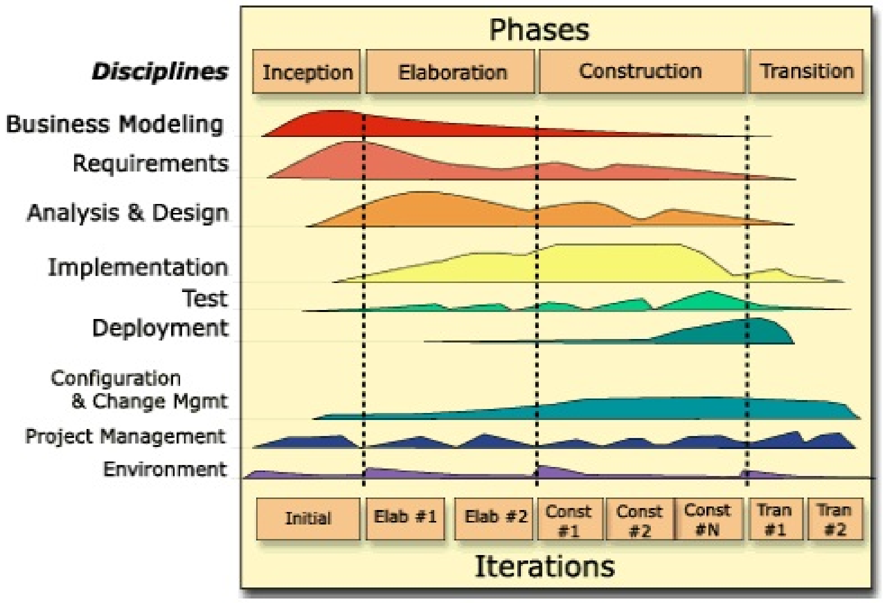


Fig. 3.1. Rational Uniform Process (RUP) model

Customer collaboration over contract negotiation, Responding to change over following a plan".

Figure 3.2 shows the AUP by Ambler [11]. With respect to RUP, AUP is a simplified version: It describes a simple, easy to understand approach to develop software using agile techniques and concepts, yet it is similar to RUP.

Specifically, the Model activity groups Business Modeling, Requirements, and Analysis and Design activities in RUP. So Model is an important part of the AUP but, in an agile view, creating models and documents are less heavy and mandatory. In addition, the Configuration & Change Management activity becomes only the Configuration Management activity, as in the agile development the change management activities are typically part of the requirements management activity, here included in the Model activity.

3.1.3 Lean Agile methods

Lean Agile software development is an umbrella term for several software development methods, including Extreme Programming, Scrum and Kanban, that were developed in the 1990s. These methods share a common philosophy described as values and principles in the Manifesto for Agile Software Development [10].

Agile methods have been a response to the drastic degree of change in the modern business and IT environments, indeed always more dynamic and

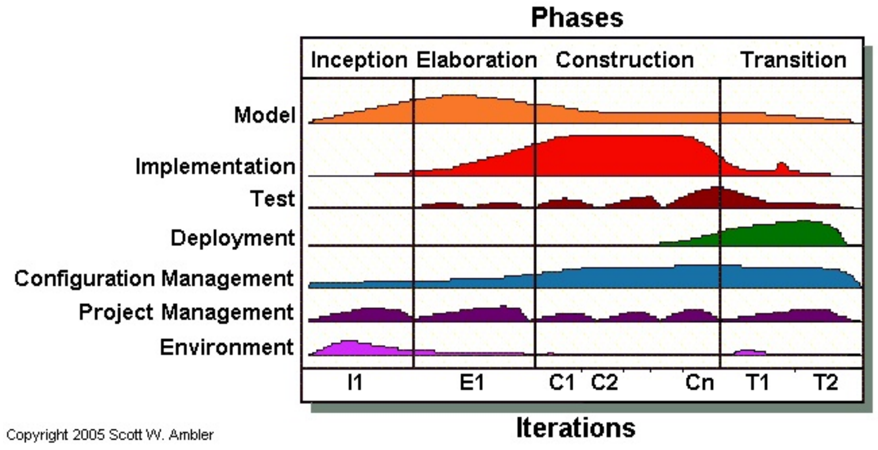


Fig. 3.2. Agile Uniform Process (AUP) model

demanding software development teams that can respond to change and continuously deliver business value[33].

Agile methods are basically (i) focused on people, (ii) based on strict communication, (iii) always ready to adapt to changes, (iv) speedy by encouraging rapid and iterative development of the software organized in small deliveries, (v) lean by focusing on short timetables and reduced cost, and on improving quality, (vi) responsive by reacting appropriately to expected and unexpected changes, and (vii) learning by focusing on improvement during and after delivering.[34]

In sum, agile is a time-focused, iterative approach aimed to build a product step-by-step (incrementally), delivering it by smaller pieces. Main benefits are the ability to adapt and change at any step (depending on feedbacks, market conditions, business obstacles, ..) and to provide only what is asked from the customer avoiding not requested work.

In the industry [35], some successful agile methods are eXstreme Programming (XP) and Scrum.

Lean philosophy[37] was born in Japan in the mid 1950 in manufacturing industry (i.e. Toyota automotive industry) and was mainly aimed at loss reduction and sustainable production.

However, Lean software development method is due to by Mary and Tom Poppendiecks [38] in the years 2000, who related it with 7 initial Lean principles and with the Agile philosophy.

As they argued, these principles provide guidance on how to deliver software faster, better, and for less cost, all at the same time:

1. Eliminate Waste
2. Build Quality In
3. Create Knowledge

4. Defer Commitment
5. Deliver Fast
6. Respect People
7. Optimize the Whole

Moreover, they say that the success of many of the practices of Agile Software Development can be explained by understanding the principles of Lean Software development.

Kanban is a well used and trendy lean agile method.

The following subsections are devoted to their brief description.

3.1.4 eXtreme Programming (XP)

eXtreme Programming (XP) [36] is the original agile approach, as its introduction in the late 1990 brought agile ideas in the forefront of the software engineering.

XP is less used today and much of the agilists moved to Scrum, even though many XP principles and practices have been integrated into other used approaches, whether or not project members are aware of their provenance.

Biggest idea of XP is *Increment then simplify.*

Key principles of XP are:

- Short iterations (as in all agile methods).
- Pair programming.
- User stories for requirements, written by the customer who is part of the team.
- Refactoring.
- Open workspace.
- Collective code ownership.
- Continuous integration.
- Test driven development.

XP pushes on the collaboration and continuous communication of the team. The main planning process within XP is called the *Planning Game.* The game is a meeting for each iteration, typically once a week. The planning process is divided into Iteration planning and Release planning.

In the team the effective roles are: (i) *Developer* who, working in pair, estimates stories, defines tasks from stories, estimates tasks, writes unit tests, writes code to pass the written unit tests, performs unit testing, refactorizes, integrates continuously; (ii) *Customer*, who should know what to program (while the developer should know how to program), is in charge to write user stories, writes functional tests, sets priorities on the stories, clarifies and decides questions related to the stories; (iii) *Manager or Tracker*, who defines the rules of planning game, clarifies to the team and to the customer the rules of the planning game, monitors the planning game, manages deviations, modifies the rules if and when required, traces the time spent by each team member and evaluates with respect to their estimate; (iv) *Coach*, who knows well the XP for the project, identifies the XP practice that should resolve possible issues, remains calm even when everyone else is panicking, observes the team silently and intervenes only when there is an issue for understand and help.

In sum, XP focuses on close customer participation, short iterations and short releases,continuous refactoring. Principal attention is on requirements, design, code, unit test, integration test, system test. Usage constraint is on management practices.

3.1.5 Scrum

Scrum has come to dominate the agile scene with its organizational technique but many teams that practice it, have added concepts coming from XP on the software-specific technical side. There exists a Scrum Alliance site available at scrumalliance.org.

Biggest idea of Scrum is *Freeze requirements during short iterations.*

Key principles of Scrum are:

- Sprint planning at the beginning .
- Requirements change but not during a sprint.
- User stories for requirements.
- Daily Scrum meeting to track progress and resolve issues.
- Task board and burn down chart to assess velocity.
- Sprint review to reflect on the closed sprint and prepare the next one.

Scrum team is composed by: (i) a *Product Owner* who represents the product's stakeholders and the voice of the customer; (ii)a *Development Team* who is responsible for delivering potentially shippable increments (PSIs) of software at the end of each Sprint. A team is made up of 3 to 9 individuals who do the whole work (i.e. analysis, design, develop, test, technical communication, document, etc.) and they are self organizing, even though there may be some interaction with a project management office (PMOs); (iii) a *Scrum Master* who is accountable for removing impediments to the team in delivering. He is not a traditional team leader or project manager (i.e. he does not have people management responsibilities), but acts as a buffer between the team and any problems and he helps the Product Owner maintain the Product Backlog, he coaches the team in order to deliver high-quality features.

In conclusion, Scrum focuses on flexibility, adaptability, productivity, through small, self motivated teams. It integrates project management process to overcome deficiencies in the development process. Issues might be on requirements gathering, on coding and all testing process as they are not completely defined.

3.1.6 Kanban

Kanban literally: "Kan" means visual and "ban" means card or board. The main idea behind the Kanban method, is "to do work just in time (JIT)".

It is known as "thinking at the contrary", because instead to have uncontrolled production of basic elements to be employed, at some unknown time, to build final products, vice versa the compound elements are built when they are needed based on a cycle time which is as long as the time needed to the production chains of the products.

Inside Toyota industry was introduced a card where the employee reads what he has to do, and after, to whom he has to pass the card where he has written that he finished his job.

In software development this means that there is a queue of work that goes through a number of stages until its done. When work is completed in a stage, it goes downstream for the next stage. When someone needs new work to do, they pull from upstream.

The team member brought a card from a stage, does the work and passed the card to the next stage in a visual and defined workflow.

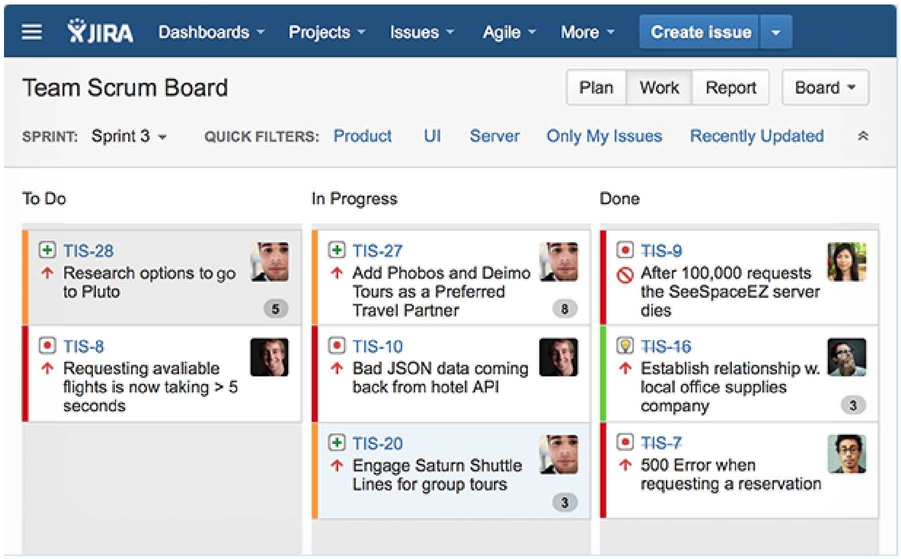
Main phases in an agile project might see a backlog queues of what to do tasks, a team member pulls out a task from the queue and he puts the task in the state 'in progress'. When he finished the task he puts the task in the state 'done'. At this point he pulls out a new upper task from the backlog.

So, a basic Kanban board has a three-step workflow: 'To Do', 'In Progress', and 'Done'. However, depending on the team size, the structure, and the objectives, the workflow can be mapped to meet the unique process of any particular team.

Kanban is useful also in conjunction with Scrum. Figure 3.3 shows an example of Atlassian tools (i.e. JIRA Agile) managing a Kanban board, by a some team members which trace their work by shifting cards trough the three previously named steps.

The main Kanban benefits [39] are:

- *Planning flexibility*: A kanban team is only focused on the work in progress. Once the team completes a work item, they start again with the item on the top of the backlog. The product owner is free to change the priorities of the work in the backlog without disrupting the team, because any changes outside the current work items don't impact the team. As long as the product owner keeps the most important work items on top of the backlog, the development team is assured they are delivering maximum value back to the business. So there's no need for the fixed-length iterations as in Scrum.
- *Shortened cycle times*: Cycle time is a key metric for Kanban teams. Cycle time is the amount of time it takes for a unit of work to travel through the team work flow, from the moment work starts to the moment it ships. By optimizing cycle time, the team can confidently forecast the delivery of future work. Overlapping skill sets lead to smaller cycle times. When



Team Scrum Board

To Do

TIS-28 Research options to go to Pluto 5

TIS-8 Requesting avaliable flights is now taking > 5 seconds

In Progress

TIS-27 Add Phobos and Deimo Tours as a Preferred Travel Partner 8

TIS-10 Bad JSON data coming back from hotel API

TIS-20 Engage Saturn Shuttle Lines for group tours 3

Done

~~TIS-9~~ After 100,000 requests the SeeSpaceEZ server dies

~~TIS-16~~ Establish relationship w. local office supplies company 3

~~TIS-7~~ 500 Error when requesting a reservation

Fig. 3.3. Kanban board built with Atlassian tools

only one person holds a skill set, that person becomes a bottleneck in the work flow. In a Kanban framework, the entire team has the responsibility to ensure work is moving smoothly through the process.

- *Fewer bottlenecks*: Multitasking kills efficiency. The more work items in flight at any given time, the more context switching, which hinders their path to completion. That's why a key tenant of Kanban is to limit the amount of work in progress (WIP). Work-in-progress limits highlight bottlenecks in the team's process due to lack of focus, people, or skill sets. A low limit encourages the team to pay special attention to its work, and to ends its work before starting a new one. This ultimately reduces the overall cycle time.
- *Visual metrics*: One of the core values is a strong focus on continuously improving team efficiency and effectiveness with every iteration of work. Charts provide a visual mechanism for teams to ensure they're continuing to improve. When the team can see data, it's easier to spot bottlenecks in the process (and remove them). Two common kanban reports used by teams are control charts and cumulative flow diagrams. A control chart shows the cycle time for each issue as well as a rolling average for the team.

3.2 DevOps

DevOps [40, 42, 44, 43, 45]is a method employed downstream the application of lean and agile processes. While lean and agile methodologies (e.g. Kanban, Scrum, XP, etc.) aim to accelerate software development and to maintain good quality levels, DevOps methods aims to deploy working functionalities into production faster.

Usually, each time the development team releases a new system version, comprehending new functionalities and bug fixes, the system is delivered and the developers' support stops there. At this point, the operation team can start the installation of the new release for the end user with no direct interaction and/or collaboration with the developers. Automatic bug reporting tools and user assistance services are the means for give feedbacks to the development process. The almost sharp separation between development and operation activities and responsibilities may lead to issues during system deployment and operation causing customer dissatisfaction.

DevOps aims to complement agile software engineering by breaking down the silos wherein development and operation are separately confined. DevOps fosters a seamless process embracing both development and operation in a cycle of shared activities and responsibilities [103, 104, 105].

Fig.3.4 illustrates how DevOps based processes organize the involved activities in a loop.

DevOps culture claims the necessity to support this loop both with automatic tools and by cooperating teams. This loop brings continuous integration, testing and delivery over an agile infrastructure, where every sprint is subject to the DevOps deployment method.

Many important IT Enterprises have adopted the DevOps culture [104, 105]. Among them, IBM, after moving RUP (Rational Unified Process) towards agile methods by introducing AUP (Agile Unified Model)[101], has recently adopted SAFe® (Scaled Agile Framework by Scaled Agile, Inc.)[102], which is an interactive knowledge base for implementing agile practices at enterprise scale.

SAFe® provides organizations with a set of best practices, artifacts and suggested tools in order to scale agile from the team level, to program/portfolio level. SAFe® includes DevOps as one of the practices it relies on.

The basic idea behind DevOps is that the support infrastructure should be able to listen to IT operations and to give suitable feedback to the development activities. Development and IT operations cooperation should share the success of every deployment in the operating environment, by the availability of a hybrid team working for every bug correction and requirement tuning.

The DevOps loop in Fig.3.4, includes a specific activity, i.e. Plan, where the next release is planned, without any explicit negotiation phase. Nevertheless, business managers should authorize indeed a new release, because it needs, at least, a scope and budget evaluation, but it should require a business conscious decision. Often this is underestimated in terms of needed time, resources and even impact on the overall EIS requirements, but it is not taken into account by the DevOps approach. The business decision is very important for the provider of the software development because of the risk of over budget and/or

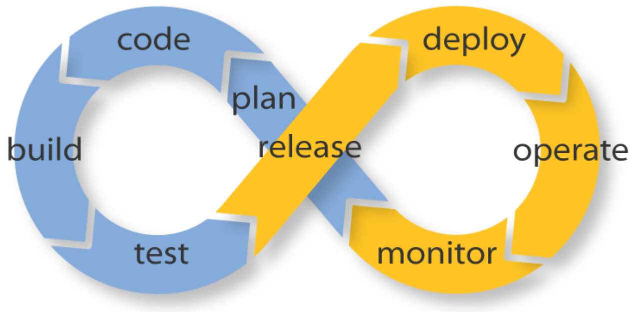


Fig. 3.4. The DevOps loop

out of scope, which might have a negative impact on the compliance with the customer agreed contract.

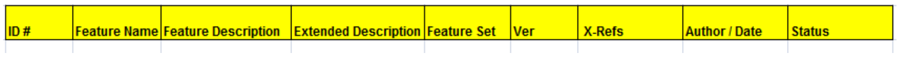
Then, even though agile methodologies afford these issues by applying the useful strategy to reduce the amount of promised functionalities and/or time established inside the contract with the customer, the achievement of customer satisfaction with a continuous change of requirements might lead to unexpected over budget and out scope issues, dangerous if not timely and properly managed.

It is worth to note that the SAFe® framework addresses and reduces the over budget risk, by introducing the interesting concept of ”agile release train”, i.e. ”a long-lived, self-organized team of Agile Teams” [102] bounded to a program, instead to a project, and provides for it a budgeting field that is under the responsibility of the Program Portfolio Management (PPM).

3.3 RE: Goal oriented methodologies

Traditional requirements engineering methods are devoted to elicit *what* the target application, to be developed, should do, with the aim to satisfy the customer needs. Many best practices, languages and tools have been provided to establish, verify, validate and trace evolution of requirements. Requirements have been represented as *features*, that is which capabilities the final application should have. Features have been pulled out by means of interviews to the users, brainstormings among stakeholders, prototypes demonstrating dummy functionalities. Specifications have been done using either natural language, or formal sheets with specific fields to be filled (i.e. 3.5) or using formal languages, even executable. Visual languages have been useful in facilitating the communication between analyst/developer and users. The most famous formal visual language for requirement is the OMG (Object Management Group) standard, named Unified Modeling Language (UML) [46].

Requirements with UML, include documents with actors, use cases and many useful diagrams like the activity diagrams, great to specify the interaction among functional components of the application. Also, use cases in UML have detailed with preconditions, triggers, main sequential and alternative steps, exception conditions, and postconditions.



ID #	Feature Name	Feature Description	Extended Description	Feature Set	Ver	X-Refs	Author / Date	Status

Fig. 3.5. An example of sheet for documenting features

According with [21], RE is related to both the user requirements (usually referred as set of features), expressing both functional and non functional capabilities of our software, and system requirements, related to the design or specification of how to implement user requirements.

Concluding, requirements engineering has been related to what capabilities should have the software to be developed, so that the user needs are satisfied.

The crisis of traditional software engineering methods, with the growing of the complexity and distribution of the systems to be built, tells us that inadequate, incomplete, ambiguous, or inconsistent traditional requirements approaches are having a significant impact on the quality of the developed software. For this reason, in the recent years, Goal Oriented Requirements Engineering (GORE) gained a lot of attention in the academia as well as in the industry [47].

Requirements with GORE approach are no more *what* the developed software has to be able to do, but what are the *goals* that the stakeholders want to reach with the developed software .

This allows to transit from an operational approach, that is ”the application should be able to”, to a goal oriented approach, that is ”I want to reach the goal of ”.

In the agile methods, this means passing from features to user stories, that is from ”the system should have the capability to” to ”As a stakeholder, I want to”.

However, it should be distinguished between early requirements (called user requirements in [21]) and late requirements (called system requirements in [21]).

In other words, there is a difference from eliciting features, as capabilities of the software, and designing a software solution, as specifying complete use cases and activity diagrams that give details on the target software.

In my opinion, while traditional RE fails when lingers about user and system requirements before passing to develop and deliver to the customer some release of the software which answers some of its needs, agile methods might fail when they use only user stories and continuous delivery approach, especially when the software to be developed is complex, involves a big number of stakeholders or must integrate with either a preexisting Information System or with the software of other software developers (in either a co-sourcing or multi teams configuration).

Nowadays software development has to take into account many interdependencies at requirements level. In addition, modeling the context where the application should operate, is extremely important.

While agile methods discussed in this chapter, have been incorporated and then enhanced in scalable software engineering frameworks aiming at supporting the IT business context model, as discussed in the next section, RE instead, is evolving from simple standalone user stories to stakeholder goals, supported by the GORE methodologies.

Thus, GORE originates from the needs to model and to understand software requirements, starting from high-level aspects of the domain of interest and to cope with non functional requirements. Various GORE approaches have been proposed, applied, compared and reviewed in literature [48, 49, 50, 52, 51]

Some notable examples of GORE follow.

The *NFR framework* [53] is a process-oriented approach that focuses on the modeling of non-functional requirements (called softgoals) and on the identification of the influences (positive or negative) among them. Softgoal refinements and influences are represented in a softgoal interdependency graph, which allows evaluating the contributions of more specific goals with respect to higher level ones and to identify and evaluate different alternatives.

The *i∗ framework* [54] is a goal-oriented modeling technique, based on an explicit representation of goals and actors, and it has been adopted in many requirement engineering contexts. The i∗ framework defines two types of model: the Strategic Dependency model and the Strategic Rationale model. The first one describes the dependencies among actors involved in a given context, where they depend on each other for goals to be achieved, tasks to be performed, and resources to be made available. The Strategic Rationale model identifies stakeholders' interests and concerns, and shows how they can be dealt with by various configurations of systems and environments. The i∗ framework is mainly concerned with the early-phase requirements engineering.

KAOS [55] is a method for deriving operational requirements of a system starting from its goals. The notion of goal, intended as a "prescriptive statement of intent that the system should satisfy through cooperation of its agents" [56], is the main concept underlying KAOS, where an agent is any entity that may influence the fulfillment of a goal. KAOS supports the definition of goals at different level of abstraction by introducing suitable refinement relations among goals.

ARMOR [57] is another goal-oriented modeling language based both on i* and on KAOS methods, trying to overcome their respective limitations. In particular, ARMOR adds support for modeling stakeholders' domain during the early requirements specification and it adopts UML use cases for the elicitation of system specific requirements.

GRL (Goal-oriented Requirement Language) is part of the URN (User Requirements Notation) (Z.151, 2012). It is a standard notation for goal modeling and simplified version of i*. GRL is supported by the jUCMNav [52], a free

Eclipse graphical editor, an analysis and transformation tool with limitations and possible improvements as discussed in [48].

3.4 Agile based Fameworks

Today, most enterprises want to move toward more agile methods. However, it is not easy to choice one or more agile approaches and to say to development teams, just to apply them. [41]

While a single team works on a project at a time, an enterprise works to programs, that is sets of several projects that may overlap. Larger projects are built by teams of teams, that may work in different physical locations. This type of work requires coordination. Managing agile projects plans, their progresses, their evaluations, scheduling, and many other business activities, are complex and need road maps and support tools that the single XP, Scrum, or whatever else agile method, does not care.

To this end, the following subsection describes two of the most famous agile frameworks: DAD and SAFe.

3.4.1 Disciplined Agile Delivery

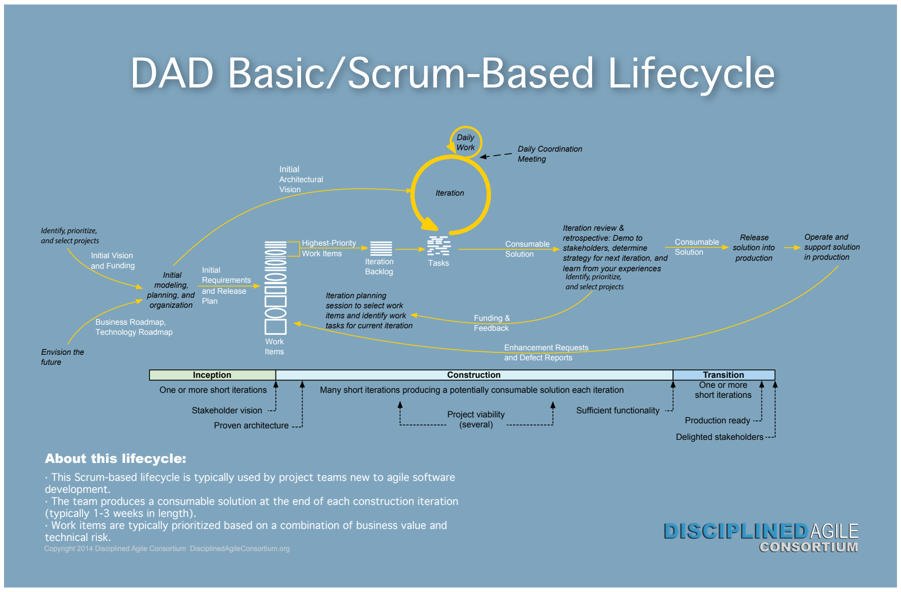
While Scrum method assumes that a team is in running, it does not say where the team starts, or how it should decide the first sprint, or when and who decides the basic platform, the software factory to be used, the programming language and, most important, the requirements, design and the architecture of the application that team should develop. Disciplined Agile Delivery (DAD) [12] framework by Scott Ambler, supports these extra development, but fundamental and highly risky activities, by including (i) an inception of the project, where the platform is decided, tools are built, project is scheduled, the architecture is defined and team is formed, and (ii) a transition at the end of the project construction, supporting the operational use of the developed software. The copyrighted figure 3.6 gives an idea of DAD.

3.4.2 Scalable Agile Framework

SAFe® [17] framework aims at providing a complete guide for enterprises that want to maximize the benefits of Lean-Agile development. Many of the largest organizations in the world have adopted SAFe, so it is mandatory to talk about.

The latest version renamed "SAFe 4.0 for Lean Software and Systems Engineering", was released in January 2016.

There are two different types of SAFe 4.0 implementations: one with 3 levels and another with 4 levels. The first one is represented in figure 3.7 and it is for small enterprises with 100 people or less, while the second one,


DAD Basic/Scrum-Based Lifecycle
Daily Work
Daily Coordination Meeting
Initial Architectural Vision
Iteration
Identify, prioritize, and select projects
Initial Vision and Funding
Initial modeling, planning, and organization
Initial Requirements and Release Plan
Highest-Priority Work Items
Iteration Backlog
Tasks
Consumable Solution
Iteration review & retrospective: Demo to stakeholders, determine strategy for next iteration, and learn from your experiences
Consumable Solution
Release solution into production
Operate and support solution in production
Identify, prioritize, and select projects
Iteration planning session to select work items and identify work tasks for current iteration
Funding & Feedback
Business Roadmap, Technology Roadmap
Envision the future
Work Items
Enhancement Requests and Defect Reports
Inception
Construction
Transition
One or more short iterations
Many short iterations producing a potentially consumable solution each iteration
One or more short iterations
Stakeholder vision
Proven architecture
Project viability (several)
Sufficient functionality
Production ready
Delighted stakeholders
DISCIPLINEDAGILE CONSORTIUM

About this lifecycle:

· This Scrum-based lifecycle is typically used by project teams new to agile software development.
· The team produces a consumable solution at the end of each construction iteration (typically 1-3 weeks in length).
· Work items are typically prioritized based on a combination of business value and technical risk.

Fig. 3.6. DAD basics framework

represented in figure 3.8, is for solutions that typically require many hundreds of people to manage, develop, deploy and maintain the enterprise information system. In addition there is a Foundation layer. The "3 levels SAFe" types are Team, Program and Portfolio, as described below:

1. Team level is fundamentally based on Agile teams. Each team is responsible for defining, building, and testing user stories from their backlog. Teams deliver value in a series of sprints or iterations. Teams use a common iteration cadence to synchronize work with other teams, allowing the entire system to iterate simultaneously. Teams employ Scrum (primarily) or Kanban methods. Many software quality best practices are derived from eXtreme Programming, while hardware and system quality practices are derived from contemporary Lean product development practices.
2. Program level sees teams organized into a virtual program structure called the "Agile Release Train" (ART). Each ART is a long-lived, self-organizing team of Agile teams (typically 5 to 12), along with other stakeholders, that plan, commit, execute, inspect, and adapt together. ARTs are organized around the principle value streams of the enterprise. They align teams to a common mission, provide architectural and user experience guidance, facilitate flow, and provide continuous objective evidence of progress.
3. Portfolio level organizes and funds a set of value streams. The value streams realize a set of solutions, aimed at helping the enterprise to achieve its strategic mission, as defined in part, by a set of strategic themes. The Portfolio level provides solutions development funding via Lean-Agile bud-

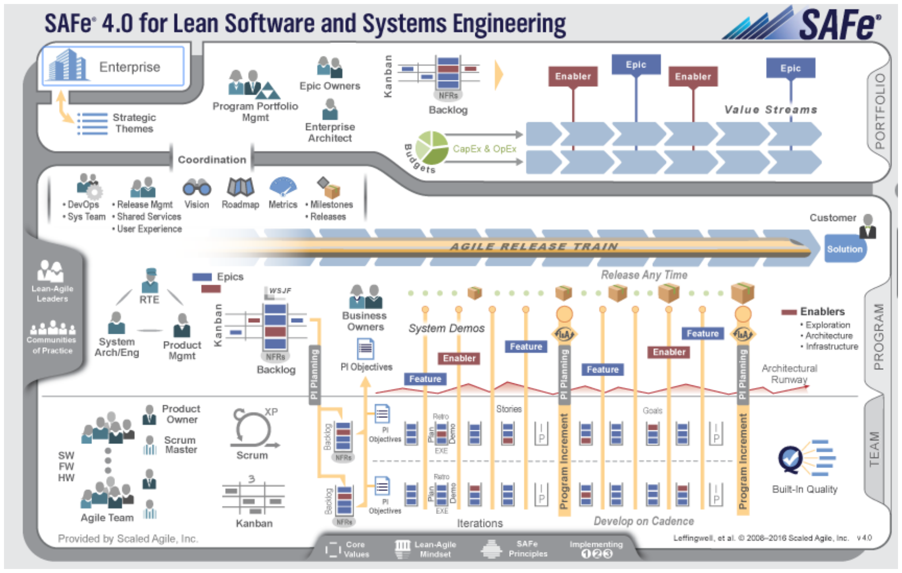


Fig. 3.7. 3 levels SAFe

geting, any necessary governance, and coordination of larger development initiatives that affect multiple value streams.

Value Stream level is the 4th level to add to the previous 3 levels for the ”4 level SAFe”. This level is optional and supports the development of large and complex solutions. These solutions require multiple, synchronized ARTs, as well as stronger focus on solution intent and solution context. Suppliers and additional stakeholders contribute to this level as well. Pre and Post Program Increment (PI) planning inform the ARTs (and vice versa) of the Value Stream mission and objectives.

Finally, the Foundation layer holds various additional elements that support development, as Lean-Agile Leaders, Communities of Practice, Core Values, Lean-Agile Mindset, and SAFe Principles.

3.5 IT business strategies

As previously discussed, both agile methods and traditional software lifecycle, concentrate on the activities strictly related to the development and delivery of the application. Frameworks like DAD and SAFe, were born to answer to the Enterprise need to manage the complexity of the always more complex, ubiquitous and pervasive EIS.

In fact, software projects have to be planned and managed to avoid risks as over budget (i.e. uncontrolled cost might grow until an insupportable surplus may cause serious problems for the Enterprise survival) and out of scope

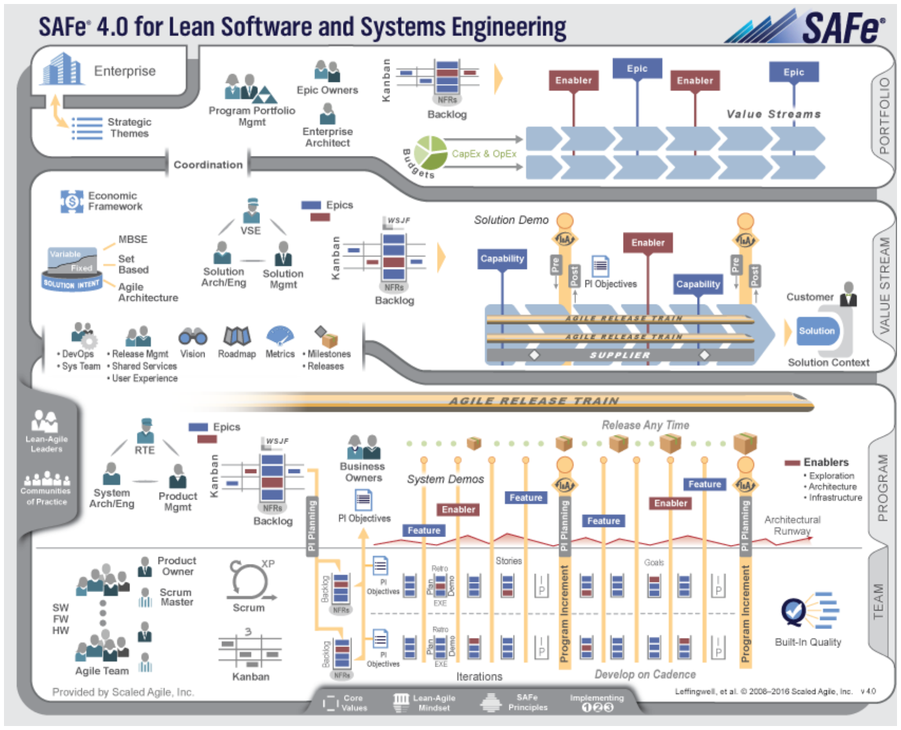


Fig. 3.8. 4 levels SAFe

(i.e. the software project was born for satisfying some business need but it fails because the resulting application is not useful at all for the Enterprise). In addition, every Enterprise should have some road maps for its needs, at least at middle term IT support. This implies to have programs constituted by a set of scheduled IT projects. They start from an initial IT supported business activities and they guide to a final state where the evolved business has its adequate IT support. Finally programs are organized in portfolio of programs or projects. They, trough being independent each other, share things as responsibilities, or budget, or resources, or divisions, or several of these and other things.

3.5.1 Agile Lifecycle Management (ALM)

While DAD and SAFe introduce IT management concepts of Portfolios, Programs and Projects, they are considered frameworks for supporting the software development of an Enterprise.

IT department in an Enterprise is thus recognized fundamental for the Business of the Enterprise.

However, as software life cycles address only the software construction, these frameworks address only the management activities needed to the software development activities, while do not consider the management activities of the applications running in the EIS.

To this end, in addition to the software devevopment lifecycles (SDLC), the application lifecycle management (ALM) was born[58].

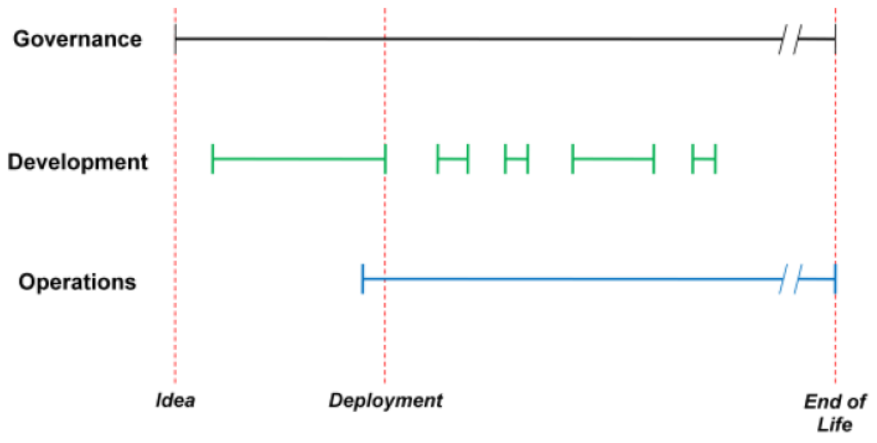


Fig. 3.9. ALM three sectors

ALM includes SDLC, as it considers the entire time during an enterprise is spending money on the software development or IT support or, generally, the application asset, from the initial idea to the end of the usage (also call retirement in RUP) of the application.

ALM can be divided into three different sectors or lines: governance, development, and operations. Figure 3.9 graphycally illustrates the concepts of ALM, showing each of these three aspects on its horizontal lines.

Governance includes all of the decision making and project management over the application entire lifecycle. Governance is the most important aspect of ALM, as the business value depends on it. In fact, the aim of governance is to manage the application suitability to the actual business needs.

Figure 3.10 shows a detailed view of the governance: first of all business case development precedes the application development process and aims to define and agree on a business case at the enterprise level. During the application development the project portfolio management (PPM) aims to control costs, resources and milestone deliverables, looking for a successful development of the application. PPM, depending on the dimension and organization of the enterprise, could be done by a project manager added to the development team (in very simple cases one member of the development team could have also the role of project manager). In other enterprises, project manage-

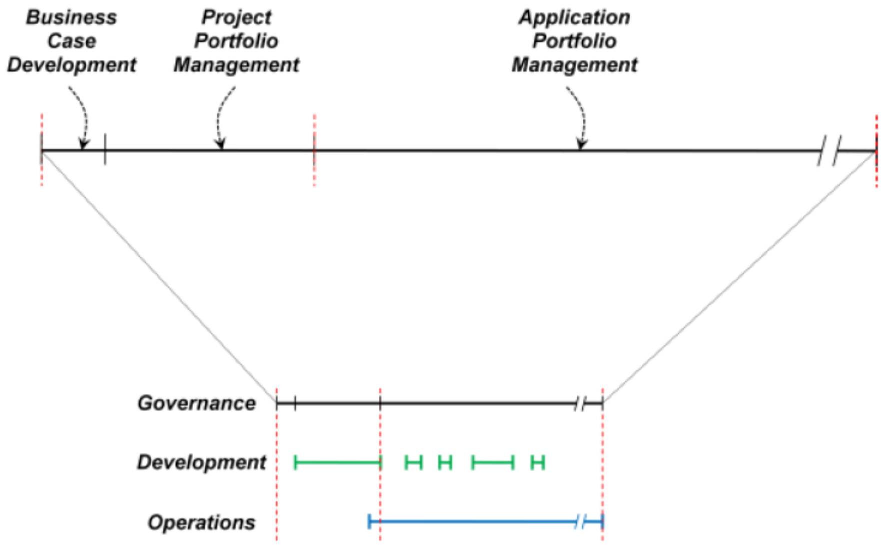

Business Case Development
Project Portfolio Management
Application Portfolio Management
Governance
Development
Operations

Fig. 3.10. ALM Governance

ment could be centered in a IT department, especially when best practices and formal procedures must be used. After the deployment of the application, the governance sector needs to add this application to the enterprise portfolio of applications of the overall EIS. Every application is an asset of the enterprise, and the Application Portfolio Management (APM) is in charge to manage its life, improvements, costs and benefits for the enterprise business. Every approved improvement is a new project with its governance line.

Development is the process of actually developing the application, from the idea to the deployment. This process could be iterative both to incrementally develop the application and to improve it with new versions.

Figure 3.11 displays details of this aspect of ALM. Once the business case is approved, the software development lifecycle begins. Expanding the SDLC parts of the Development line shown in the figure, a modern process would probably show software development as a series of iterations. Each iteration would contain some requirements definition, some design, some development, and some testing. This iterative style of development could not be always the chosen lyfecycle, as some projects are still preferring more traditional lyfecycle, even though nowadays iterative methods, like the agile method, are the current trend in many enterprises.

Once the SDLC process for a first version of the application is complete, the application is deployed. This is not the end of development, because the application might needs some full SDLC efforts to create new versions and continuous updates for new requirements or improvements to better support the business, as shown in the figure 3.12.

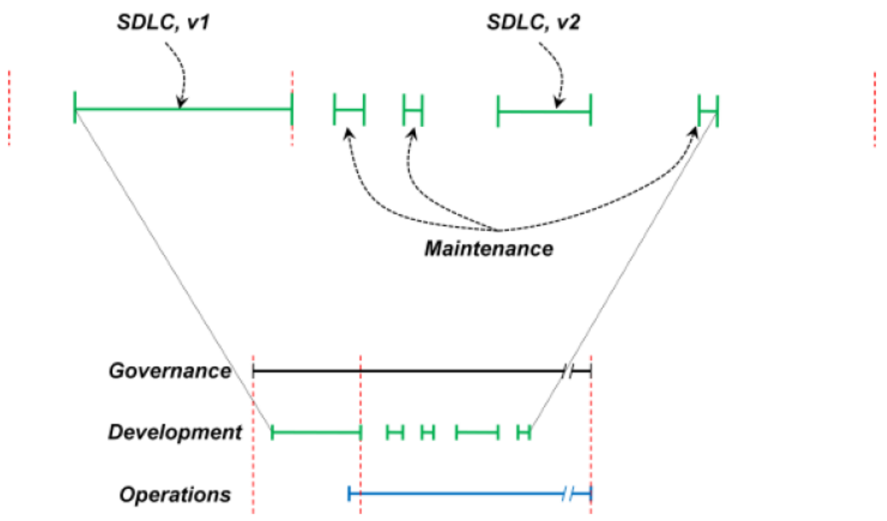


Fig. 3.11. ALM Development

Operations, the work required to install, maintain in execution and manage the application in the target environment. Typically it begins with the deployment and go on continuously until the application is retired. Every deployed application must be monitored and managed during its lifetime in production. Figure 3.12 shows some of the important parts in the operations process. As with Governance, the Operations line is strongly connected to the Development line. In other words, the deployment planning begins just before the application is completed, and the deployment activity is a fundamental part of operations. Once the application is deployed, it must be continuously monitored. The monitoring could result in the necessity to govern some update to the application, which requires new development and then new deployments in operations, as the figure shows.

ALM is the current trend on software engineering. Its value is in (i) including SDLC of whichever kind, that is from waterfall to agile, (ii) including the management of deployments and operations, indeed the main objectives of DevOps, and(iii) it foresee frameworks as DAD and SAFe, which include all the enterprise IT business.

Concluding, the major value of ALM is to allow whichever IT business strategy, which has to cope with both the development projects answering to the business needs and to the numerous applications composing the EIS operating in the enterprise and needing continuous support, improvements and updates.

To this end, it is necessary to have a strategy, distributed or centralized, for managing every projects (PM), every programs (i.e. set of projects with a road map to be followed for reaching a specific business support) and every

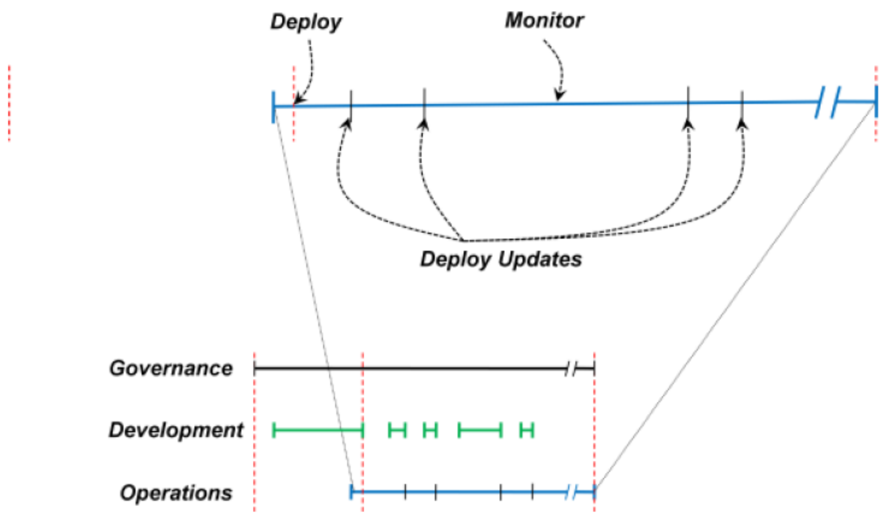


Fig. 3.12. ALM Operations

portfolio of projects (PPM) managed by specific managers with a given budget and fitting the enterprise business organization.

A number of vendors today provide tools that are horizontally integrated on one of the three lines.

For example, Microsoft Visual Studio Team System offers a set of tools supporting several aspects of the development process. However tools should be integrated not only horizontally but also vertically as well, helping enterprises in building and maintaining connections across the three lines. For instance, project management tools should be connected to development tools, which in turn should have connections to tools used for operations.

Today DAD and SAFe last releases suggest their frameworks where PM and PPM are included, while programs are currently managed with release trains only in SAFe. They also include whichever agile methods and DevOps method to support the operation line of ALM.

Visul Studio ALM

Visual Studio Application Lifecycle Management (ALM) is a collection of integrated software development tools developed by Microsoft. These tools currently consist of the IDE (Visual Studio 2015 Community and greater editions), server (Team Foundation Server), and cloud services (Visual Studio Team Services). Visual Studio ALM supports team-based development and collaboration, Agile project management, DevOps, source control, packaging, continuous development, automated testing, release management, continuous delivery, and reporting tools for apps and services.[60, 59]

Atlassian ALM

Atlassian [61] is another ALM vendor. Its software suite works as Software as a Service (SaaS) running on Windows and Linux and using only Java. Initially, it was born for supporting development teams, while nowadays the suite supports all three ALM lines. It is adaptable to small-scale projects like startups as well as major corporate enterprises. The Atlassian suite of ALM software is composed of six individual products, all of them are integrated with each another. In particular:

1. JIRA, the engine for tasks and projects management, lets users track issues or tasks through a predefined, customizable workflow. Tasks can be organized by project, allowing an enterprise to transparently track issues within projects. The platform features both native workflow options and tools to customize workflows based on users' individual needs. Several add-ons can be added to JIRA, including *JIRA Agile and JIRA Portfolio*. Its most recent update, include native agile tools and it optimizes performance for granular tracing and UI usability.
2. The Atlassian suite includes two communication tools: *Confluence*, a wiki allowing users to share, view and discuss meeting notes, files, links and messages, and to assign tasks to individuals or groups, and a tool for immediate communication named *HipChat*, a real-time messaging tool. These two products integrated with JIRA, allow to facilitating mass communication within an enterprise.
3. *BitBucket* and *Stash* are Atlassian's development tools for managing code and for creating and managing code repositories. They enable collaboration and integration with JIRA. Bitbucket is the cloud code manager while Stash is the server or on-premises code manager.

Many other vendors exist. However, there is still plenty of room for improvement towards ALM tools that fully integrate vertically all the three lines.

4 Requirements Engineering: GOReM

According to Meyer, requirements engineering activity is fundamental for achieving the right application, while often software engineers concentrate on developing applications in the right way, failing in the primary objective, that is to well satisfy the user needs.

Requirements Engineering, as explained in the previous chapter, covers early and late tasks such as concept exploration, requirements specification and analysis. Also, verification and validation activities as well as development activity, are contemplated after each iteration and incremental design. All these activities have the utmost importance in the development life-cycle of complex systems, especially when the business domain is unfamiliar to the designers and when numerous partners are jointly involved in a project. For example, in big industrial research projects, funded by government institutions, or in medium - large, often distributed Enterprise, where many departments are involved in a continuous development activity that evolves their complex information system. A critical role is then played by the characteristics of the methods, notations and tools adopted for carrying out the activities related to requirements. As discussed in the previous chapter, goal oriented requirements engineering (GORE) is a method widely accepted and it is the current trend in the area.

This Chapter presents a new GORE methodology, devised at University of Calabria, named Goal Oriented Requirements Methodology (GOReM).

GOReM addresses requirements engineering phases in a lean, easy to master and to apply, as well as clear and rigorous method. The methodology is not the result of a pure speculative process, but has been concretely shaped in the context of some real large research project where it has shown its effectiveness in fostering cooperation and clean project evolution.

In this chapter the methodology is defined and motivated, while some case studies from real projects centered on different domains as Tourism, Cloud and Systemic Risk, which have been also appreciated in some important international conference, are presented.

4.1 Introduction to GOReM: yet another requirements methodology

Large projects, such as those funded by government institutions, usually involve many partners that need to work concurrently and to cooperate among them to achieve project goals in an orderly and timely way. The organization and management of such projects are complex activities that are critical for a successful achievement of project results. Many are the risk sources: coexistence of different interpretations of project goals and requirements, conflicting specifications, late discovery of redundancy, fragmentation of efforts, weak focus on objectives, partner coordination issues, and work-product integration. In such complex and heterogeneous settings, a crucial role is played by the adopted methodologies, in particular during the requirements engineering phases [21, 54]. Most of the available modeling techniques are mainly focused on the identification, analysis and specification of the requirements that must be satisfied by the functionalities that are going to be realized. Often, little or no attention is paid on the motivations that lead stakeholders to demand for such features and this may result in late discovery of incompleteness and inconsistencies due to conflicting goals and needs. With the aim to overcome these issues, in the last few years, several goal-oriented methodologies, already discussed in the previous chapter, have been proposed [55, 57, 62] and limitations of the related existing tools have been addressed [48]. However, only a few methodologies support both early and late requirements engineering phases, most of them adopt proprietary or very formal notations and languages [63] and they are difficult to integrate with the methodologies used in the downstream development process. In large research industrial projects with many partners and stakeholders, only one or some of them are industrial enterprises playing the role of target user whose requirements have to be addressed, while many of the partners are researchers and/or experts in specific areas, although precious for the reference application domain. In this case, the adoption of one of those methodologies and related tools do not facilitate communication and convergence of intents in mastering concept exploration. Thus, the project will end with important research results for every research partner in its specific area of interest, but with poor industrial applicable results for the target users. Incomplete or complex language notation is often the origin of an insufficient domain concept exploration and, then, of an unclear and inconsistent project requirement elicitation phase[64]. As a consequence, misunderstandings in the way of achieving goals in the specific industrial contexts of the target users are indeed inevitable[65].

A novel, lean and easy to adopt approach that aims at addressing the above introduced issues in large industrial research projects is presented. It is a Goal-Oriented Requirements Methodology (GOReM) for mastering concept exploration, which seamlessly supports all the stages of requirements engineering. GOReM is structured in three main phases (i.e. Context modeling, Scenario Modeling and Use Case modeling) that lead to requirements specification analysis starting from stakeholders' goals. Key aspects of GOReM are the following: (i) clear understanding of the context where the system under definition is going to operate in terms of governing rules, stakeholders and their goals; (ii) modeling of business scenarios and analysis of the strengths, weaknesses, opportunities and threats of each business scenario; (iii) use case and process-based modeling of application scenarios. As a consequence, GOReM enables global/analytical views of the application to be developed and of its context at different abstraction levels. This provides a solid ground for partners' cooperation, efforts harmonization and outcomes validation. Moreover, GOReM is based on a lean process and on an UML-based notations, that is a standard of the Object Management Group (OMG) [46], having a smooth learning curve and easing the integration and reuse of the released work-products for the subsequent design and development phases [114]. GOReM derives from the needs and criticalities arising in the context of real projects. Indeed, the definition of GOReM has been carried out starting in a real large research project - DICET-INMOTO[66], funded by the Italian Ministry of Education, University and Research (MIUR), where it showed its effectiveness in fostering cooperation and clean project evolution.

GOReM aims to capitalize the Goal oriented experiences mentioned in the previous chapter by also resorting to concepts, abstractions, and methods coming from the AOSE (Agent-Oriented Software Engineering) domain [68] and, specifically, derived from both the Tropos [69] and Gaia [70] methodologies. Moreover, it exploits an UML-based graphical notation, so to produce a documentation both expressive and easy to read, embracing the thesis in [71] where visual notation enhances user comprehension of requirements (i.e. cooperation and sharing) and it allows to move towards mastering concept exploration, at least for 'known unknowns' as defined in [50] (i.e., expressing what is behind stakeholders' expressions as "obviously" and "of course") which is a typical situation in large industrial research projects.

In conclusion, while existing GORE methods often define ad-hoc languages for expressing requirements, GOReM uses UML, visual language, known at many and easy to understand, allowing minimization of the time to learn, harmonization in the collaboration among very diverse partners and tracing of the goal reaching by following "who do what and why".

In addition, existing GORE methods put a precise distinction between goals that the stakeholders want to reach (i.e. needs) and functional processes thought to reach the goals (i.e. design of solutions). GOReM instead, even though puts a distinction between context and scenario model from one side, and application model on the other side, really use cases in the application model have only the aim to defining "what" (i.e. early and late requirements) and not "how" (i.e. specific steps), given a continuous but different direction to the design phase of a software solution.

Finally, GOReM gives considerable attention to rules and regulations governing a context and each business scenario. This is particularly important in the Era of decentralization, Internet and Cloud computing.

4.2 GOReM overview

In this section the GOReM Methodology is described by first explaining a reference meta-model that highlights the performed activities and the relationships among them. Then the GOReM reference process together with its work-products are presented. The GOReM Methodology consists of three specific phases of modeling activities:

1. Context Modeling, where the system stakeholders are identified along with their goals as well as the relationships and dependencies between stakeholders and goals; moreover, the rules and norms that govern the context are specified.
2. Scenario Modeling, where different business scenarios are specified in terms of roles played by the stakeholders involved in the scenario, their specific goals, and the rules and norms that govern the business scenario. A SWOT Analysis (Strengths, Weaknesses, Opportunities and Threats) is also performed [73] with the aim to guide future work decisions.
3. Use Case Modeling, where application scenarios are introduced in order to fully specify the functionalities which should be provided by each business scenario resulting from the previous phase.

Figure 4.1 shows an UML based representation of the reference meta-model and all the involved entities and the associations among them. The three different colors used in the schema are useful to easily identify the corresponding three GOReM phases. A detail of each single phase is described in the following. The Context Modeling phase aims at delimiting the project scope within which the requirements should find precise boundaries. Context Modeling can be related to a large partnership that would like either to participate to a public call for proposal or to realize the starting phase of an industrial research project. This crucial activity has to be carried out by a (small set of) designer(s) who has to identify main elements useful to fix a clear, unique, specific and shared project scope. In the Context Modeling phase, the set of stakeholders (i.e., Stakeholder in Figure 4.1) and whatever their specialization (i.e., specialize association) should be identified. Each stakeholder is in turn interested in pursuing a set of softgoals (i.e., Softgoal), a common concept in the goal-oriented community for denoting generic goals and often non-functional requirements. The GOReM methodology usually identifies the following relationships among softgoals: decompose, contribute, hinder and specialize, but, if needed, other association stereotypes can be also added [115]. Moreover, during this phase, it is important to clearly identify and carefully take into account the set of Rule and Regulations, governing the specific context, as they (and their possible changes) can influence the achievement of stakeholderś goals.

The Scenario Modeling phase aims at specializing each context in many specific scenarios (i.e., Scenario). The decision process able to establish the set of scenarios covering the modeled context, is not a formal process, but it is the

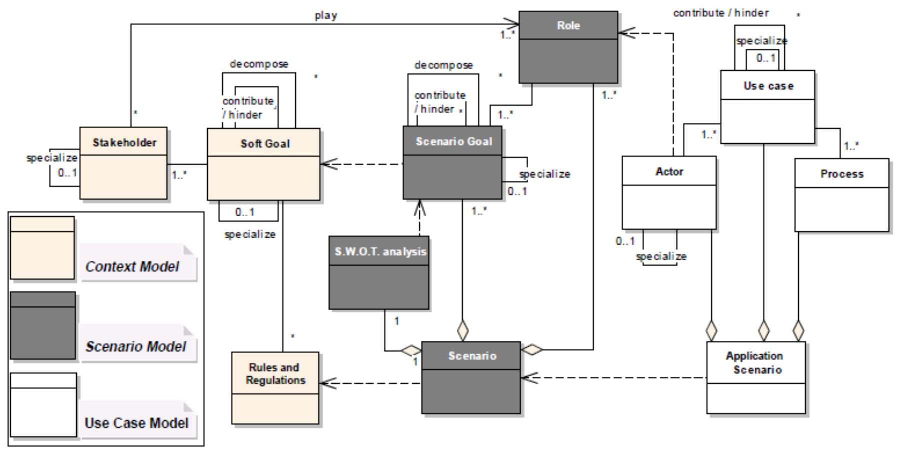


Fig. 4.1. GOReM meta model

result of several interactions among project partners based on the results of the Context Modeling phase. Each scenario should be modeled by a (possible small set of) designer(s) specialized in the specific scenario domain. Starting from the stakeholders identified in the previous Context Modeling phase, a set of specific stakeholder roles (i.e., Role) is identified for each scenario. For each stakeholder playing a role (i.e., play) in the scenario that is going to be modeled, a set of specific scenario goals (i.e., Scenario Goal) is established as well as the association among them (i.e., contribute, hinder, include, extend, specialize). These scenario goals depend on one or more softgoals (see the dependency association in Figure 4.1) identified in the Context Modeling phase. Furthermore, it is important to define the subset of previously identified rules and regulations that govern each scenario (see the dependency association in Figure1 from Scenario to Rules and Regulations). For each scenario a SWOT analysis is also performed based on the scenario goal (see the dependency association in Figure 1 from SWOT analysis to Scenario Goal). This analysis may influence the decisions about "if and how" to proceed, for each scenario, with the subsequent Use Case Modeling phase. In the Use Case Modeling phase, for each modeled scenario, an application scenario (i.e., Application Scenario) can be set. Note that it is possible to decide to not specify any application scenario for one or more already modeled scenarios (see the dependency association from Application Scenario to Scenario in Figure 4.1); this happens often in large industrial research projects where only a small subset of scenarios is considered for developing some prototypes aimed at demonstrating the value of specific project results. Thus, the Use Case Modeling phase guides the project in the direction of what has to be considered in the subsequent development phase. The decision process behind this phase is not a formal process, but it is the result of a strong interaction among project partners based on the results of the Scenario Modeling phase. Each application scenario is defined in terms of its actors, use cases and processes (i.e., Actor, Use Case, Process). Actors and whichever of their specialization (i.e., specialize association) refer to roles identified in the previous phase (see the dependency association from Actor to Role). Among use cases GOReM considers contribute, hinder and specialize associations, but also other association stereotypes can be added if needed.

Figure 4.2 reports a BPMN diagram [72] that illustrates the GOReM reference process together with its main workproducts. Modelers having different skills perform the three phases. As an example, the Scenario Modeling phase concerns the parallel definition of different scenario models that can be performed by different scenario experts. A scenario model can be considered ready for the Use Case Modeling phase independently by other scenarios. For each business scenario, several use cases can be introduced in the Use Case Modeling phase in order to define the application scenarios able to define a specific set of functionalities. In this phase, each application scenario model can develop itself independently from the others and, thus, can be released or can require going back to the Scenario Modeling phase, as further specifications are needed. The decision to go back or to proceed to the next process phase is a critical decision, taken by the team working on the specific business and/or application scenario.

Summarizing, GOReM consists of three specific phases of modeling activities:

- *Context Modeling*, which aims to clearly represent the reference domain and it is described by some workproducts as follows. A Stakeholder Diagram shows an often hierarchical specification of all the stakeholders involved in the specific context. Each Stakeholder is in turn characterized by a set of Softgoals he intends to pursue [53]; a Softgoal Dependency Diagram shows the relationships between the stakeholders and the softgoals,as well as the relationships among softgoals (i.e., contributes, hinders, includes, extends, specializes); moreover, the Rules and regulations that govern the context are individuated.
- *Scenario Modeling*, where different business scenarios are specified in terms of the roles played by the involved stakeholders (i.e. some of those identified in the Context Model), their specific goals, and the specific rules and regulations that govern the business scenario. A SWOT Analysis (Strengths, Weaknesses, Opportunities and Threats) [73] is also performed with the aim to guide decisions on future work. From the Scenario Modeling phase, for each specific scenario, it is always possible to go back to the Context Modeling phase to better define or elicit some context elements of interest for the scenario (i.e., stakeholders, softgoals, rules and regulations).
- Application Modeling, where application scenarios are introduced in order to specify the functionalities which should be provided by a single

business scenario resulting from the previous phase. Each application scenario is characterized by functionalities that are modeled by UML-based Use Cases, Actors and Processes. From this phase, it is always possible to go back to the Scenario Modeling phase to better define or elicit some scenario elements (i.e., goals, roles, scenario rules and regulations, SWOT analysis).

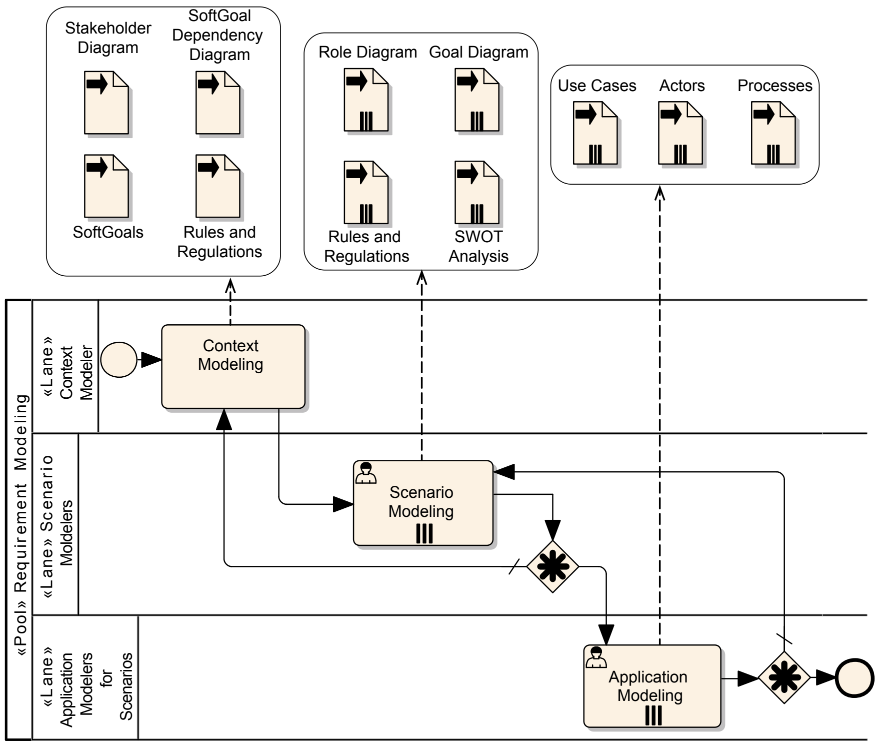


Fig. 4.2. GOReM Methodology

4.3 GOReM: Lesson learned

The lessons learned from the experience derived by exploiting the GOReM method on important research projects by cooperating with industrial partners such as ACI Informatica [66] and Poste Italiane [67], allowed to catch not only strengths but also weaknesses of the method, considered to refine and improve GOReM. The most interesting and relevant ”lessons learned” are reported in the following.

Lesson 1: human interactions and cooperation. It is probably the most difficult task due to different skills, backgrounds and knowledge that lead to big misunderstandings, lethal for establishing system requirements. It is likely to encounter mistakes when a new application domain is being explored because of: (i) misleading interpretation, due to the coexistence of different interpretations of stakeholder goals and requirements, that usually happens when people have different skills and the same concepts are interpreted differently according to the stakeholderś background; (ii) conflicting specifications, when specific strategies, that could potentially create strong disadvantages in other application scenarios are adopted in order to reach a specific goals in a specific application context; (iii) late discovery of redundancy, when in advanced development project stages the same concept is described and represented differently several time or different terminologies is used for describing the same concepts (iv) fragmentation of efforts; (v) weak focus on objectives for achieving the desired goals and being competitive and effective; (vi) partner coordination, when there exist different partners having different objectives to reach; (vii) workproduct integration, when there is a need to integrate, harmonize and handle deliverables, services and products coming from different tasks.
Lesson 2: cross-domain aspects. There are some recurrent features that might be identified once for all and there are also common characteristics for each domain of interest that have to be considered and properly represented, arising questions that need to be answered, such as:

- space: Is the considered context model influenced by the location and the territorial extension (e.g. regional, national, international, members states)?
- time: Is the considered context model influenced by temporal aspects (e.g. a new law replaces partially or totally a previous one)?

Whereas there are some features that need to be identified and analyzed according to the specific scenario, such as:

- subject: who/what is the subject of the described context?
- user profile: are the user preferences/personal features represented in the context model? Does the system describe the userś characteristics one by one or does it provide a role-based model of user classes?
- context history: does the current context state depend on a previous one?

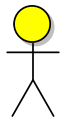
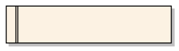
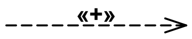
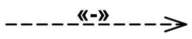
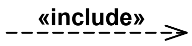
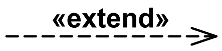
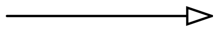
Lesson 3: legal aspects. The specific context model and the different business scenarios are handled by several Rules and Regulations that might be in conflict. As a consequence, it is important for modeling a context and any specific business scenario, to understand what laws are involved, what policy can be adopted as a "standard" or what best practice as a "guideline", depending on the stakeholders needs. In addition, there are stakeholders of specific customers that can have a set of internal policies which, in turn, should be considered and their eventual contrast with

some laws or requested best practices should be discovered and resolved. Finally, as a desired service can be used in different Nations, the requirement model has to analyze and manage the legal usability of a service for a given customer. Furthermore, requirements engineering process should manage legal aspects by continually monitoring their changes over the time, during the overall system lifecycle.
Lesson 4: tracing evolution. Business context, scenarios and applications can evolve because of their dynamic nature. It is important to have some tracing mechanism that allows knowing the application model versions derived from a scenario model version, and this last one which business context model version refers to. For big and continuously evolving system engineering process, this has a fundamental importance and especially for maintaining control and governing the system evolution along its life.
Lesson 5: interscenarios dependencies and reuse. Quite often, business scenarios evolve with a specific team of analyst/designer (sub)domain experts that have the objective to go ahead following their requirements engineering for specific final services. This can lead to duplication of work and, even worse, to services which do the same thing (same requirements) but in a different way. This is often difficult to discover and create customer dissatisfactions. This happens, for example, when the same stakeholder has two different goals belonging to two different scenarios, and the two application models reaching the two goals, share many "what to do" but unawares.

In the light of the above reported lessons learned during the method exploitation, starting from lesson n.1, an updated and refined version of the GOReM method is provided.

1. The Context Modeling phase.
   The Context Modeling phase aims at clearly representing the reference business domain for the project under consideration. The work-products of this phase are: a Stakeholder Diagram, showing a (hierarchical) specification of all the involved stakeholders, each of them is in turn characterized by a set of Softgoals they intend to pursue; a Softgoal Dependency Diagram, showing the relationships among Softgoals, (i.e., contribute, hinder, include, extend, generalize); a Rules and Regulations report shortly describing the rules and regulations governing the Context, distinguishing between Laws, National or International, and known used Policies and Best practices. Table 4.1 shows symbols already used in the first version of the methodology, while table 4.2 shows the identified and considered types of rules and regulations.
2. The Scenario Modeling phase.
   The Scenario Modeling phase specializes the Context Model through the identification of evolutionary scenarios that have to be modeled within the context of interest. Such scenarios are identified through an analysis that takes into account the roles played by stakeholders in each scenario, by

Table 4.1. THE CONTEXT MODEL - MAIN CONCEPTS

Concept	Graphical Notation	Description
Stakeholder		The UML Actor symbol extended through a yellow-filled head stereotype
Softgoal/Goal		TheSysML [74] Requirement native construct
Contribute Dependency		A UML Dependency symbol extended with a "+" stereotype
Hinder Dependency		A UML Dependency symbol extended with a "-" stereotype
Include Dependency		The UML native dependency applied among softgoals or goals
Extend Dependency		The UML native dependency applied among softgoals or goals
Generalize Dependency		The UML Generalize native symbol

indicating the specific Goals related to some Softgoals in the context model and the Rules and Regulations that govern the scenario. Table 4.3 shows symbols used for roles and for the associations with the stakeholders.
The SWOT Analysis activity [73], represented in a matrix (as showed in Table 4.4), provides an assessment of internal and external factors that may affect the scenario and may support decisions whether to continue with the next phase, that is the Application Modeling. For Goals and dependencies diagram, symbols in Table 4.1 are used.
Rules and Regulations selection activity considers the rules and regulations, identified in the Context Modeling phase, to be considered in the modeled scenario, by identifying them with a structured ID, describing them,specifying if they are laws, policies and best practices, indicating the adopters,and warning possible dependencies with other considered rules. In particular, GOReM uses the matrix formats, showed in table 4.5. This is an improvement introduced and allows to better manage the issues discussed in lesson 3 related to legal aspects.

Table 4.2. THE CONTEXT MODEL - RULES AND REGULATIONS

Type	**Description**
Best Practices	Best practice is considered a business buzzword, used to describe the process of developing and following a standard way of doing things that multiple organizations can use to maintain quality. It is not mandatory and can be based on self-assessment or benchmarking.
Polices	A Policy is a deliberate system of principles to guide decisions and achieve rational outcomes. It is a statement of intent, and it is implemented as a procedure or protocol.
National Law	National laws are valid and affect the State or Country that has enacted them.
International Law	International laws are enacted by specific Authorities and they govern the behavior of the Members States belonging to a specific community according to specific agreements.

Table 4.3. THE SCENARIO MODEL - ROLES

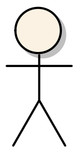
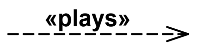
Concept	**Graphical Notation**	**Description**
Stakeholder's Role		The UML Actor symbol extended through a pink-filled head stereotype
Plays Dependency		A UML Dependency symbol extended with a "plays" stereotype

Table 4.4. THE SCENARIO MODEL - SWOT ANALYSIS

	HELPFULL	HARMFULL
Internal Origin	*Strengths*: what are the strengths(i.e. benefits controllable)	*Weaknesses*: what are the weak points (i.e. disadvantages controllable)
External Origin	*Opportunities*: possible opportunities (i.e. advantages not controllable)	*Threats*: potential threats (i.e. disadvantages not controllable)

3. The Application Modeling phase
   Starting from the scenarios defined during the previous phase, in the Application Modeling phase, a set of specific business scenarios might be identified. This phase defines application scenarios that are used to specify in detail the capabilities to be provided in the specific scenarios identified in the previous phase, along with main use cases description, actors

and processes. In particular, each main use case may become a service to be developed as a research prototype and/or developed and engineered as part of a more complete industrial system. In addition, some processes can be specified using UML or BPMN notations. Table 4.6 shows basic used symbols in modeling an application scenario. The Package is a Namespace of use cases, not in the scope of the modeled application, that it is assumed they exist in some different Application model, even in Application models obtained from different Scenario Models, while in this Application Model they have to be identified and extended through the standard "extend" UML relationship. This is how GOReM is now responding to lesson n.2 cross-domain aspects and lesson n.5, Inter-scenarios dependencies and reuse. The corresponding work-products should be more precise, and should indicate exactly the use case belonging to a scenario an the extending use case referred to, as well as the kind of needed extension. Every UML based diagram can be enriched with the UML comment symbol which allows adding a description to all the GOReM diagrams. However, a textual description and complete information is located in the corresponding work-product. Finally, concerning lesson n.4, tracing evolution, some shared existing policy of naming and versioning method/tool, for every model (context, scenario, application) and each of its work-products, must be used. In addition, some configuration management tool should help in maintaining the requirements evolution of the whole system. This allows knowing exactly for each application model, the scenario model and context model referred to. In addition, whichever refinement for a model created in one of the three GOReM phases must produce a new model referring the model it wants to improve. Moreover, each application model, if implemented, should refers to its development artefacts and releases in operation.

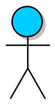
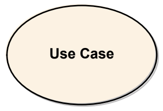
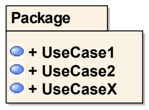
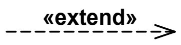
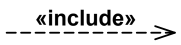
4.4 Case Studies

In this section some case studies, developed for real industrial research projects are presented. In particular, in the first subsection the application of GOReM in its first version is applied in the context of the Tourism as defined in [28]

Table 4.5. THE SCENARIO MODEL - RULES AND REGULATIONS

Identifier	Rules and Regulations	Type	Location/ Adopter	Warnings
StructuredID	Description	Policy/Best Practices/National Law/International Law	Locations and/or names of known adopters	List of identifiers of other rules and regulations which can have influence on its application

Table 4.6. THE APPLICATION MODEL - MAIN CONCEPTS

Concept	**Graphical Notation**	**Description**
Application Scenario Actor		The UML actor symbol extended through a blue-filled head stereotype
Use Case		The UML Use Case native symbol
Package		The UML NameSpace for Use cases supposed already existent in another Application Model.
Extend		The UML extend relationship
Include		The UML include relationship

and as published in the conference CIISE 2014 [23]; in the second subsection the application of GOReM to the Cyber Security domain and in particular to the Cyber Systemic Risk in the Business Continuity Plan of a Bank, analyzed and modeled for Poste Italiane in the behalf of the project related to the District of Cyber Security in Cosenza [67] and as published in the international conference CD-ARES 2016 [29] is presented; in the third subsection, the Cyber Security related to the Compliance Analysis in Cloud, is analyzed and modeled in two scenarios of interest for the cited District of Cyber Security. One of these scenarios was published in a paper for the international conference CloudTech 2016 [26].

4.4.1 Case Stydy: InMoto - Tourism

The first case study shows the exploitation of GOReM in a recently finished large industrial research project named "DICET- INMOTO" - ORganization of Cultural HEritage for Smart Tourism and Real-time Accessibility (OR.C.HE.S.T.R.A.), funded by the Italian Ministry of Education, University and Research (MIUR) within the PON Project - Research and Competitiveness 2007-2013. This Project involved 13 partners including universities, public research centers, small and medium-sized enterprises (SMEs), and large enterprises. The Project, started on November 2012 and finished at the end of 2015, aimed at defining and implementing models, processes and tools for

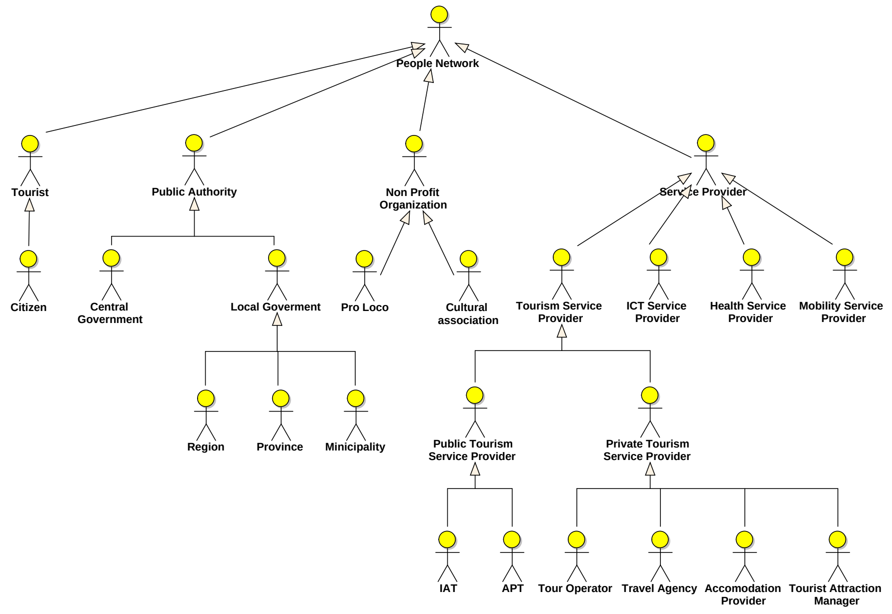


Fig. 4.3. Tourism in mobility Context Model: Stakeholder Diagram

sustainable development of an intelligent territory through the exploitation of its cultural heritage and environmental resources and the promotion and marketing of its tourist offer. In particular, the INMOTO Project dealt with technologies and innovative methods for goods and cultural contents exploitation and for the promotion of the linked territories for a sustainable tourism development. GOReM was used for the Context Modeling, Scenario Modeling and Use Case Modeling in the INMOTO Project.

The selected case study shows the *Tourism in mobility* Context Model, where it is possible to view stakeholders involved in the Tourism domain together with their hierarchical decomposition (Figure 4.3) where stakeholders are represented by using the standard UML actor symbol but with a yellow-filled head. For each stakeholder it is possible to view their softgoals (Figure 4.4), represented by rectangles. The relationships among softgoals are represented in the Softgoal Dependency Diagram work-product in Figure 4.5.

Finally, the Tourism in mobility context model is governed by many rules and regulations to be taken into account. For a comprehensive analysis refer to [75]

Starting from the Tourism in Mobility Context Model, the following scenarios have been identified: "Business to network for Tourism in Mobility", "Tourism in Mobility Brokerage", "Tourism in Mobility Knowledge Exploitation " As an example of Scenario Modeling, the "Tourism in Mobility Broker-

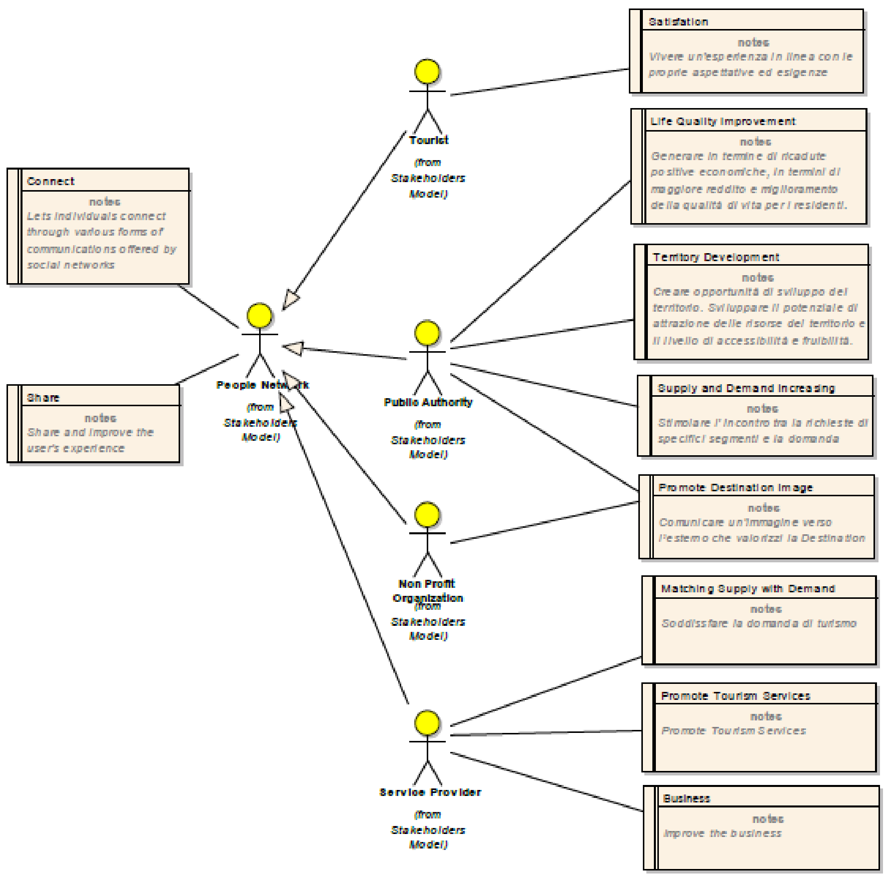

Satisfation
notes
Vivere un'esperienza in linea con le proprie aspettative ed esigenze
Life Quality Improvement
notes
Generare in termine di ricadute positive economiche, in termini di maggiore reddito e miglioramento della qualità di vita per i residenti.
Connect
notes
Lets individuals connect through various forms of communications offered by social networks
Tourist
(from Stakeholders Model)
Territory Development
notes
Creare opportunità di sviluppo del territorio. Sviluppare il potenziale di attrazione delle risorse del territorio e il livello di accessibilità e fruibilità.
People Network
(from Stakeholders Model)
Public Authority
(from Stakeholders Model)
Share
notes
Share and improve the user's experience
Supply and Demand Increasing
notes
Stimolare l' incontro tra la richieste di specifici segmenti e la domanda
Promote Destination Image
notes
Comunicare un'immagine verso l'esterno che valorizzi la Destination
Non Profit Organization
(from Stakeholders Model)
Matching Supply with Demand
notes
Soddissfare la domanda di turismo
Promote Tourism Services
notes
Promote Tourism Services
Service Provider
(from Stakeholders Model)
Business
notes
Improve the business

Fig. 4.4. Tourism in mobility Context Model: Softgoals Diagram

age" scenario is here investigated, where stakeholders and their roles played in the scenario are highlighted.

Brokerage activity in Tourism in Mobility fosters mutual collaboration, services and level of quality monitoring through coordinating policies and supporting activities made by actors belonging to the relational network. This scenario is particularly effective when the involvement of local institutions is foreseen in order to guarantee a real economic and organizational impact on the territory. Thus, the "Tourism in Mobility Brokerage" scenario derives from a best practice analysis of the sector that validate the idea that tourism policies are more effective when they come along with technological solutions seen as an organized collection of applications that make use of social networks. Tourism Broker enhances tourist offer on a territory and becomes supporter of the integration needed among the different system components and, in particular, among public and private actors, both during managing phases

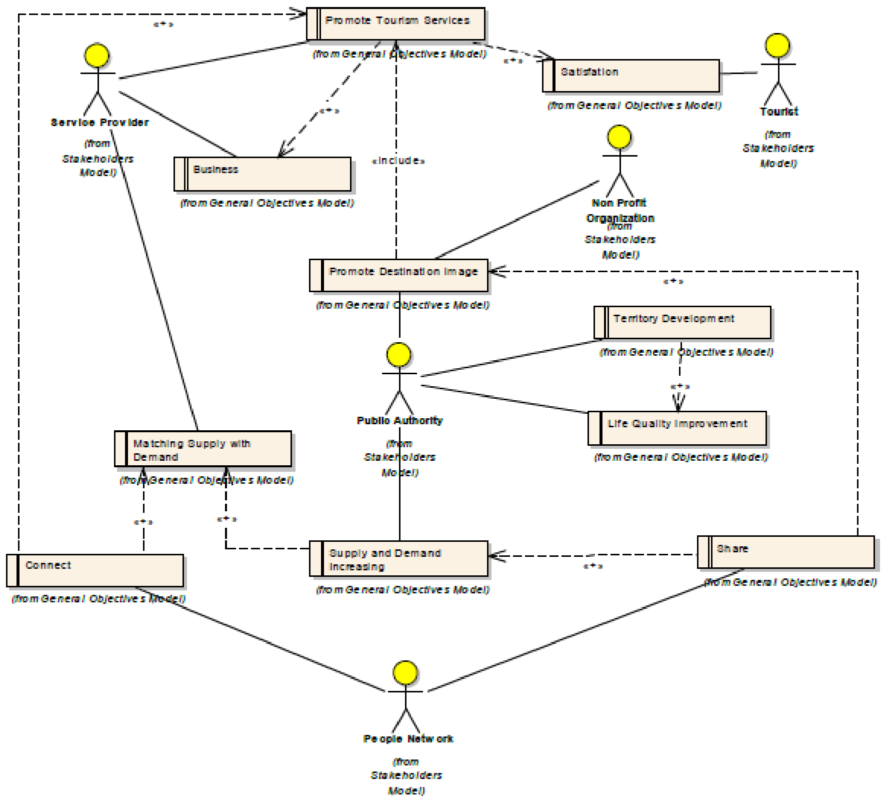

Promote Tourism Services
(from General Objectives Model)
Satisfation
(from General Objectives Model)
Tourist
(from Stakeholders Model)
Service Provider
(from Stakeholders Model)
Business
(from General Objectives Model)
«include»
Non Profit Organization
(from Stakeholders Model)
Promote Destination Image
(from General Objectives Model)
Territory Development
(from General Objectives Model)
Public Authority
(from Stakeholders Model)
Life Quality Improvement
(from General Objectives Model)
Matching Supply with Demand
(from General Objectives Model)
Connect
(from General Objectives Model)
Supply and Demand Increasing
(from General Objectives Model)
Share
(from General Objectives Model)
People Network
(from Stakeholders Model)
«+»

Fig. 4.5. Tourism in mobility Context Model: Softgoals Dependency Diagram

and project development phases. The Role Diagram of "Tourism in Mobility Brokerage" scenario is depicted in Figure 4.6.

Figure 4.7 shows the main goals that each involved stakeholder with a specific role is reaching in this business scenario, while Figure 4.8 shows the dependencies among goals of the modeled scenario.

Finally, a specific Application model named "Event Tourism", is presented. This Application model includes a main Use Case diagram where actors, corresponding to specific roles, can perform a basic set of functionalities for events organization (i.e. creation, planning, production and follow up). In agreement with [76], organization of new events can be considered as resources to exploit, since they are highly valued as attractions, catalysts, animators, place marketers and image for increasing the value of a destination.

The following Top Level Use Case diagram has been identified: "Event Tourism". This functionality involves as primary actor the Event Broker and, as primary process, the composite process "Event Production" including an

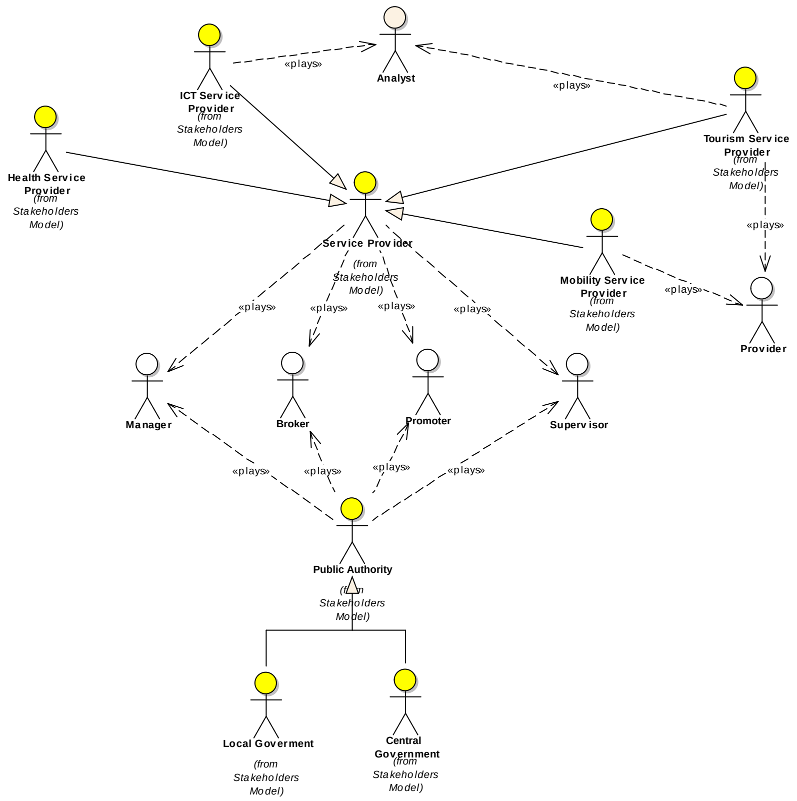

Analyst
«plays»
ICT Service Provider
(from Stakeholders Model)
Health Service Provider
(from Stakeholders Model)
Tourism Service Provider
(from Stakeholders Model)
Service Provider
(from Stakeholders Model)
Mobility Service Provider
(from Stakeholders Model)
Provider
Manager
Broker
Promoter
Supervisor
Public Authority
(from Stakeholders Model)
Local Goverment
(from Stakeholders Model)
Central Government
(from Stakeholders Model)

Fig. 4.6. Tourism in Mobility Brokerage Scenario Model: Role Diagram

internal (sub) process named "Packaging with collateral services", from which other related use cases arose.

These information are sketched in the Use Case Diagrams showed in Figure 4.9, while Figure 4.10 e Figure 4.11 shows the process and the named (sub) process.

Results and Conclusions of the Case Study

Concluding, in the project DICET-INMOTO, GOReM has proved its effectiveness in fostering cooperation and clean project evolution due to its capability to fully support the requirements engineering phase by offering system models that, although easy to draw and understand, can be used as a valid ground to feed the design and development phases.

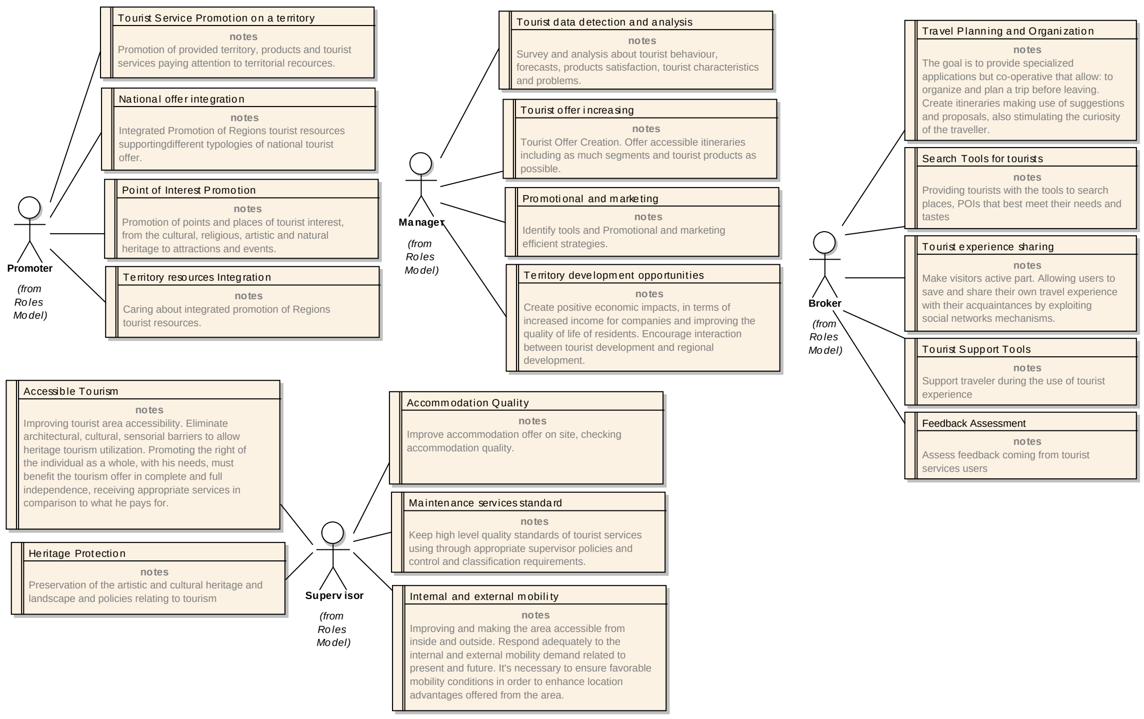

Tourist Service Promotion on a territory
notes
Promotion of provided territory, products and tourist services paying attention to territorial recources.
National offer integration
notes
Integrated Promotion of Regions tourist resources supportingdifferent typologies of national tourist offer.
Point of Interest Promotion
notes
Promotion of points and places of tourist interest, from the cultural, religious, artistic and natural heritage to attractions and events.
Territory resources Integration
notes
Caring about integrated promotion of Regions tourist resources.
Promoter
(from Roles Model)
Tourist data detection and analysis
notes
Survey and analysis about tourist behaviour, forecasts, products satisfaction, tourist characteristics and problems.
Tourist offer increasing
notes
Tourist Offer Creation. Offer accessible itineraries including as much segments and tourist products as possible.
Promotional and marketing
notes
Identify tools and Promotional and marketing efficient strategies.
Territory development opportunities
notes
Create positive economic impacts, in terms of increased income for companies and improving the quality of life of residents. Encourage interaction between tourist development and regional development.
Manager
(from Roles Model)
Travel Planning and Organization
notes
The goal is to provide specialized applications but co-operative that allow: to organize and plan a trip before leaving. Create itineraries making use of suggestions and proposals, also stimulating the curiosity of the traveller.
Search Tools for tourists
notes
Providing tourists with the tools to search places, POIs that best meet their needs and tastes
Tourist experience sharing
notes
Make visitors active part. Allowing users to save and share their own travel experience with their acquaintances by exploiting social networks mechanisms.
Tourist Support Tools
notes
Support traveler during the use of tourist experience
Feedback Assessment
notes
Assess feedback coming from tourist services users
Broker
(from Roles Model)
Accessible Tourism
notes
Improving tourist area accessibility. Eliminate architectural, cultural, sensorial barriers to allow heritage tourism utilization. Promoting the right of the individual as a whole, with his needs, must benefit the tourism offer in complete and full independence, receiving appropriate services in comparison to what he pays for.
Heritage Protection
notes
Preservation of the artistic and cultural heritage and landscape and policies relating to tourism
Supervisor
(from Roles Model)
Accommodation Quality
notes
Improve accommodation offer on site, checking accommodation quality.
Maintenance services standard
notes
Keep high level quality standards of tourist services using through appropriate supervisor policies and control and classification requirements.
Internal and external mobility
notes
Improving and making the area accessible from inside and outside. Respond adequately to the internal and external mobility demand related to present and future. It's necessary to ensure favorable mobility conditions in order to enhance location advantages offered from the area.

Fig. 4.7. Tourism in Mobility Brokerage Scenario Model: Role Diagram

4.4.2 Case Study: Cyber Security - Systemic Risk in Bank

This second case study is related to the modeling Cyber Systemic Risk for the Business Continuity Plan of a Bank, which has been a topic of interest for Poste Italiane in the project Cyber Security [67].

The pervasive growth and diffusion of complex IT systems, that handle critical business aspects of today's enterprises and cooperate through computer networks, has given rise to a significant expansion of the exposure surface towards cyber security threats. A threat, affecting a given IT system, may cause a ripple effect on the other interconnected systems often with unpredictable consequences. This type of exposition, known as cyber systemic risk, is a very important concern especially for the international banking system and it needs to be suitably taken into account during the requirement analysis of a bank IT system. GOReM seemed appropriate during the requirements specification, to consider adequate provisions for prevention and reaction to cyber systemic risk in banking systems.

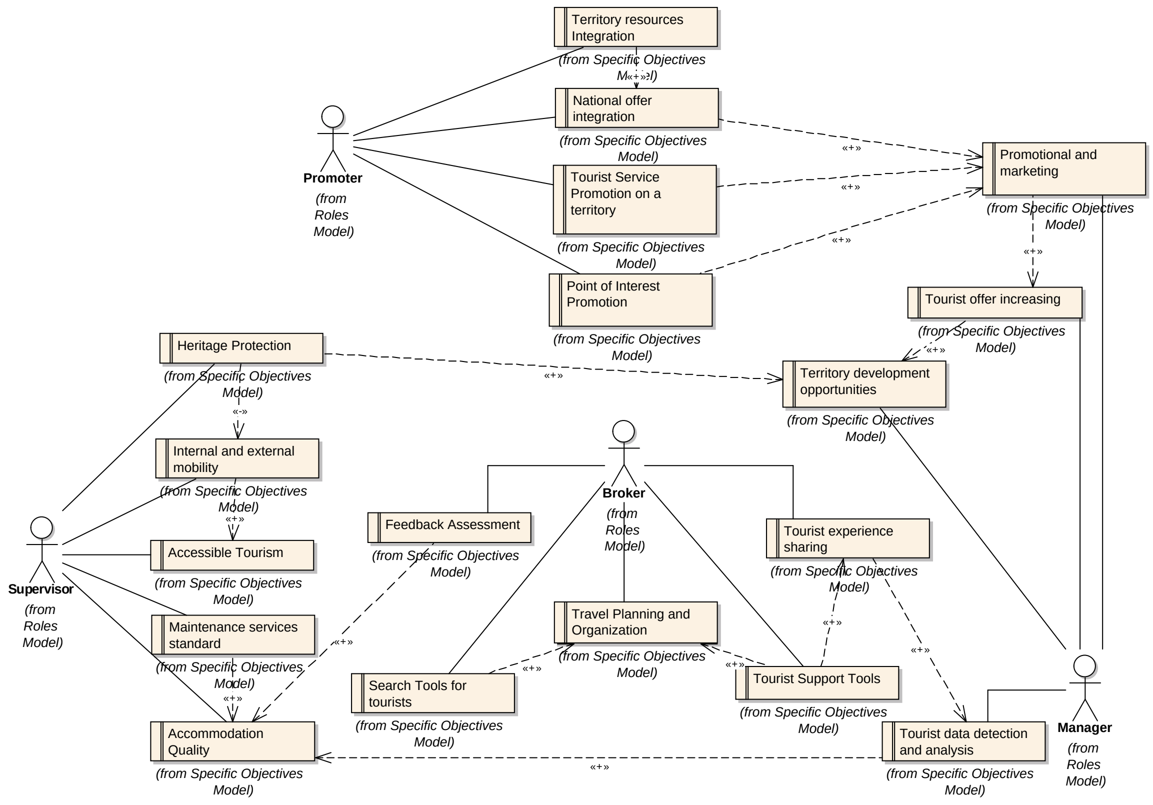

Territory resources Integration
National offer integration
Tourist Service Promotion on a territory
Point of Interest Promotion
Promotional and marketing
Tourist offer increasing
Territory development opportunities
Heritage Protection
Internal and external mobility
Accessible Tourism
Maintenance services standard
Accommodation Quality
Feedback Assessment
Travel Planning and Organization
Search Tools for tourists
Tourist experience sharing
Tourist Support Tools
Tourist data detection and analysis
(from Specific Objectives Model)
Promoter
Supervisor
Broker
Manager
(from Roles Model)
«+»
«-»

Fig. 4.8. Tourism in Mobility Brokerage SCENARIO: Roles Dependency Diagram

In particular, the context of the Italian banking system is considered in this case study.

Nowadays, a big-enterprise IT system is usually geographically distributed, pervasive and ubiquitous for its internal and external users. Therefore, each of such systems consists of a network of subsystems where the cyber systemic risk must be reduced as much as possible. Cyber security risk has to be continuously monitored, while real-time recovery and support procedures, assuring an enough degree of system availability, have to be provided [77, 78]. Systemic effects have to be reduced and global collaboration among all stakeholders, both public and private, should be provided for an effective proactive prevention of a cyber shock of our global, not only financial, networked systems [79].

In a recent white paper [80], the Depository Trust & Clearing Corporation (DTCC) states that a global cyber systemic risk could become less dangerous if the defense is both collective and coordinated, otherwise the failure is quite sure. The last DTCC report on systemic risk [81] is very alarming on the cyber risk for the worldwide financial markets. Therefore, instead of providing

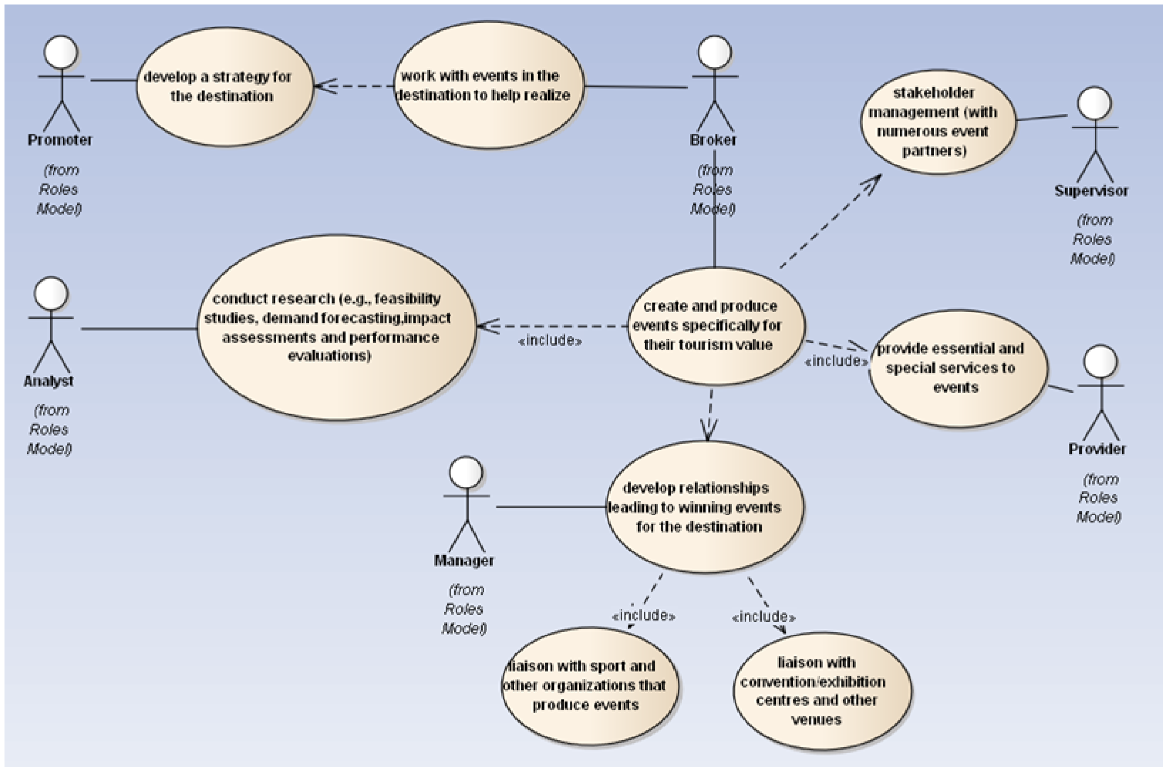

Promoter
(from Roles Model)
develop a strategy for the destination
work with events in the destination to help realize
Broker
(from Roles Model)
stakeholder management (with numerous event partners)
Supervisor
(from Roles Model)
Analyst
(from Roles Model)
conduct research (e.g., feasibility studies, demand forecasting,impact assessments and performance evaluations)
«include»
create and produce events specifically for their tourism value
«include»
provide essential and special services to events
Provider
(from Roles Model)
Manager
(from Roles Model)
develop relationships leading to winning events for the destination
«include»
«include»
liaison with sport and other organizations that produce events
liaison with convention/exhibition centres and other venues

Fig. 4.9. Event Tourism Application Model: Use Case Diagram

specific cyber risk defenses for each system, a global cyber systemic risk [83, 84] strategy should be devised and enforced by means of the adoption of shared rules, regulations and common approaches.

This case study focus on the banking context and, specifically on the business continuous plan (BCP) and its disaster recovery plan (DRP), as regulated by the Bank of Italy for the banking operators located in Italy [85]. However, each Bank operating in the European Union must provide similar guidelines for BCP and DRP of their banking operators.

The definitions of BCP and DRP are driven worldwide by many sectoral rules and regulations [82], without any global coordination. A supervising institution, having the authority to push and actually drive the different BCPs, would be able to manage the global systemic risk by a coordinated strategy. Moreover, the 2016 edition of "The Global Risks Report" [86], by the World Economic Forum, outlines the need for cooperation among stakeholders for risk management and mentions some tests performed in Germany.

By means of GOReM, the requirements for the cyber systemic risk treatment for a bank operating in Italy are here modeled. All Italian banks follow rules and regulations delivered by the Bank of Italy. However, each European

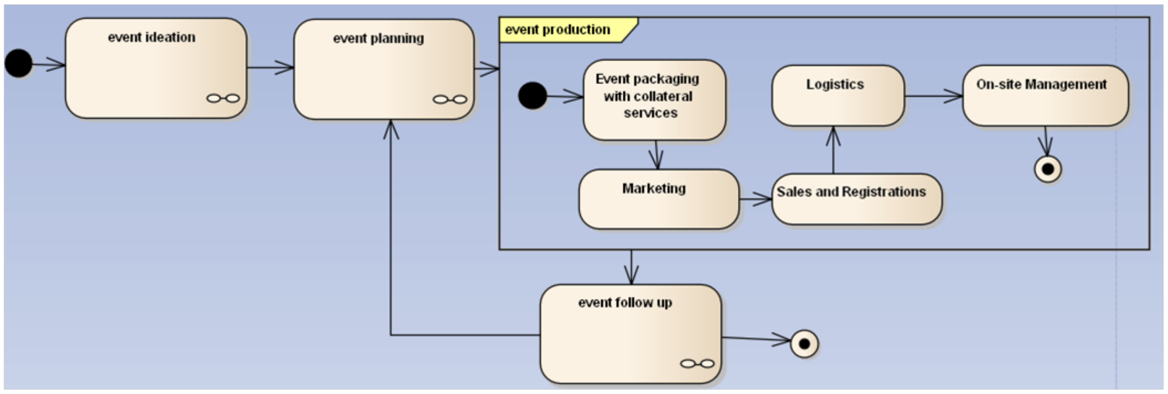


Fig. 4.10. Event Tourism Application Model: Main process

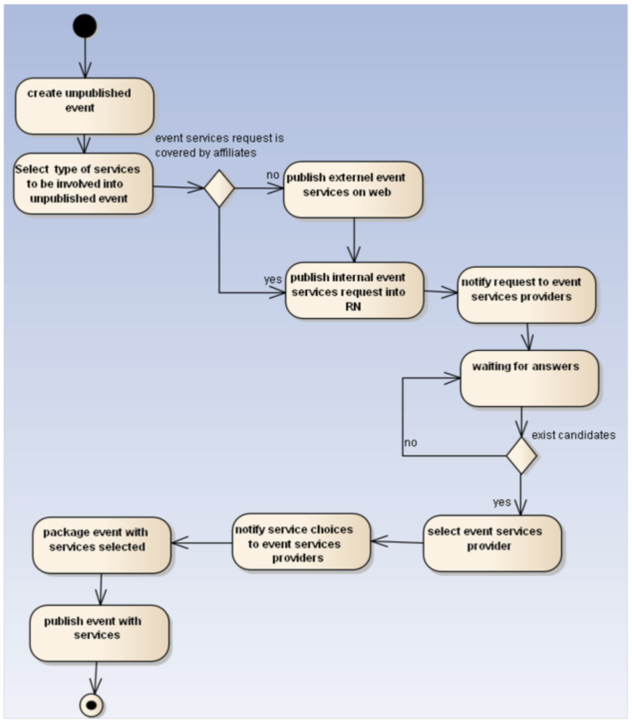

create unpublished event
Select type of services to be involved into unpublished event
event services request is covered by affiliates
no
publish externel event services on web
yes
publish internal event services request into RN
notify request to event services providers
waiting for answers
exist candidates
no
yes
select event services provider
notify service choices to event services providers
package event with services selected
publish event with services

Fig. 4.11. Event Tourism Application Model: Event packaging process

Nation has a central banking institution which establish similar guidelines for the local banks. Then, the developed models might be applied, with small adaptations, to whichever bank in Europe.

In particular, the models obtained using GOReM, are compliant with the guidelines established by Bank of Italy [85]. Those models allowed to easily highlight how a BCP has two different ways to handle the cyber systemic risk. The first includes critical processes which might develop contagion only to the internal stakeholders of the bank (including counterparts cooperating to the business of the bank). In this case the ripple effect of an incident is treated at the bank level and the Bank of Italy is only notified. The second cyber systemic risk treatment is related to the safeguard of systematically important processes of the payment systems and of the access to financial markets. In this case, both BCP and the handling of a possible ripple effect of an incident on other banks and, more generally, on external entities, is strongly centralized by the Bank of Italy. The latter is a hierarchical control, although with rigid response time, which might introduce delays in the tentative to slow down or stop contagion in the European and even worldwide financial system [87].

Following GOReM, the Context Model of the business continuity in a bank and the description of one of the possible business scenarios, that is the risk treatment in bank, are first described. Then, one specific application scenario, concerning the treatment of cyber systemic risk for the so called "systematically important processes" [85] of a bank, is modeled in terms of actors, use cases and processes.

The Context Model: banking business continuity

The banking system has a complex organizational infrastructure. In the following, it is modeled a small subset of such an infrastructure, with the only objective to give an idea of the effectiveness and powerfulness of employing GOReM for this purpose.

The term *Business Continuity* (BC) refers to all of the organizational, technical and staffing measures employed in order to: (i) ensure the continuation of core business activities in the immediate aftermath of a crisis and (ii) gradually ensure the continued operation of all business activities in the event of sustained and severe disruption

To this end, each bank must define a Business Continuity Plan (BCP), i.e. a formal document stating the principles, setting the objectives, describing the procedures and identifying the resources for business continuity management concerning critical and systemically important corporate process [85]. The bank must also use internal audit, testing activity and continuously improvement implementations of its BCP, with the aim to: (i) analyze well the exposure to risks, (ii) identify vulnerabilities, and (iii) evaluate, implement and maintain updated, appropriate BC and Disaster Recovery (DR) solutions. A critical part of the BCP is the Disaster Recovery Plan (DRP), i.e. a document establishing the technical and organizational measures to cope with events that put electronic data processing (EDP) centers out of service.

Despite suitable tools and countermeasures constantly in action to prevent their occurrence, unfortunately accidents happen. In these cases, it is essential that a BCP is promptly put in operation, to ensure the continuity of services.

Hence, the appropriated Disaster Recovery procedures, as specified by the DRP, have to begin immediately.

Fig. 4.12 shows a GOReM diagram that depicts the stakeholders which were identified for this context, their softgoals and the dependencies among them.

The main stakeholders are: the **Bank of Italy**; **Banking System Operator**, which can be of two different types, that is **Operator of technological infrastructures or networks**, and **operating companies**, i.e. wholesale markets in government securities, multilateral wholesale trading facilities in government securities, multilateral deposit trading systems, securities settlement systems, central counterparties and central securities depositories, with registered offices and/or operational headquarters in Italy; **Bank personnel**, that is people, including corporate bodies, working internally in the bank; **Service Provider**, i.e. external stakeholders that provide IT services and other commodities, by stipulating specific contracts with the bank; **Selling Net**; **Shareholder** and **Customer**.

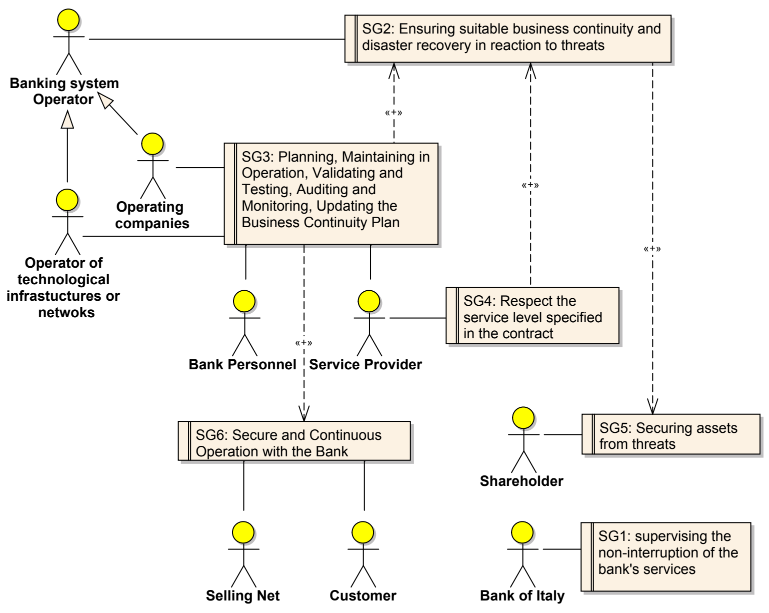

Banking system Operator
SG2: Ensuring suitable business continuity and disaster recovery in reaction to threats
«+»
«+»
«+»
«+»
Operating companies
SG3: Planning, Maintaining in Operation, Validating and Testing, Auditing and Monitoring, Updating the Business Continuity Plan
Operator of technological infrastuctures or netwoks
SG4: Respect the service level specified in the contract
Bank Personnel
Service Provider
SG6: Secure and Continuous Operation with the Bank
Shareholder
SG5: Securing assets from threats
Selling Net
Customer
Bank of Italy
SG1: supervising the non-interruption of the bank's services

Fig. 4.12. Context Model: The Banking Business Continuity's Softgoals and Dependencies Diagram

Each stakeholder is associated to a set of softgoals, as it can be seen from Fig.4.12. The identified softgoals are resumed in the following.
SG1: Supervising the non-interruption of the bank's services.

SG2: Every operator has to put into execution the suitable provisions, according to the BCP, for ensuring business continuity and disaster recovery in reaction to threats.
SG3: Planning, keeping into operation, validating and testing, auditing and monitoring, updating the BCP. This softgoal is shared by all kinds of considered bank operators.
SG4: Guaranteeing the service level specified in the contract (i.e. external providers must stipulate a contract with the bank that specifies a service level agreement among the parties and that has to be compliant to the business continuity needs).
SG5: Safeguarding assets from threats. Shareholders need to be ensured about the safety of their financial assets.
SG6: Guaranteeing secure and continuous operation of the bank. Both customers and selling nets need always working and safe banking services.

Figure 4.12 also outlines the existing dependencies among Softgoals. In particular, the achievement of SG3 and SG4 contributes to SG2. Similarly, reaching SG2 has a positive effect on SG5 and analogously the same holds for SG3 on SG6.

Scenario Model: Risk treatment

The scenario chosen to be modeled is about the treatment of risks coming from bank defaults, financial and market crashes, human mistakes, cyber threats and so on. This scenario includes situations such as: destruction or inaccessibility of important structures, unavailability of critical information systems, unavailability of human resources essential to corporate processes, interruption of operation of infrastructure (e.g. electricity, telecommunications, interbank networks, financial markets), alteration or loss of critical data and documents.

According to the Bank of Italy guidelines [85], operators must define, monitor, test and maintain updated, a BCP for coping with the above situations of crisis, involving the operators or significant counterparts such as, other group members, major suppliers, prime customers, specific financial markets, clearing, settlement and guarantee systems.

An important step in applying GOReM is the identification of the roles played by each involved stakeholder. Each Stakeholder, while playing a given role, has some specific goals he want to reach inside the scenario. The stakeholders-roles mapping alongside the role-specific goals are resumed by the diagram reported in Fig.4.13.

Table 4.7 give details on each goal of the considered scenario while Table4.8 lists a subset of the rules and regulations of interest for the scenario of Risk treatment. A unique identifier is associated to each rule/regulation and some relationships of warning with respect to others rules/regulations is given (column W) for indicating the need of a deeper analysis when applied in practice.

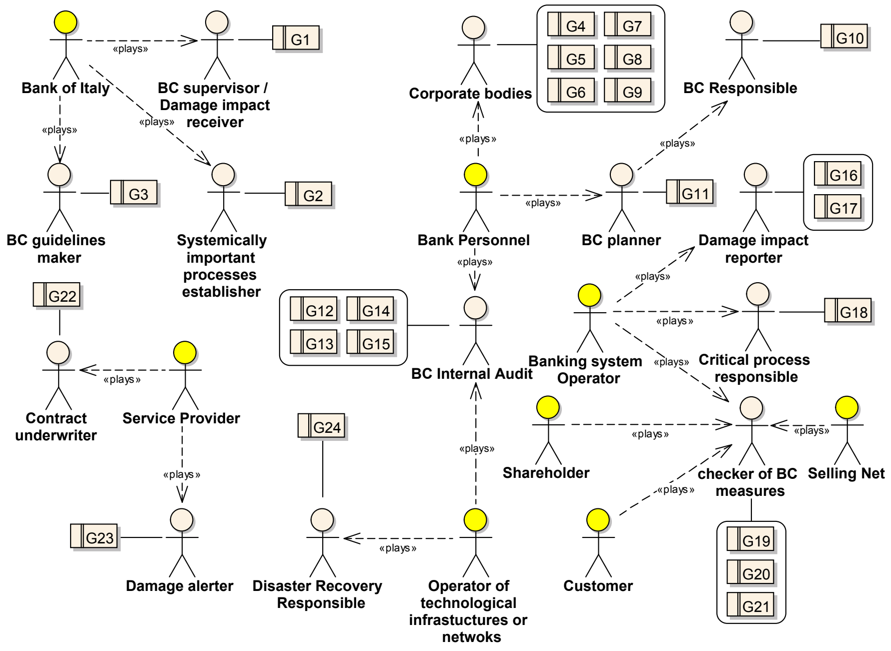


Fig. 4.13. Scenario Model for Risk treatment: SoftGoals-Roles-Goals Diagram

Application Model: Cyber Systemic Risk for banks in Italy

The application model considered is related to the Cyber Systemic Risk as dealt with by the Bank of Italy. This is one of the application models that might be derived from the above presented Risk Treatment scenario.

This application model deals with business continuity and safeguards for the so referred "Systematically Important Processes", identified and controlled directly by the Bank of Italy, that govern essential services in the payment system and in the access to the financial markets. A malicious exploitation of a cyber threat for these processes might evolve in a systemic crisis inside other operators and on the whole financial system. For those processes, the Bank of Italy controls, asks for updates, and manages every risk of crisis and incidents.

The Bank of Italy requires that the operators, involved in systematically important processes, work actively for adjustment of the BCP. These operators must comply with stricter business continuity requirements than those which normally apply to all operators. In particular, these requirements are concerned with the recovery time of systemically important processes, the location of standby facilities, and the resources allocated to crisis management (see section III of [85]).

Table 4.7. SCENARIO MODEL FOR RISK TREATMENT: ROLE AND GOAL DESCRIPTION

Stakeholder	Role	Goal	Description
Bank of Italy	BC supervisor / Damage impact receiver	G1	Each operator has its suitable Business Continuity Plan and the impact of possible damages undergone by its Banks for specific systemically important processes, is managed
	Systemically important processes establisher	G2	Systematically important processes are individuated and assigned for being protected by a suitable operator
	BC guidelines maker	G3	Each operator refers to guidelines for business continuity aligned with the actual European level of risk knowledge
Bank personnel	Corporate bodies, i.e. considered part of the bank personnel	G4	Establish objectives and strategies for BCP of the bank
		G5	Assign human, technological and financial resources sufficient to attain the objectives of the BCP
		G6	Approve the BCP and successive modifications resulting from technological and organizational adjustments and formally accept the residual risks not covered by the BCP
		G7	Control the results of checks on the adequacy of the BCP and of its measures, done at least once a year
		G8	Designate the person responsible for business continuity planning
		G9	Promote the development and regular checking of the BCP and its adaptation to any significant organizational, technological or infrastructural innovations and in the case of detection of shortcomings or the materialization of new risks
	BC Responsible	G10	Supervise the planning of the BCPs by means of the coordination of every involved BC planner
	BC Planner	G11	Establish the BCP for the operator, compliant to the guidelines provided by Bank of Italy
	BC Internal Audit	G12	Check, at fixed times, the BCP and its updating by examining the test programs, taking part in the tests and checking the results, and suggesting changes to the BCP on the basis of the shortcomings found
		G13	Analyze the criteria for escalation in the case of incidents, by evaluating the length of time required to declare the state of crisis
		G14	Test the BCPs of the outsourcers and other critical suppliers and may decide to rely on the controls performed by the structures of the latter if they are deemed professionally capable, independent and transparent
		G15	Examine the outsourcing contracts to make sure that the level of safeguards conforms the corporate objectives and standards
Banking System Operator	Damage Impact reported	G16	Produce an impact analysis, preliminary to the drafting of the BCP and regularly update the impact analysis, with the aim to determine the level of risk for each corporate process and highlight the repercussions of a service outage. The impact analysis considers, in addition to operational risks, also such other risks as market and liquidity risk
		G17	Document the residual risks, not handled by the BCP, which must be explicitly accepted by the competent corporate bodies
	Critical process responsible	G18	Identify relevant processes relating to corporate functions whose non-availability, owing to the high impact of the resulting damage, necessitates high levels of business continuity to be achieved through preventive measures and BC solutions activated in case of incident
	Checker of BC measures	G19	Shareholders, together with the bank system operators and the selling net , as well as with customers, cooperate in defining the procedures for testing the planned business continuity measures in real crisis scenarios
		G20	Determine an appropriate frequency of the testing task for each measures
		G21	Write down and notify the results of tests to the competent corporate bodies and transmit, for the matters under their respective competence, to the operational units
Service provider	Contract underwriter	G22	Ensure the service levels agreed with the operators, as formally state in the signed contract, in the case of crisis and ensure the continuity provisions to be put in place in keeping with attainment of corporate objectives and with the indications of the Bank of Italy
	Damage alerter	G23	Notify promptly the operator of any incident, in order to allow immediate activation of the BC procedures
Operator of technological infrastructures or networks	Disaster Recovery Responsible	G24	Define and maintain updated the DRP, with reference to central and peripheral information systems

Fig. 4.14 shows the Actors Diagram relevant to this application model, where scenario roles are mapped to actors, and Fig. 4.15 resumes the main use cases involving the identified actors.

Some use cases are extensions of some others that are supposed to be already defined in another application model, named “Business continuity management”, where the constraints by the Bank of Italy are less stringent and related to critical processes that are not systematically important processes. A short description of these use cases is given below.

BCP adjustments and compliance monitoring (**eUCa**). This use case extends the use case **UCa**, part of the use cases concerning critical processes,

Table 4.8. SCENARIO MODEL FOR RISK TREATMENT: RULES AND REGULATIONS DIAGRAM

Id	**Rules and Regulations**	**Type**	**Location/ Adopter**	**W**
A	CPMI-IOSCO consultative paper "Guidance on cyber resilience for financial market infrastructure", November 2015	Best practice	EU	B, C
B	Opinion of the European Central Bank of 25 July 2014 on a proposal for a directive of the European Parliament and of the Council concerning measures to ensure a high common level of network and information security across the Union (CON/2014/58)	Policy	EU	A, C
D	Guidelines on business continuity for market infrastructures	Best practice	Italy	A,B, E, F
E	Legislative Decree 385/1993 (the Consolidated Law on Banking)	Law	Italy	A, B, G
F	Legislative Decree 58/1998 (the Consolidated Law on Finance)	Law	Italy	A, B, H
G	Business continuity oversight expectations for systemically important payment systems, issued by Eurosystem in June 2006	Best practice	European Union	E, F, H
H	Principles for Financial Market Infrastructures, issued by Bank for International Settlements Committee on Payment and Settlement Systems (CPSS) and IOSCO Technical Committee, April 2012	Best practice	European Union	E, F, G

not belonging to the set of those considered *systematically important*. BC and DR plans, defined in the application model "Business Continuity Management", require some adjustments to become compliant with the stricter requirements defined by the Bank of Italy. The operator must also ensure continuous compliance with the special requirements and all this must be done by the responsible for these activities (i.e. the actor "BC and DR plans requirement adjustment and compliance responsible").

Main incidents and recurrent criticality check (**eUCc**). This use case extends the use case **UCc** (*Business continuity plan checking*). The Bank of Italy requires at least one test per year for the safeguards provided for the continuity of the systemically important processes. Operators must actively participate in tests and market-wide simulations, organized or promoted by authorities, by markets and by the main financial infrastructures. In addition, this use case prescribes the drafting of a yearly report about: the main features of the business continuity plan; the adaptations that have been made to it; the additions implemented during the year; the tests conducted on the main incidents and criticalities.

Managing crisis notification to and from Bank of Italy (**eUCd**). This use case extends the use case **UCd**, related to notification to Bank of Italy, when

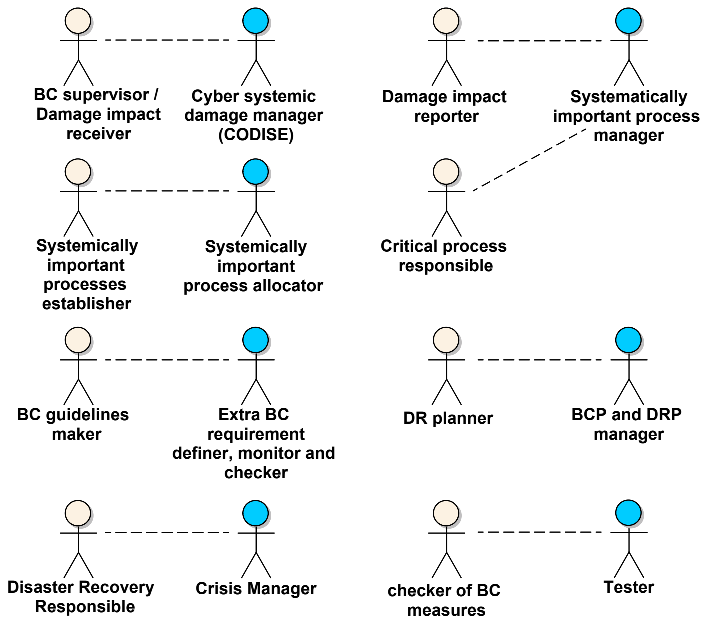

BC supervisor / Damage impact receiver
Cyber systemic damage manager (CODISE)
Damage impact reporter
Systematically important process manager
Systemically important processes establisher
Systemically important process allocator
Critical process responsible
BC guidelines maker
Extra BC requirement definer, monitor and checker
DR planner
BCP and DRP manager
Disaster Recovery Responsible
Crisis Manager
checker of BC measures
Tester

Fig. 4.14. Application Model for Cyber Systemic Risk: Actor Diagram

the blockage of essential infrastructures is related to internal critical processes. In this case, the actor ”crisis manager” must instead communicates promptly to the Bank of Italy every cyber attack and state of crisis coming from some threat to its systemically important processes. Furthermore, the use case includes the sending of an assessment report, drafted according to **UC4**. In addition, this use case dictates that the “crisis manager” receives the notification, coming from the Bank of Italy, that other operators are subject to a cyber attack, which might cause contagion to some of its systematically important processes. Then, this actor should raise an alert so that recovery procedures may immediately begin (see **UC3**).

Systemically important processes breach management (**eUCe**). This use case extends the use case **UCe**. The actor “systemically important processes manager” activates immediately the recovery procedures as indicated by the BCP and DRP when a breach to some process under its observation occurs. As specified by the guidelines, these procedures govern:

(i) the recovery time that, if the cause of the blockage is internal to the operator, must not exceed four hours and the restart time must not exceed two hours. If the blockage is due to an external contagion, the operator must activate his DR within two hours from the restart of the first affected operator. For information systems with on line duplication of operational data the time between the recovery point and the incident should zeroed.

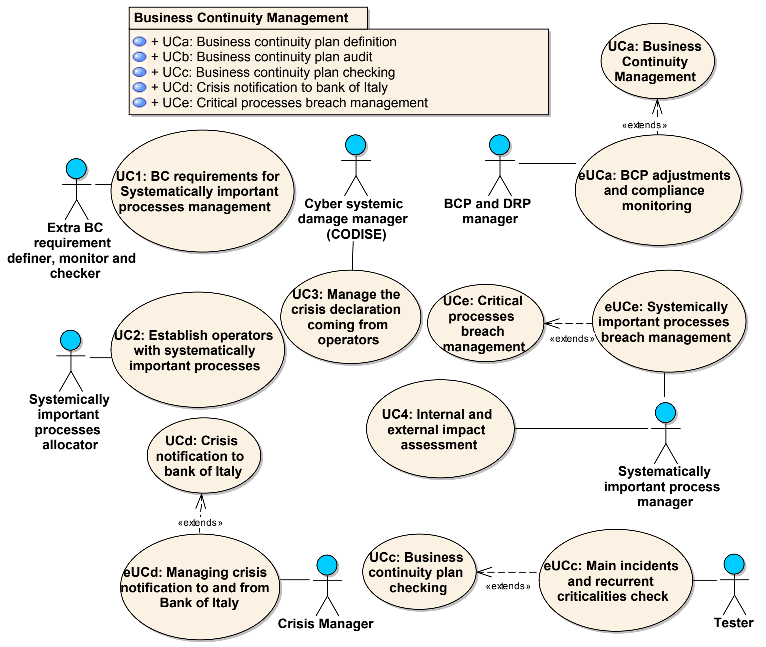

Business Continuity Management
+ UCa: Business continuity plan definition
+ UCb: Business continuity plan audit
+ UCc: Business continuity plan checking
+ UCd: Crisis notification to bank of Italy
+ UCe: Critical processes breach management
UCa: Business Continuity Management
«extends»
eUCa: BCP adjustments and compliance monitoring
BCP and DRP manager
Cyber systemic damage manager (CODISE)
UC1: BC requirements for Systematically important processes management
Extra BC requirement definer, monitor and checker
UC3: Manage the crisis declaration coming from operators
UCe: Critical processes breach management
«extends»
eUCe: Systemically important processes breach management
UC2: Establish operators with systematically important processes
Systemically important processes allocator
UC4: Internal and external impact assessment
Systematically important process manager
UCd: Crisis notification to bank of Italy
«extends»
eUCd: Managing crisis notification to and from Bank of Italy
Crisis Manager
UCc: Business continuity plan checking
«extends»
eUCc: Main incidents and recurrent criticalities check
Tester

Fig. 4.15. Application Model for Cyber Systemic Risk: Main Use Case Diagram

In case of extreme situations, promptly recovery of systemically important processes, using protected off line PCs, faxes, and telephone contacts with selected counterparts, is allowed.

(ii) the location of standby facilities, which must be distant from their primary facilities, possibly outside the metropolitan area where the primary facility is located and it must be served by utilities (i.e. telecommunications, electricity, water) different from those serving the primary facility.

(iii) the resources allocated to crisis management. Human, technological and logistical resources needed to keep systemically important processes operating are established in the BCP.

BC requirements the management of systematically important processes (**UC1**). Stricter BC requirements for systematically important processes are established by Bank of Italy. This use case directly controls the operators adjustment and the compliance of their BCPs to the evolving requirements imposed by the Bank of Italy.

Establish which operator has systematically important processes (**UC2**). The Bank of Italy is in charge to individuate the specific set of operators having systematically important processes.

Manage the crisis declaration coming from operators (**UC3**). For incidents that may have significant impact on systemically important processes, the declaration of the state of crisis is managed by CODISE, part of Bank of Italy,

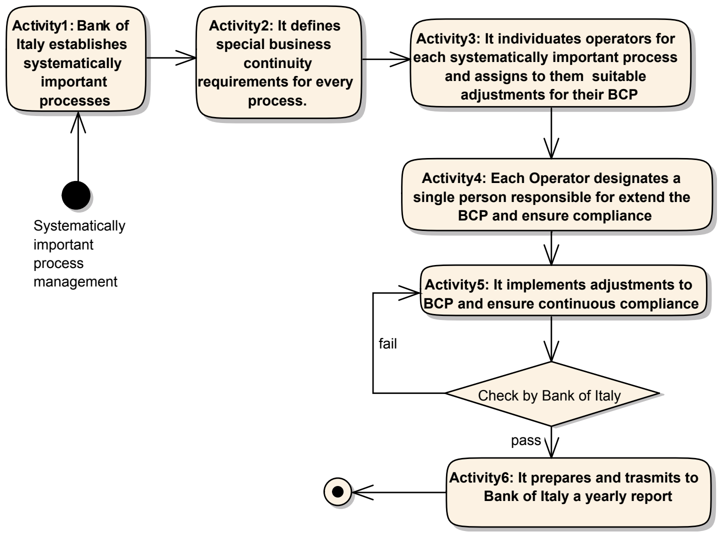

Activity1: Bank of Italy establishes systematically important processes
Activity2: It defines special business continuity requirements for every process.
Activity3: It individuates operators for each systematically important process and assigns to them suitable adjustments for their BCP
Activity4: Each Operator designates a single person responsible for extend the BCP and ensure compliance
Activity5: It implements adjustments to BCP and ensure continuous compliance
Check by Bank of Italy
fail
pass
Activity6: It prepares and trasmits to Bank of Italy a yearly report
Systematically important process management

Fig. 4.16. Application Model for Cyber Systemic Risk: Sistemically important processes Management Process

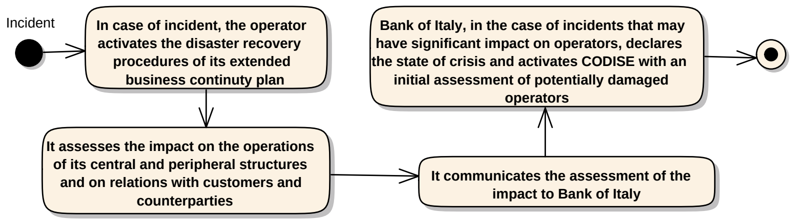

Incident
In case of incident, the operator activates the disaster recovery procedures of its extended business continuty plan
It assesses the impact on the operations of its central and peripheral structures and on relations with customers and counterparties
It communicates the assessment of the impact to Bank of Italy
Bank of Italy, in the case of incidents that may have significant impact on operators, declares the state of crisis and activates CODISE with an initial assessment of potentially damaged operators

Fig. 4.17. Application Model for Cyber Systemic Risk: Incident Management Process

who begins this activity with an initial assessment of potentially damaged operators.

Internal and external impact assessment (**UC4**). In the occurrence of crisis, the actor "systematically important processes manager", prepares the assessment of the impact on operations of its central and peripheral structures and of the current relations with customers and counterparts.

Fig 4.16 and Fig. 4.17 report two activity diagrams that respectively model the process of handling cyber systemic risk for important banking processes and the process of managing the possible knock-on effect of a cyber systemic incident inside the important processes of an operator.

Results and Conclusions for the case study

We tested the suitability of GOReM for modeling the context of business continuity in of a Bank and the requirements related to handling of the cyber systemic risk as regulated by the guidelines issued by the Bank of Italy. In the complex and touchy scenario of banking, the relevant models have been defined and graphically represented. GOReM allowed to easily identify stakeholders, roles and specific goals from which the main use cases and processes, have been derived. The result of this study is a requirements specification that may be employed as a good starting point for the devising a global approach towards the management of the cyber systemic risk in the financial and banking domain. In fact, it gives the adequate planning independence to the single bank, but under the riverbed of the constraints dictated by a supervisor authority, like the Bank of Italy for the Italian banking operators. Moreover, this vision could scale at the behavior of a node inside a bigger network, where other nodes are the other Bank of the other European Nations. In turn, this model might be applied to a coordinated European supervision institution, e.g. the European Systemic Risk Board. Even more, it is also desirable to scale worldwide under the control of a global authority, who might coordinate business continuity and disaster recovery for preventing and managing a cyber systemic risk, for the global financial world.

As a final consideration, special attention should be paid to the time needed for a given operator to react to a cyber incident: be "promptly" might not be an adequate answer. Two observations come from this modeling experience:

(i) Cyber systemic effects are here handled by a central authority, who in this case is the Bank of Italy, that establishes the state of crisis and manages the know-on effect on other operators. This centralization may result in a waste of time even though prudential politics suggest that this is a good strategy.
(ii) Time of response to a state of crisis that is communicated after some "hours" (see use case **eUCe**) might be a very large interval of time, especially at a worldwide level, compared to the speed of cyber threats.

A possible solution might be in modifying the hierarchical organization in a more horizontal and collaborative one, which might come only from common decided rules and regulations. However, Cyber Systemic Risk treatment in Europe and worldwide is nowadays urgent. While business areas are already supervised, the supervision agreement for network and information security is still a work in progress. This is a big delay for cyber systemic risk that must be regained soon.

4.4.3 Case Study: Cyber Security - Compliance analysis

Cyber Security Compliance Analysis is the process of assessing whether the behavior of an IT system or application conforms or not to the cyber security rules and regulations in force. This assessment can be offered as a service by exploiting available Cloud technologies [93] and indeed is one of the services classified by the Cloud Security Alliance (CSA) as part of the Security Information and Event Management (SIEM) category of the SecaaS domain [92, 88]. The definition and implementation of this typology of Cloud services [91] are challenging activities due to the complexity of both the reference business domain and the Compliance Analysis Services to be provided themselves. In this context, the case study applies GOReM to model the conceptualization and subsequent implementation of Cyber Security Compliance Analysis services[89]. In particular, after identifying the stakeholders involved in the SecaaS domain along with their goals, the SIEM scenario is considered and two main application scenarios for the Cyber Security Compliance Analysis are identified: (i) assessment of an existing IT system/application, aiming at producing a report about the conformance of the system with the cyber security rules and regulations in force as well as a set of indications on its possible enhancement; (ii) individuation of functional and non-functional requirements of an IT system/application under development that need to be met by its design.

The case study models and discusses both the application scenarios and, thanks to GOReM, provides a clearly understanding of them so as to guide their concrete implementation.

The requirements specification for these scenarios emerged as a real need inside the project Cyber Security [67]. The exploitation of GOReM allows to highlight in a lean, yet accurate, way some peculiars requirements that have to be taken into account when designing a Cloud service for Security Compliance Analysis; specifically, the importance to include and control the requirements coming from rules and regulations governing the external and the internal world of the considered context for every specific country where the cloud service is provided.

This allows to grasp and understand the many and complex facets to designing, building and offering a cloud service, because of the numerous, different and ever changing legal aspects [90] that have to be considered by service providers wanting to sell their services to worldwide customers.

Modeling Compliance analysis as a service

This section shows how the requirements of a security compliance analysis service [94] are achieved by applying GOReM. By following the CSA classification of security-related cloud services [92], SecaaS is identified as the general reference context and SIEM [93] as the scenario where the application model of compliance analysis finds its place.

Context Model: Security as a Service (SecaaS)

In this phase, stakeholders and softgoals relevant to the SecaaS context are identified. Specifically, SecaaS involves several aspects that can be clustered

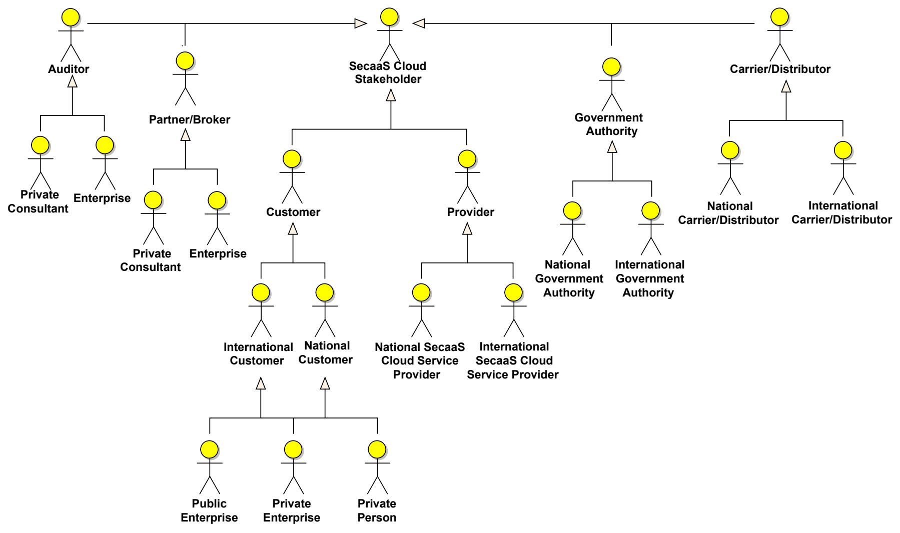


Fig. 4.18. Context Model: SecaaS Stakeholder Diagram

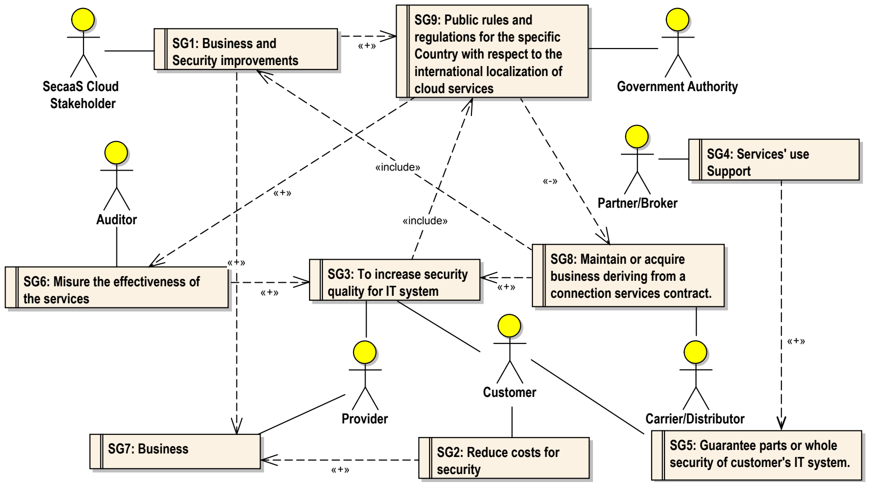

SecaaS Cloud Stakeholder
SG1: Business and Security improvements
SG9: Public rules and regulations for the specific Country with respect to the international localization of cloud services
Government Authority
Partner/Broker
SG4: Services' use Support
Auditor
«+»
«include»
«-»
«include»
SG6: Misure the effectiveness of the services
SG3: To increase security quality for IT system
SG8: Maintain or acquire business deriving from a connection services contract.
Provider
Customer
Carrier/Distributor
SG7: Business
SG2: Reduce costs for security
SG5: Guarantee parts or whole security of customer's IT system.

Fig. 4.19. Context Model: SecaaS Softgoal Dependency Diagram

into the following categories [93]: Cloud Identity and Access Management (IAM/IDaaS), Data Loss Prevention, Web Security (WAF), Email Security, Cloud IDS/IPS, Security Information and Event Manager (SIEM), Encryption, Business Continuity and Disaster Recovery (BCDR), Network Security, Cloud Database Security, Virtualization Security.

Fig. 4.18 shows the Stakeholder diagram obtained for the SecaaS context. This diagram illustrates the hierarchy of the stakeholders that was identified according to the National Institute of Standards and Technology (NIST) of the U.S. Department of commerce [91]. The considered stakeholders and the corresponding tasks are as follows.

SecaaS Customer: uses a service offered by a SecaaS provider. He might be either a Private Person, or a Public/Private Enterprise.

SecaaS Provider: develops and offers SecaaS services.

SecaaS Partner/Broker: offers service customization on behalf of the provider and supports the customer during the usage of the service. He might be a Private Consultant or an Enterprise.

Government Authority: manages the interaction among providers, customers and partners/broker with the aim to control eventual law violation and/or providing new discovered needs of rules and regulations.

SecaaS Cloud Auditor: a third part whose business is to assess the security compliance of the services provided in cloud. He might be a Private Consultant or an Enterprise.

SecaaS Cloud Carrier/Distributor: connectivity provider between cloud providers and consumers.

SecaaS Cloud Stakeholder: the generic stakeholder at the head of the hierarchy.

It is important to distinguish between national and international stakeholders, because cloud services can refer to a single Nation, more than one, or worldwide, with the consequence that different rules and regulations have to be respected.

For each stakeholder, his main Softgoals are established. Softgoals, are general goals, often non-functional, that a Stakeholder wants to reach. These Softgoals are resumed in the *Softgoal dependency diagram*, as reported in Fig. 4.19. Note that, UML stereotypes "+", "-", "includes" and "extends" , are used to express specific relationships among Softgoals. In particular, "+" means that the achievement of the source Softgoal has a positive effect on the accomplishment of the target Softgoal, "-" means that the achievement of the source Softgoal hinders the accomplishment of the target Softgoal, "include" means that the achievement of the source Softgoal implies, at least in part, the achievement of the target Softgoal, and "extend" means that the source Softgoal extends the target Softgoals.

The identified Softgoals are the following:

- SG 1. *Business and Security improvements*: for a SecaaS stakeholder, offering and using new type of cloud services can be a good opportunity for business and/or security improvements.
- SG 2. *Reduce costs for Security*: customers desire to reduce costs for all services used in Cloud.
- SG 3. *Increase the security quality for IT system*: it is a need for the provider, as a specialized people that continuously study and increase security techniques and strategies for reducing risks connected to security of IT.

- SG 4. *Services' use Support*: supporting activity can be internal to the provider or referred to a third part (a partner/broker) whose aim it to support the correct exploitation of the services defined within the contract between customers and providers.
- SG 5. *Guarantee parts or whole security of customerś IT system*: a customer, through system logging and event monitoring, wants to enforce compliance assessment methods and obtain reports about it.
- SG 6. *Measure the effectiveness of the services*: an Auditor estimates the quality of the SecaaS services.
- SG 7. *Business*: the main and general goal for the SecaaS provider is to improve his business.
- SG 8. *Maintain or acquire business deriving from a connection services contract:* the Carrier/Distributor stakeholder is an ISP enterprise, which can be also the same provider that subscribes a contract to allow Internet communication in cloud between the provider and the customer. This is a very hazardous service to be considered as a basic service in the contract of SecaaS between provider and customer.
- SG 9. *Public rules and regulations for the specific Country with respect to the international localization of cloud services*: supports the binding among providers, customers and partners to assure that rules and regulations constantly cover coherently emerging needs and their applications. For example, rules for privacy may differs from a country to another.

According to GOReM, the rules and regulations falling in the SecaaS context are identified and divided into four categories.

In particular, in this SecaaS context the focus is on rules and regulations related to data security. Specifically, as shown in Table 4.9, rules and regulations are collected and classified as Code, Reference, Type, along with their range of validity (i.e. Location/Adopters), helping the provider to understand National or International location for its service coverage. Note that, because of space reasons, Table 4.9 shows only a subset of rules and regulations, for the given context model.

Scenario Model: Security Information and Event Manager (SIEM)

In this phase, a specific scenario of interest, among the SecaaS context, is identified and better defined. In particular, the Security Information & Event Manager (SIEM) scenario is considered.

SIEM, as defined in the Gartner glossary (http://www.gartner.com/it-glossary/), "is a technology supporting threat detection and security incident response through the real-time collection and historical analysis of security events from a wide variety of event and contextual data sources. It also supports compliance reporting and incident investigation through analysis of historical data from these sources. The core capabilities of SIEM technology are a broad scope of event collection and the ability to correlate and analysis events across disparate sources".

Table 4.9. CONTEXT MODEL: SecaaS CONTEXT RULES AND REGULATIONS DIAGRAM

Code	**Rules and Regulations**	**Type**	**Location/Adopters**
001_PCI3.0_2013_INT	Data security standard - requirements and security assessment procedures, v. 3.0, 11/01/2013, PCI security standards council, LLC	**Policy**	International, principal adopter: American Express, Discover Financial Services, JCB International, MasterCard e Visa Inc.
002_ECBfinal_2013_Europe	Recommendations for the security of Internet payments (Final version after public consultation),v.final,1/1/2013, European Central Bank	**Int. law**	Europe
003_law196_2003_ITA	Code relating to protection of personal data, law 196/03 30/06/2003, Authority for the Protection of Personal Data	**National law**	Italy
004_NISTrev.4_2013_USA	Security and Privacy Controls for Federal Information Systems and Organizations, rev.4, 04/01/2013, National Institute of Standards and Technology (NIST)	**Int. law**	USA
005_PCI2.0_2013_INT	PCI DSS Cloud Computing Guidelines, v. 2.0, 02/01/2013, PCI Security Standards Council, LLC	**Best practice**	International, principal adopter: American Express, Discover Financial Services, JCB International, MasterCard e Visa Inc.
006_NIST_2011_USA	Guidelines on Security and Privacy in Public Cloud Computing, 12/01/2011, National Institute of Standards and Technology (NIST)	**Best practice**	USA
007_ISO27001_2005_ INT	ISO/IEC 27001 Information Security Management System (ISMS).	**Policy**	International

The scenario model phase includes the identification of the roles that are played by the Stakeholders, bounded in the considered SIEM scenario. The role that have been identified are reported in Table 4.10, in terms of their names, the corresponding stakeholders and a brief description.

The *Role Diagram* in Fig. 4.20, shows the proper relationships established among roles (white-head actors) and stakeholders (yellow-head actors) of the SIEM scenario. Each stakeholder, who plays some role in the modeled scenario, wants to reach some goal. Every goal of the stakeholder contributes to some of his softgoals. Goals and stakeholders' roles included in the SIEM scenario are shown in Fig. 4.21 and detailed as follows.

Goal 1: *Mastering Cloud Rules and Regulations*. This is the goal of a Government Authority who, in the role of *Master of rules and regulations for Cloud services*, should control and find deficiencies or contradictions among rules and regulations related to a specific Country and also with respect to the international context of the Cloud.

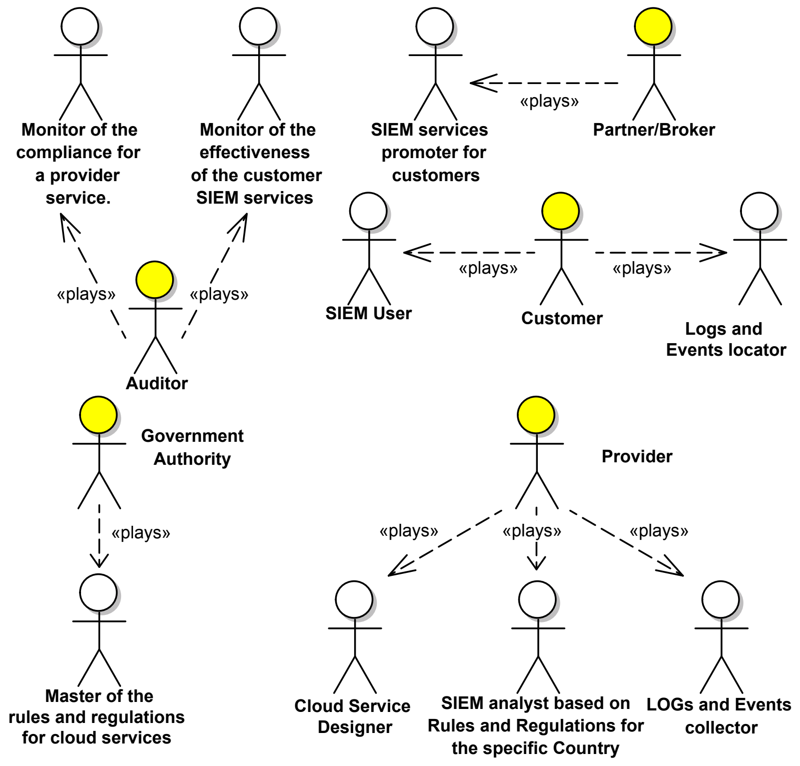

Monitor of the compliance for a provider service.
Monitor of the effectiveness of the customer SIEM services
SIEM services promoter for customers
«plays»
Partner/Broker
«plays»
«plays»
Auditor
SIEM User
«plays»
Customer
«plays»
Logs and Events locator
Government Authority
Provider
«plays»
«plays»
«plays»
«plays»
Master of the rules and regulations for cloud services
Cloud Service Designer
SIEM analyst based on Rules and Regulations for the specific Country
LOGs and Events collector

Fig. 4.20. SIEM Scenario Model: Role Diagram

Goal 2: *Analysis and Design of SIEM services for a customer*. This is the goal of a provider that, in the role of *SIEM analyst based on rules and regulations for the specific Nation*, should analyze and assure that every SIEM service does not contradict the Rules and Regulation mastered by the Government Authority of the Customer Country.

Goal 3: *Services parameters identification.* This is the goal of the customer that, in the role of *Logs and Events locator*, should locate all the Logs and Events inside the customer system that have to be considered by the involved SIEM services.

Goal 4: *Correct services utilization.* This is the goal of the customer, that in the role of *SIEM user* of the final service, should use the service as explained and established in a service user guide and/or as learnt by doing specific training.

Goal 5: *SIEM services parameters set up.* This is the goal of the provider that, in the role of the *LOGs and Events collector*, should insert all the Logs

Table 4.10. SIEM SCENARIO MODEL: ROLES

SIEM Stakeholder's Roles		
Role	*Stakeholder*	*Description*
LOG and Event locator	Customer	He is charge to locate and to provide to the LOG and Event collector all logs and events of the IT System under consideration.
LOG and Event collector	Provider	He is in charge to customize the security service by including in suitable structures the access to every considered customer's log and event.
SIEM services promoter for customers	Partner/ Broker	This can be a person or a software Help Desk tool, in charge to provide assistance to the customers for the correct usage of the security service. It could be located inside the provider structure or it can belong to a third part item operating for the Provider.
SIEM analyst based on rules and regulations for the specific Nation	Provider	He is an expert on security rules and regulations, in particular for Internet and cloud services. He is in charge to find, supervise and maintain the coherence among rules and regulations of the customer Nation, with respect to international rules and regulations. In the service customization phase he is in charge to suitable include the customer internal policy. He is fundamental during the construction, the customization and the tuning activities that can be necessary during the service life, because of changes or further security improvements.
SIEM User	Customer	He is in charge of the correct usage of the SIEM service.
Monitor of the effectiveness of customer services	Auditor	He is a third part who is in charge to control the effectiveness of the service usage in terms of security, e.g. in the context of a Public Administration.
Master of rules and regulations for cloud services	Government Authority	He is in charge to control that new written cyber security rules and regulations (i.e. laws, standards, guidelines, policies, ...) established by his Government Authority or other recognized organisms are coherent with respect to the existing ones. In addition, he proposes new rules and regulations whereas there exists a lack or a specific need.
Cloud service designer	Provider	He is in charge to design new SIEM cloud services for the provider. He has to consider the compliance of the new service to every security measure.
Monitor of the compliance for a provider service	Auditor	He is in charge to guarantee and control over time the security compliance of the offered SIEM service of the provider.

and Events pointed out by the customer system that have to be considered by the involved SIEM services.

Goal 6: *SIEM services utilization effectiveness monitoring*. This is the goal of an Auditor that, in the role of *Monitor of the effectiveness of customer services*, should control over the time that every SIEM service behaves, inside the customer site, as expected.

Goal 7: *Support the customer using the SIEM services*. This is the goal of a partner/broker, which either may belong to or works for the provider that, in the role of *SIEM services promoter for customers*, gives a help service to the customers for a correct usage of the SIEM services.

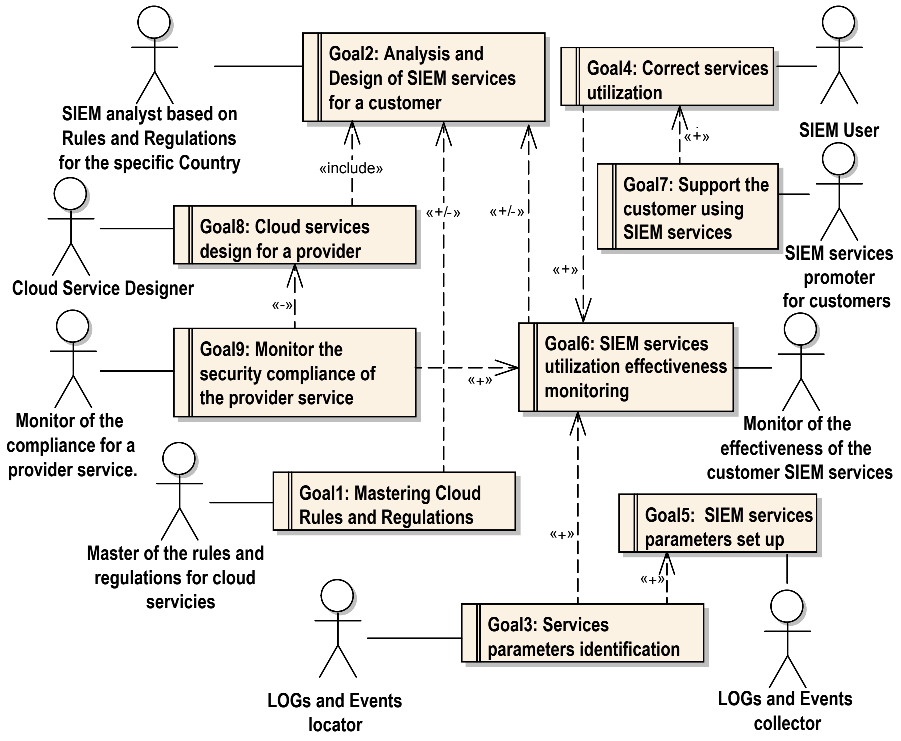

SIEM analyst based on Rules and Regulations for the specific Country
Goal2: Analysis and Design of SIEM services for a customer
Goal4: Correct services utilization
SIEM User
«include»
«+»
«+/-»
«+/-»
Goal7: Support the customer using SIEM services
Goal8: Cloud services design for a provider
Cloud Service Designer
SIEM services promoter for customers
«+»
«-»
Goal9: Monitor the security compliance of the provider service
Goal6: SIEM services utilization effectiveness monitoring
«+»
Monitor of the compliance for a provider service.
Monitor of the effectiveness of the customer SIEM services
Goal1: Mastering Cloud Rules and Regulations
Goal5: SIEM services parameters set up
«+»
Master of the rules and regulations for cloud servicies
«+»
Goal3: Services parameters identification
LOGs and Events locator
LOGs and Events collector

Fig. 4.21. SIEM Scenario Model: Goal Diagram

Goal 8: *Cloud services design for a provider*. This is the goal of a provider that, in role of *Cloud service designer*, designs every new cloud service, considering the compliance to the security rules and regulations.

Goal 9: *Monitor the security compliance of the provider service*. This is the goal of an auditor that, in the role of *Monitor of the compliance for a provider service*, guarantees that the cloud services offered by the provider are compliant to a defined set of security rules and regulations

Fig. 4.21 shows also the goal dependencies. For example, it is possible to note how Goal 7 helps reaching Goal 4, because an aid for the SIEM User allows to improve the correct usage of the service; while Goal 9 represents an obstacle in reaching Goal 8, because the monitoring of the security compliance can detect issues that are negative for the designer. As a consequence, the designer may need to modify the service design.

Before developing a SIEM in cloud, a company which is a Cloud service provider, should consider SWOT analysis. Table 4.11 shows a partial SWOT produced for the aims of the considered Cyber Security project. In the scenario modeling phase, domain experts are in charge of producing a *Rules and Regulations Diagram* containing a suitable selection from the overall list in-

Table 4.11. SWOT ANALYSIS

	Helpful	**Harmful**
	Strengths	**Weaknesses**
Internal origin	- The wide dissemination of Cloud services - Need to maintain high innovation of the provided services - Affiliation of a consistent set of companies and/or Individual customers - Possibility to experiment for real cases with security rules hiding - Leverage Public co-financing for the very timely topic - Competitive advantage, if there exists cooperation with the public authorities responsible for national security - Capillary dissemination of company's offices that allows easier access to information and service activation	- Excessive uncertainty on trust from potential customers, mainly contingent to national circumstances - Lack of cooperation of the competent authorities - The need for continuing education of internal experts prepared to prevent and remedy the attack - Skepticism and criticism by potential customers on the effectiveness of services in cloud and SIEM in particular - The need for a system of compliance reporting which should be easy to understand but complete and comprehensive for the various levels of responsibility in the organizational structure of the client - Cost/benefit ratio from the client point of view
	Opportunities	**Threats**
External origin	- Interest of the Public Administration to manage the security of information - Legitimacy of the system due to the spread and convenience of cloud services in both the public and private sectors -Increasing of Help Desk services also outsourced - High chance to establish a comprehensive supporting network of Partner / Broker, because of the reputation and trust of the company	- The complexity and contradictions of the system of rules and regulations at national level - Difficulty of reconciling the rules National and International ones because of location issues of the Cloud based modality - Size of the national and especially international competition - Stiffness of the Auditors that monitor the effectiveness of services - Continuous improvement at the global level of the cyber attacks - Security System of involved Carrier Distributor

dividuated in the context modeling phase. They are also in charge to raise warnings by highlighting rules and regulations which may affect each other. By referencing Table 4.9, Table 4.12 shows rules and regulations limited to SIEM Scenario for data cloud services, showing code and warning columns.

According to GOReM, in the third phase, different Application Models can be derived from the SIEM scenario model. The two following application models were produced:

1. *Security compliance analysis for a customer IT system*, aiming to analyze the compliance of the Information System of a customer enterprise;
2. *Security compliance analysis in the design of a new cloud service*, aiming at building cloud services compliant with specific security requirements.

Both the application models are presented in the next two sub-sections.

Table 4.12. SIEM SCENARIO: RULES AND REGULATIONS DIAGRAM

Code	**Warning**
001_PCI3.0_2013_INT	003_law196_2003_ITA
003_law196_2003_ITA	001_PCI3.0_2013_INT, 004_NISTrev.4_2013_USA, 006_NIST_2011_USA
004_NISTrev.4_2013_USA	003_law196_2003_ITA, 007_ISO27001_2005_INT
005_PCI2.0_2013_INT	006_NIST_2011_USA
006_NIST_2011_USA	005_PCI2.0_2013_INT, 003_law196_2003_ITA, 007_ISO27001_2005_INT
007_ISO27001_2005_INT	004_NISTrev.4_2013_USA, 006_NIST_2011_USA

Application Model 1: Security compliance analysis for a customer IT system

This Application Model is about the analysis of service compliance based on (i) a set of Logs and Events of the customer IT system, (ii) a set of a customer internal policies and (iii) national rules and regulations related to the specific customer's cyber security. In other words, it is assumed that there already exists, i.e. as another SIEM service, a defined security customer profile with a set of working system Logs and a set of Event catchers flows to be considered usual and then secure, as well as a dictionary of LOGs operations sequences and of events to be considered anomalies, and to be appropriately treated and signaled by some alerts. The compliance analysis service that is under definition, includes a) establishment and representation of the company internal policy, inside some 'structure' defined by the provider, tuned with all involved national rules and regulations (both laws and national policies) related to the security concerns of the customer. Such 'structure' is very important and a lot of design intelligence must be spent on it, as it ensures, for the most part, the success of the service; b) monitoring and detecting of the company daily IT operations life and its representation in a quite similar 'structure'. The compliance analysis will make a 'constant' and 'real-time' comparison between the two structures: the secure structure one represented by enterprise policies and national rules and regulations, and the structure detected in real time, during the life of the company. Note that, activities such as managing and treating LOGS and Events and their monitoring and alert system, belong to a different security service, supposed to be modeled in some other application scenario. The results of this compliance analysis service are a timely set of reports showing if there is compliance between the security mandatory behavior and the company life and in what level of confidence, possibly with evidence of a quantitative measure of the distance from full compliance.

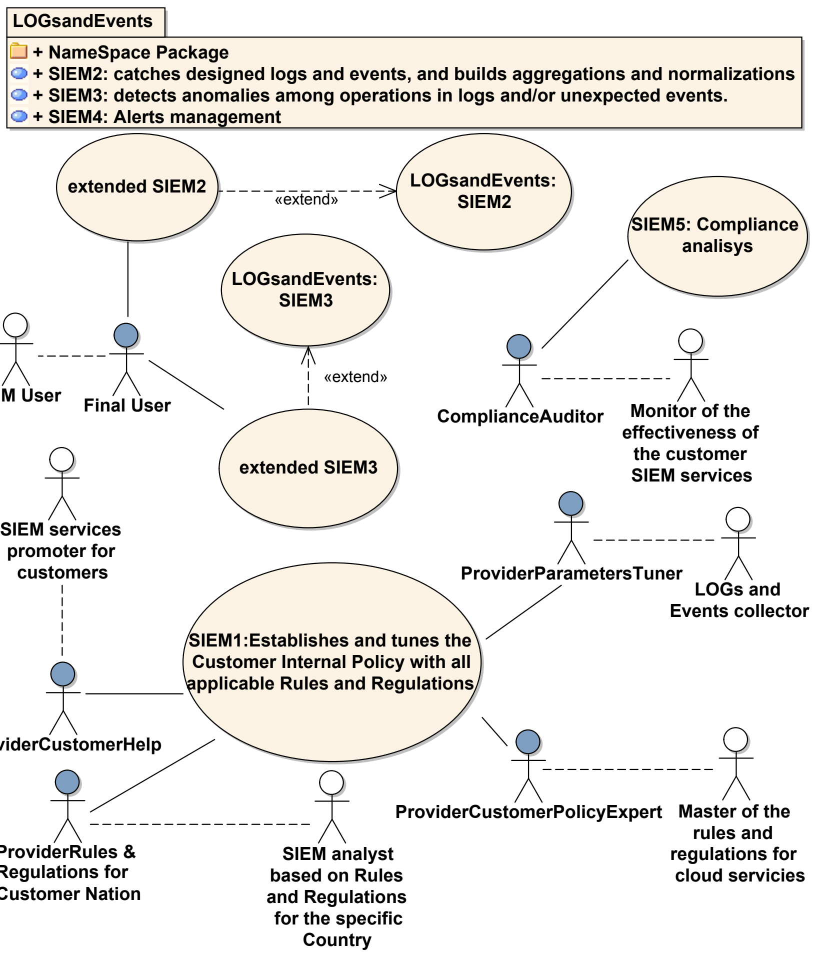

LOGsandEvents
+ NameSpace Package
+ SIEM2: catches designed logs and events, and builds aggregations and normalizations
+ SIEM3: detects anomalies among operations in logs and/or unexpected events.
+ SIEM4: Alerts management
extended SIEM2
«extend»
LOGsandEvents: SIEM2
SIEM5: Compliance analisys
LOGsandEvents: SIEM3
«extend»
SIEM User
Final User
extended SIEM3
ComplianceAuditor
Monitor of the effectiveness of the customer SIEM services
SIEM services promoter for customers
ProviderParametersTuner
LOGs and Events collector
SIEM1:Establishes and tunes the Customer Internal Policy with all applicable Rules and Regulations
ProviderCustomerHelp
ProviderCustomerPolicyExpert
Master of the rules and regulations for cloud servicies
ProviderRules & Regulations for Customer Nation
SIEM analyst based on Rules and Regulations for the specific Country

Fig. 4.22. Compliance Analysis Application Model 1: Roles, Actors and (Main) Use Case Diagram

In this phase, the specific actors involved in the Application Model under consideration and the use cases of interest, are identified and defined: only those Stakeholders playing a specific Role of interest for the Application scenario and their potential dependencies, if any, are selected and represented.

Fig. 4.22, shows use cases, actors and roles. The extended use cases are those mainly belonging to other SIEM application scenarios, that should be extended or integrated with new features. The identified use cases are detailed in the following.

1. *NameSpace LOGsandEvents package*. It contains use cases defined in a different Application model, and related to the localization and management of Logs and Events flow. In particular it contains:
   a) *Use Case SIEM2*, is related to the SIEM service, based on the collected logs and events established as parameters to be monitored during a dynamic temporal window of the IT system life. This service catches such data and performs suitable aggregations and normalizations to be organized in a structure as that used in SIEM1
   b) *Use Case SIEM3* is related to the SIEM service that detects anomalies on the normal flow of logged operations and/or normally expected events. This use case is based on a knowledge base of safe operations and events refined during a first period of testing in the customer's operational environment. In addition, this use case provides a structure compatible with that used in SIEM1 and SIEM2, to the detected anomalies.
   c) *Use Case SIEM4*, is responsible for alert management, selecting the appropriate level among: warnings and low/medium/high level alerts. It knows people to be alerted and the necessary blocking actions to be actuated.
2. *Use Case extended SIEM2*. It extends the Use Case SIEM 2 by organizing its output in a specific structure to be defined for the purpose of the compliance analysis.
3. *Use Case extended SIEM3*. It extends the Use Case SIEM3 by organizing its output in the structure for the purpose of the compliance analysis
4. *Use Case SIEM1*. It is in charge to establish a customer global policy, that verifies the legality and provides some tunes among the customer internal policy and the known mandatory laws, policies and best practices, to be considered.The result should be available in a structure established for the compliance analysis aims.
5. *Use Case SIEM5*. It is in charge of the compliance analysis between the Customer Policy provided in the structure resulting from SIEM1 and the detected behavior and eventual anomalies that the corresponding use cases (i.e., extended SIEM2 and extended SIEM3) have put in the corresponding structures. A specific elaboration based on some comparisons among these structures, will give, as a result, a set of well organized reports about compliance, with respect to the whole customer policy, possibly with evidence of the values that bring the performed compliance analysis to be different from full compliance and then requiring some tuning.

Application Model 2: Security compliance analysis in the design of a new cloud service

This Application Model faces with the compliance analysis of security services based on (i) the identification of rules and regulations (from now on called security laws) to which a new cloud service to be designed should be compliant,

(ii) functional and non-functional requirements, that the cloud service is meant to offer to its users. Through the Application Model, such requirements are expressed in terms of Use Cases and actors (equipped by processes, if needed) associated to a service able to guarantee the compliance of a newly designed cloud service with a set of security laws. This kind of issue has been studied in literature [89] as it is of great interest to have secure cloud services and also to guarantee continuously their compliance, according to the evolution of laws and regulations. This Application Model needs: (a) a deep study of the rules and regulations with which a new service to be designed has to be compliant, defining what it is called the security laws for the cloud service; (b) a comprehensive elicitation of the security requirements, that are indicated in each selected security law and that must be satisfied by every cloud service wanting to be compliant with it. Examples of security requirements belonging to a security law are: requiring to access to some data by using a password with a given length and special symbols composition, asking to have some specific permissions for using some functionalities, establishing the possibility to write down a log with a number of specific data, and so on. These security requirements may be optional or mandatory, depending on the security law type (e.g. policy, standard, best practice, law, etc); (c) a deep enumeration of the requirements, both functional and non-functional, that the cloud service is meant to provide and a specific set of attributes to be considered for each service. For each service requirement and security requirement, it is possible to ask for a complete compliance or a compatible level of compliance; (d) finally, a (semi) automatic verification of the compliance between the given service requirement (i.e. service attribute in [89]) and the given security requirements of the security law under compliance analysis, should provide a compliance evaluation. In particular, it should be analyzed if, and possibly to which level, on a user defined scale of measures, the designed service is compliant with the specific security law. The result of this application model gives to the designer indications on how to design the new cloud service in order to be compliant to a specific security law. A way to measure the distance between an optimum and the reached compliance should be one of the quality parameter of the cloud service.

In this phase, the specific actors involved in the Application Model under consideration and the use cases of interest, are identified and defined: only those Stakeholders' Roles of interest for the Application scenario are selected and represented. See figure 4.23

Fig. 4.24, shows use cases and actors. The identified use cases are detailed in the following.

According to [89], there are 2 main activities to be performed: (i) the *Support Activities* consist of a series of operations that allow to ensure the compliance of the service over the time, on the basis of the updates of the security laws; and (ii) the *Core Activities* to be carried out each time a service has to be developed and analyzed in terms of compliance with respect to the security laws. In this context, both the use case related to the security laws

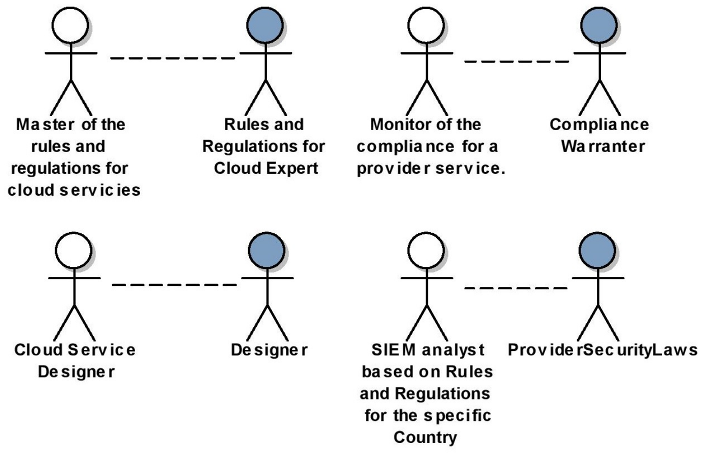

Master of the rules and regulations for cloud servicies
Rules and Regulations for Cloud Expert
Monitor of the compliance for a provider service.
Compliance Warranter
Cloud Service Designer
Designer
SIEM analyst based on Rules and Regulations for the specific Country
ProviderSecurityLaws

Fig. 4.23. Compliance Analysis Application Model 2: Actors and related Role Diagram

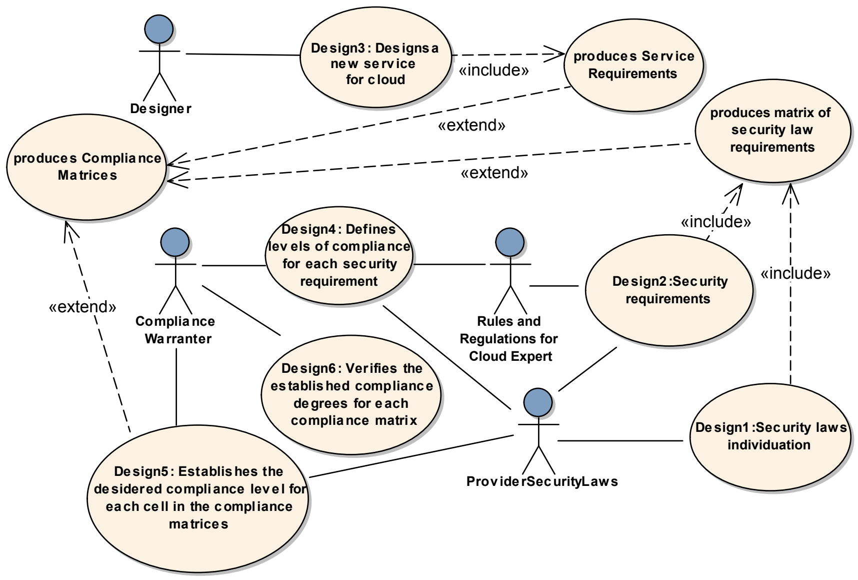

Designer
Design3: Designs a new service for cloud
«include»
produces Service Requirements
«extend»
produces matrix of security law requirements
produces Compliance Matrices
«extend»
«include»
«include»
Design4: Defines levels of compliance for each security requirement
Design2:Security requirements
«extend»
Compliance Warranter
Rules and Regulations for Cloud Expert
Design6: Verifies the established compliance degrees for each compliance matrix
Design1:Security laws individuation
ProviderSecurityLaws
Design5: Establishes the desidered compliance level for each cell in the compliance matrices

Fig. 4.24. Compliance Analysis Application Model 2: Main Use Case Diagram

individuations (Design1) and the Use Case related to study each security law requirements (Design2) fall among the *Support Activities*. In particular,

- *Design1.* Use Case related to the security laws individuations: this use case aims to identify rules and regulations related to the type of new cloud service to be designed, and that may be required to be fulfilled.

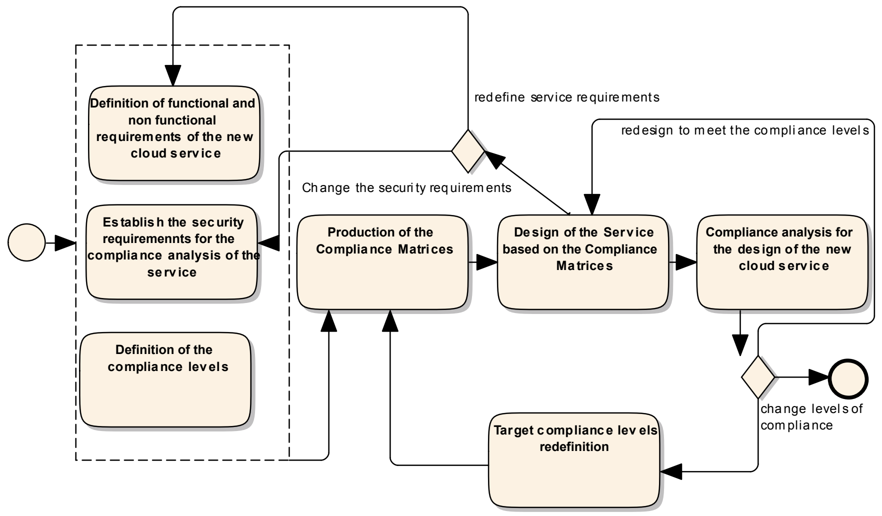

Definition of functional and non functional requirements of the new cloud service
Establish the security requirements for the compliance analysis of the service
Definition of the compliance levels
Production of the Compliance Matrices
Design of the Service based on the Compliance Matrices
Compliance analysis for the design of the new cloud service
Target compliance levels redefinition
redefine service requirements
Change the security requirements
redesign to meet the compliance levels
change levels of compliance

Fig. 4.25. Compliance Analysis Application Model 2: Process Diagram

- *Design2*. Use Case related to study each security law and to extract the law requirements, that the new service must satisfy to be compliant to the security law. This Use Case is performed by the provider and by an expert of rules and regulations of cyber security in cloud. Both Design1 and Design2 include the Use Case *produces matrix of security law requirements* as they allows to establish the security laws and for each security law the set of security requirements to be considered for the compliance analysis that, according to [89], might be done with a matrix.

These activities have to be performed not only during the initial setup of the service design, but also periodically, since new rules and regulations might be defined over the time and might require some re-design of (part of) the service.

The following use cases are strictly related to the specific service. Specifically,

- *Design3*. Use Case related to design a new cloud service. This is an activity that in the native approach described in [89] is not contemplated or actually it is considered given. In this work, this represents an important step, since it includes the Use Case *produces Service Requirements* that provides a detailed service description in terms of both functional and non-functional requirements that the service should fulfill.
- *Design4*. Use Case related to establish for each security requirement a set of compliance levels. For example: 'Useful', 'Recommended', 'Mandatory' 'Mostly Mandatory'. The levels are established by the expert of the security law and the compliance warranter. This is really critical activity,

since the different security levels may depend on several factors, such as the place or country where the service is provided, if it is available in Internet (public network) or in an Intranet (private network), and so on. By considering such different possibilities, specific Security Controls, needed to be considered during the realization of the service, are identified in this activity.

- *Design5*. This Use Case extends the compliance activity verification proposed in [89], based on the generation of Compliance Matrices, as stated by the Use Case *produces Compliance Matrices*. In particular, for each security requirement, it establishes the level of compliance that the provider wants, or has, to reach for each security requirements with respect to the considered service to be designed. For example, a security requirement considered 'Useful', might be 'Mandatory' for the provider. However, the provider might want modify its choice if this becomes difficult or too much expensive to reach. In this activity, the provider cooperates with the compliance warranter who does not allow, as an example, to consider 'Useful' a security requirement which for its compliant needs 'Mandatory'. All in all, (i) the uses cases *produce Service Requirements* together with *produces matrix of security law requirements* and with *Design5*, participate, then extend, the Use Case *produces Compliance Matrices*.
- *Design6*. Use Case related to the actual compliance assessment. In this activity, the compliance warranter evaluates the fulfillment of the requirements according to their compliance level of security as 'Not Applicable', 'Recommended but Unsatisfied', 'Recommended and Satisfied', 'Mandatory and Satisfied', 'Mandatory but Unsatisfied'. If some of them does not reach the expected/required compliance level, then it may happen that: (i)the Designer uses again the Design3 Use Case where the designer has to take care of updating the new service design in order to make it compliant or (ii) the ProviderSecurityLaw and the security warranter modify the expected compliance values established in the Use Case Design5 and then the Use Case Design6 allows a new compliance verification.

Fig. 4.25 shows the process of this compliance analysis. The activities in the dashed box aim to (i) define the service requirements, (ii) establish the security requirements and its compliance level (i.e. 'Useful', 'Recommended', 'Mandatory'), for each security law to which the new service has to be compliant, (iii) define the compliance levels that the new service should reach for each service requirement and each security requirement, with respect to the compliance levels of the corresponding security requirements. All these activities are supported by a multi-dimensional model that enables to keep track of rules and regulations over the time, how they are linked to the service and, in the end, to verify its compliance by exploiting OLAP analysis. The output of this approach consists of a set of matching tables, called compliance matrix. Each compliance matrix describes if the service requirements (functional and non-functional) satisfy one specific security requirement. Based on this model, the service designer and compliance analyst define the compliance degree of a service to certain security requirements. Possible values might be 'Not Applicable', 'Recommended but Unsatisfied', 'Recommended and Satisfied', 'Mandatory and Satisfied', 'Mandatory but Unsatisfied'. Finally the analysis is done by using (semi-)automatic tools or even manually. At this point there will be an overall evaluation, and it is necessary to take a decision manly for value 'Mandatory but Unsatisfied': changing level of compliance wanted, if possible and desired by the provider, or redesign the service, with a better carefulness. From there, if redesign fails, it is possible to consider to eliminate/modify some of the service requirement or eliminate/modify some security requirements, if possible. Then the production of the compliance matrices may restart.

This Application Model is conceived as a continuous process. This means that, even after the new cloud service is developed and it is in operation, when a new security law is introduced (or a previous one is updated), the cloud service is kept under monitoring in order to check whether it is still compliant with the new regulation or not. In the latter case, the compliant/new version of such service needs to be designed, implemented and delivered again, in order to replace the previous one.

Results and Conclusions of the Case Study

This work aimed to highlight some peculiars requirements to be taken into account when modeling a Cloud service for Security compliance analysis. The adopted methodology, i.e. GOReM, showed the importance to include and control those requirements coming from rules and regulations governing the external and the internal world of the considered context, for every specific country where the cloud service is provided. This allows to understand how complex can be to build and to offer a service in cloud, because of the numerous, different and ever changing legal aspects, to be considered when a cloud service provider has to sell his service to worldwide customers. This is especially true for the cyber security domain which is extremely important in the cloud context. The analyzed services for compliance analysis, had to take into account many stakeholders and the roles they play in order to achieve a good model for the application scenario. In addition, it is important to focus on the strengths, weaknesses, opportunities and threats that can affect a cloud business scenario and can guide the cloud provider to decide to go ahead or to stop spending on further investments on a specific service. Yet, the two application models show how a goal-oriented but graphical, still formal, methodology, helps clarifying many aspects of the requirements engineering related to the applications to be developed, usually difficult to grasp with a traditional methodology based on many pages of boring, unmanageable, unmaintainable documentation.

A final note refers to a possible tool development to support the methodology: with the aim to trace the evolution, some shared existing policies of naming and versioning method/tool, for every model (i.e. context, scenario, application) and every of its work-products, must be used. In addition, some configuration management tool should help in maintaining the requirements evolution of the different application models. This allows to know exactly for each application model, both the scenario model and the context model referred to. In addition, whichever refinement for a model created in one of the three GOReM phases, must produce a new model referring the model it wants to improve. Moreover, each application model, if implemented, should refer to its requirements' artifacts and releases currently in operation. Concerning the validation of the two modeled application scenarios, future work should deal with their implementations and in their validation by means of an experimentation on some real case, which would find space inside the next activity plans of the Cyber Security District [67].

4.4.4 Case Study: Cyber Security - Systemic Risk in Payment On Line

In this section, an extension of GOReM with a methodology of simulation called RAMSoS, is presented in the context of the Cyber Systemic Risk for the District of Cyber Security, as published in the conference CIISE 2016 [30]. A more complete presentation of the extended methodology for System Risk Analysis is in[116].

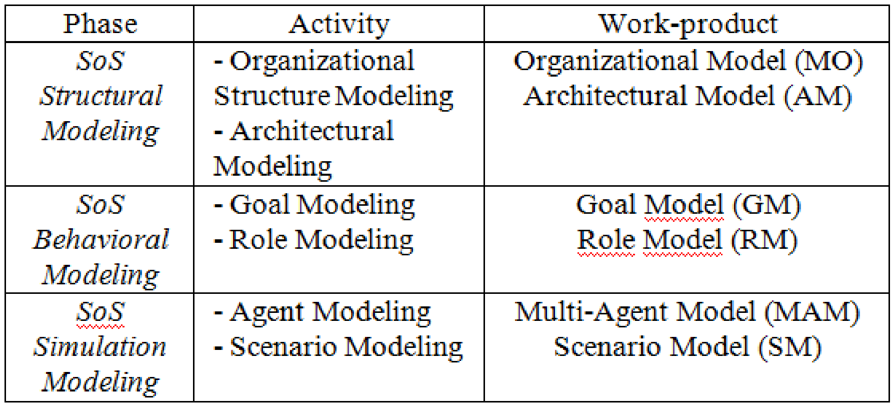
In the following, a case study related to a Payment On line (PEO) service is shown.

Case Study: An extension of GOReM with RAMSoS for on-line payments

The Cyber Systemic Risk has been treated in a previous case study in relation to the area of banks and, in particular, to the way as this topic is treated by Bank of Italy. Here the systemic events are more generally referred to systems/services, as could be a system of payment which is of great interest for Poste Italiane.

The causes that lead to systemic events reside primarily in the influence that the various actors in the network have with each other; furthermore the systemic importance of the various actors is not determined by their size, but by the correlation degree among them. Similarly, it is not always true that a negative event of large dimensions can be always defined as systemic. In fact, the propagation mechanism can be realized not only through the direct exposure to a negative event caused by the shock, but also indirectly.In this context, it is interesting to understand how it is possible to model actors and factors arising from systemic risk, in order to fully consider them in the different phases of risk analysis.

Table 4.13. PHASES, ACTIVITIES AND WORK PRODUCTS OF RAMSoS



Phase	Activity	Work-product
SoS Structural Modeling	- Organizational Structure Modeling - Architectural Modeling	Organizational Model (MO) Architectural Model (AM)
SoS Behavioral Modeling	- Goal Modeling - Role Modeling	Goal Model (GM) Role Model (RM)
SoS Simulation Modeling	- Agent Modeling - Scenario Modeling	Multi-Agent Model (MAM) Scenario Model (SM)

In this context, the case study aims at investigating in such direction by exploiting engineering tools for representing relationships among systems/services and observing their behavior. Specifically, the adoption of the Requirements Engineering approach, combined with Modeling and Simulation techniques, are used to catch how and which entities of the overall system influence the operation of the entire system and, as a consequence, the evaluation of the Systemic Risk. In particular,the combination of GOReM with the RAMSoS method [95], natively conceived for supporting systems dependability analysis, is provided. Such combination enables the modeling and the evaluation of the Systemic Risk by exploiting an agent based simulator that has been ad-hoc implemented.

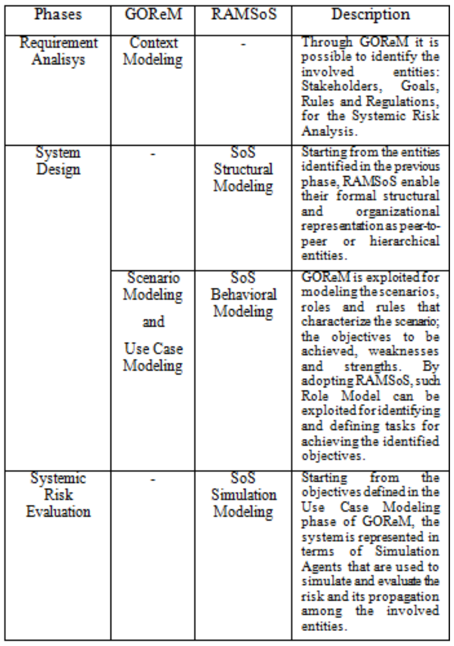
Combining RAMSoS and GOReM

RAMSoS is an agent-based method that aims at supporting the dependability analysis of Systems of Systems (SoSs). It is conceived as an extension of RAMSAS [96], a model-based method for the reliability analysis of systems through simulation, based on UML/SysML for modeling the system structure and behavior, and on well-known simulation platforms, such as Mathworks Simulink and OpenModelica. The RAMSoS method defines three main phases, which in turn are divided into activities (see Table 4.13).

A full description of RAMSoS can be found in [95]; whereas table 4.14 reports the main phases (Requirement Analysis, System Design, e System Risk Evaluation) that are identified by combing GOReM and RAMSoS for modeling the systemic risk aspects and supporting its analysis through agent-based simulation.

In particular, some phases are complementary and some others use the output produced from a method as input for the other one. The resulting method is exemplified through a case study.

Table 4.14. GOReM EXTENSIONS THROUGH THE RAMSoS METHOD



Phases	GOReM	RAMSoS	Description
Requirement Analisys	Context Modeling	-	Through GOReM it is possible to identify the involved entities: Stakeholders, Goals, Rules and Regulations, for the Systemic Risk Analysis.
System Design	-	SoS Structural Modeling	Starting from the entities identified in the previous phase, RAMSoS enable their formal structural and organizational representation as peer-to-peer or hierarchical entities.
	Scenario Modeling and Use Case Modeling	SoS Behavioral Modeling	GOReM is exploited for modeling the scenarios, roles and rules that characterize the scenario; the objectives to be achieved, weaknesses and strengths. By adopting RAMSoS, such Role Model can be exploited for identifying and defining tasks for achieving the identified objectives.
Systemic Risk Evaluation	-	SoS Simulation Modeling	Starting from the objectives defined in the Use Case Modeling phase of GOReM, the system is represented in terms of Simulation Agents that are used to simulate and evaluate the risk and its propagation among the involved entities.

The case study under consideration falls within the online payment services and, in particular, exemplifies the approach based on the combination of GOReM and RAMSoS adopted for systemic risk analysis and applied to a service of Electronic Payment Online (PEO) of Poste Italiane. The main objectives of this study are: (i) The assessment of systemic risk, when there is a dysfunctional behavior in one of the service components, in terms of the propagation of a disservice among other components; and (ii) impact of a service failure to the services.

Service Description, Risk Factors and Involved Actors

The PEO service is based on two services: SMS Notifications and Payments and Transactions, both designed to be used from smartphones and tablets. SMS Notifications allows to receive SMS messages on transactions made on a bank account or by "PostePay" card; whereas Payments/Transactions allows bank transfers, payment of bills, money transfer via MoneyGram, PostePaytop up, or balance check and movements.In this context, the aim of this experience is the identification and the analysis of systemic risk factors linked to the PEO service. In particular, the risk of success or failure of the PEO service relies on two complementary services: SMS Notifications and Payments and Transactions, plus the IT Internal Infrastructure.A preliminary analysis shows that the SMS Notification service is linked to the Mobile Service Provider whose goal is to notify the user of the transaction (payment, charging, etc.). Whereas the Payments and Transactions is related both to the Web Service Provider that provides access to the Intranet/Internet and the Energy Provider that supports the entire infrastructure with the electrical service. An additional risk factor is related to the underlying IT infrastructure (hardware, servers, etc.). In this context, the following risk factors: IT Internal, Outsourcing and Contracts, Infrastructure Upstream, are identified and described along with the related actors. In particular: (i) the ITInternal risk relies on the reliability of the Internal IT infrastructure; (ii)the Outsourcing and Contracts risk depends on the WebServiceProvider for supporting the monetary transactions; (iii) whereas Infrastructure Upstream risk is related to the availability of both the mobile notification service offers by the MobileServiceProvider and the electricity provided by the ElectricityProvider. Furthermore, since the approach requires the input of information related to potential risk groups (e.g. contract type, involved partner), for each actor, the following risk groups have been identified:

- ITInternal-Infrastructure: Good, Standard, Poor;
- WebServiceProvider: High, Medium, Low;
- Energy Provider: High, Standard;
- MobileServiceProvider: HighLevelOfService, StandardLevelOfService;
- SMS Notification: Good, Low;
- Payments and Transactions: LowRisk, HighRisk.

The output of this analysis is the risk level of the PEO service according to the different levels of risk of the other services. It is estimated in terms of success and failure, where Success = 1-Failure, therefore Success + Failure = 1. The higher the percentage / value of the Success, the lower the level of risk associated to it and as a consequence the lower the risk level of the PEOservice. Vice versa the lower the percentage of the Failure variable, the lower the level of risk associated to it, and then the lower the risk level of the PEO service. In the following, the extended version of GOReM is employed for the modeling and evaluating the system above described.

Context Modeling

As described above, the context falls within the scope of online payment systems where, through a website, it is possible to make purchases, transfers of money etc. A particular important diagram of GOReM is the Dependency diagram 4.26 that at the same time allows to represent the stakeholders, the goals that they are meant to achieve and dependencies (such as conflicts/extensions and so on among goals).

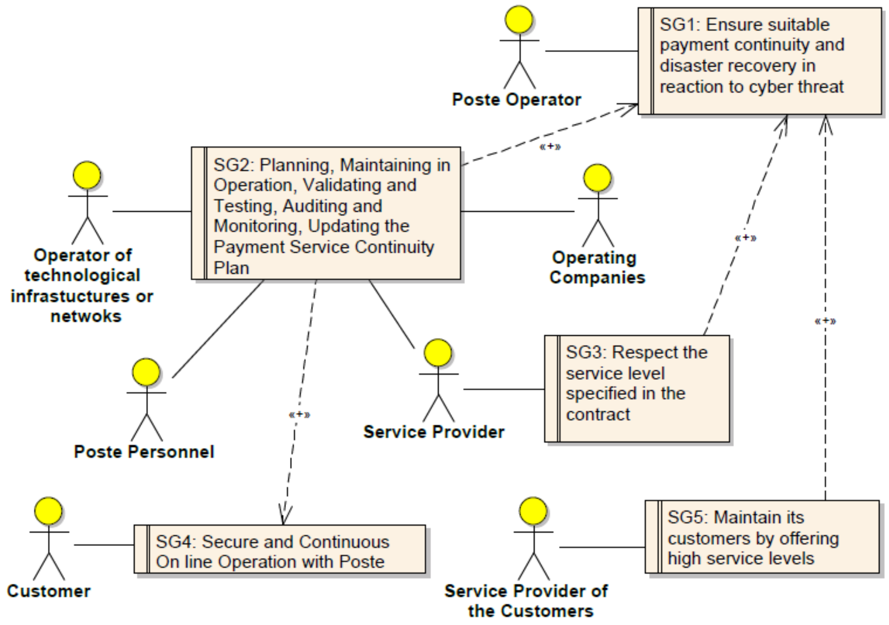

Poste Operator
SG1: Ensure suitable payment continuity and disaster recovery in reaction to cyber threat
«+»
SG2: Planning, Maintaining in Operation, Validating and Testing, Auditing and Monitoring, Updating the Payment Service Continuity Plan
Operator of technological infrastuctures or netwoks
Operating Companies
«+»
«+»
Poste Personnel
Service Provider
SG3: Respect the service level specified in the contract
«+»
Customer
SG4: Secure and Continuous On line Operation with Poste
Service Provider of the Customers
SG5: Maintain its customers by offering high service levels

Fig. 4.26. PEO Context Model: Stakeholders, Softgoals and Dependency Diagram

Scenario Modeling

In this phase of the method, as it is shown in Figure 4.27, both the roles played by the stakeholders in each specific scenario are identified,and the goals related to each identified role are highlighted. Furthermore the dependencies among the Goals are shown in Table 4.15.

Application Modeling

The application model allows to describe, with more details, a particular instance of the scenario under consideration. Specifically, Figure 4.28 represents the case of failure of a service to third parties (i.e. the notification via SMS) necessary for the provision of online payment services, and the impact on the other users who use the service, possible costs (impact) for the failure to provide the service.

Simulation based evaluation

Once the model and relationships among actors and their goals are well described and defined, it is possible to use simulation to provide an assessment about what can happen into an application scenario according to specific inputs to the system. In the following, first a statistic based tool is exploited

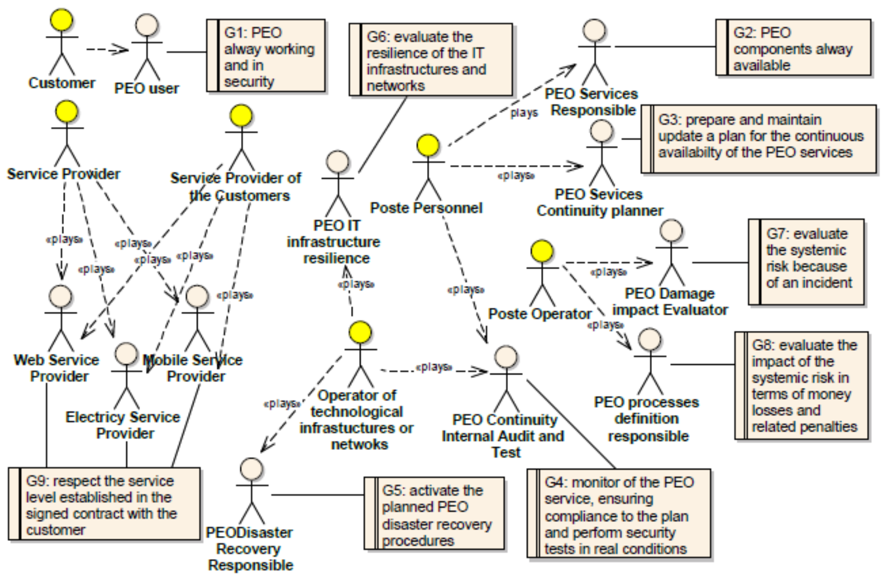

Customer
PEO user
G1: PEO alway working and in security
G6: evaluate the resilience of the IT infrastructures and networks
PEO Services Responsible
G2: PEO components alway available
plays
G3: prepare and maintain update a plan for the continuous availabilty of the PEO services
Service Provider
Service Provider of the Customers
PEO IT infrastructure resilience
Poste Personnel
«plays»
PEO Sevices Continuity planner
PEO Damage impact Evaluator
G7: evaluate the systemic risk because of an incident
Poste Operator
Web Service Provider
Mobile Service Provider
Electricy Service Provider
Operator of technological infrastuctures or netwoks
PEO Continuity Internal Audit and Test
PEO processes definition responsible
G8: evaluate the impact of the systemic risk in terms of money losses and related penalties
G9: respect the service level established in the signed contract with the customer
PEODisaster Recovery Responsible
G5: activate the planned PEO disaster recovery procedures
G4: monitor of the PEO service, ensuring compliance to the plan and perform security tests in real conditions

Fig. 4.27. PEO Scenario Model: Stakeholders, Roles and Goals

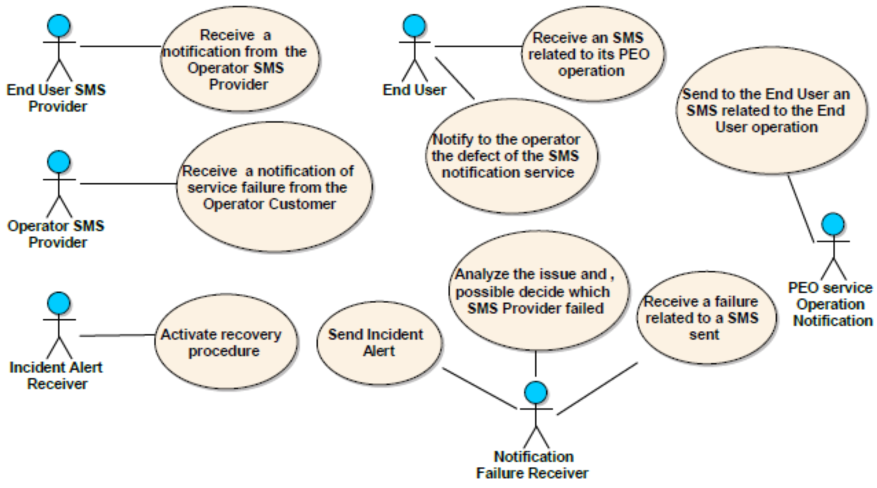
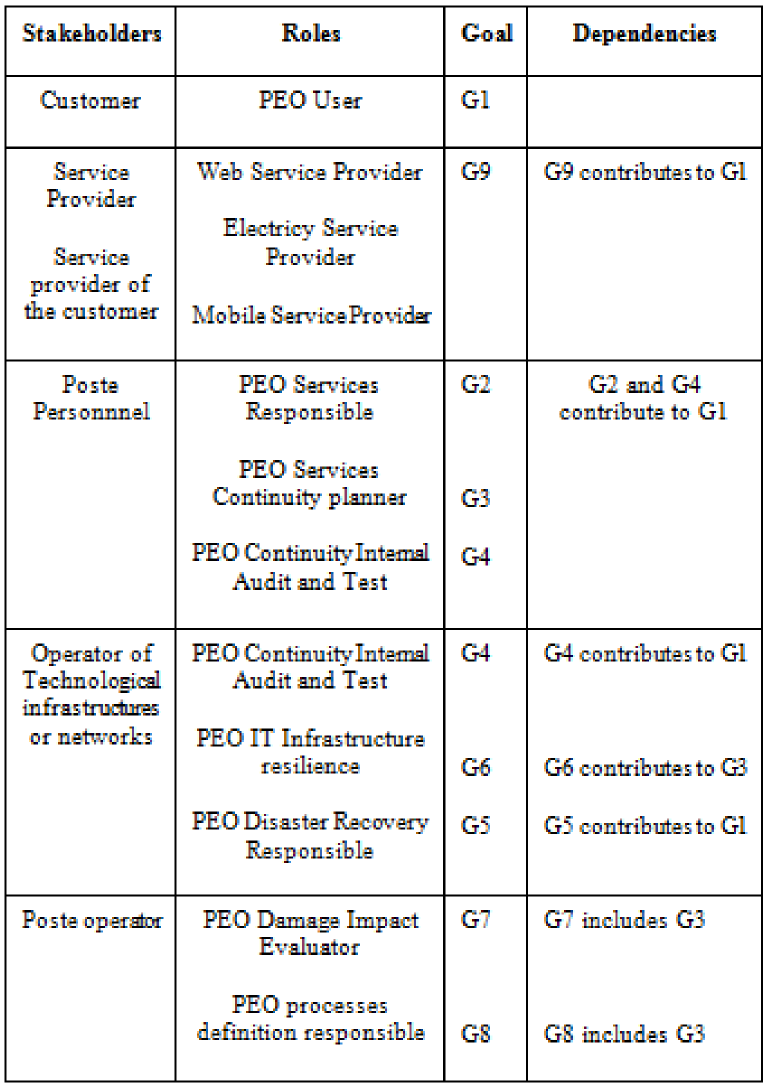

End User SMS Provider
Receive a notification from the Operator SMS Provider
End User
Receive an SMS related to its PEO operation
Send to the End User an SMS related to the End User operation
Operator SMS Provider
Receive a notification of service failure from the Operator Customer
Notify to the operator the defect of the SMS notification service
PEO service Operation Notification
Incident Alert Receiver
Activate recovery procedure
Send Incident Alert
Analyze the issue and , possible decide which SMS Provider failed
Receive a failure related to a SMS sent
Notification Failure Receiver

Fig. 4.28. PEO Notification SMS Application Model: MAIN Use Cases

for a static analysis (i.e. GeNIe) and then a more dynamic is adopted (i.e. REActor).

Table 4.15. STAKEHOLDERS, ROLES, GOALS AND DEPENDENCIES



Stakeholders	Roles	Goal	Dependencies
Customer	PEO User	G1	
Service Provider Service provider of the customer	Web Service Provider Electricy Service Provider Mobile Service Provider	G9	G9 contributes to G1
Poste Personnnel	PEO Services Responsible PEO Services Continuity planner PEO Continuity Internal Audit and Test	G2 G3 G4	G2 and G4 contribute to G1
Operator of Technological infrastructures or networks	PEO Continuity Internal Audit and Test PEO IT Infrastructure resilience PEO Disaster Recovery Responsible	G4 G6 G5	G4 contributes to G1 G6 contributes to G3 G5 contributes to G1
Poste operator	PEO Damage Impact Evaluator PEO processes definition responsible	G7 G8	G7 includes G3 G8 includes G3

A statistics-centered approach

GeNIe (Graphical Network Interface) is a development environment for the creation of decision models [112]. It is presented as a graphical user interface of SMILE, a platform-independent library that implements functions for the execution and analysis of probabilistic and decision models, such as Bayesian networks, used to make probabilistic reasoning in decision-making situations

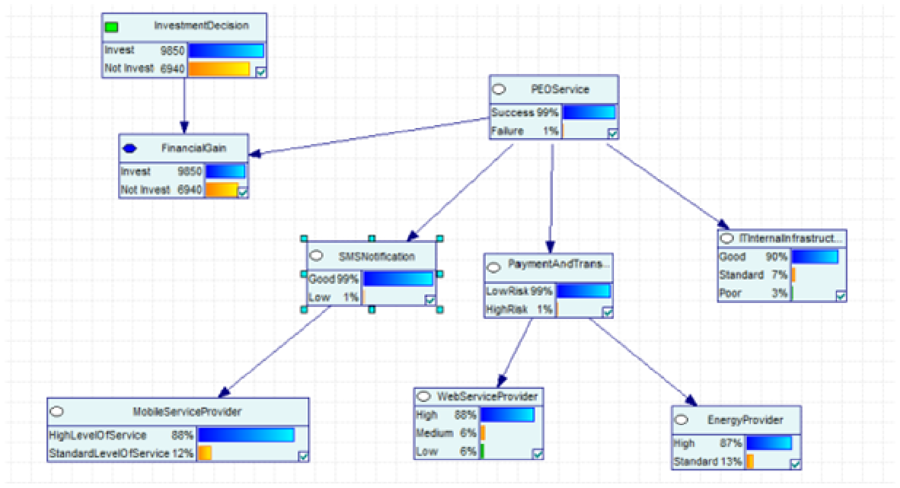


Fig. 4.29. Low Risk of the PEO service

under uncertainty. Starting from different contractual terms of the services described above, it is possible to obtain an assessment in terms of the level of success (and complementary to the failure level) of the PEO service, which in turn can be associated with a level of risk. From the experience of the domain experts of Poste Italiane, the following percentage range is used:

- Success >90% then LowRisk
- 89% >= Success >70% then MediumRisk
- Success <= 70% then HighRisk

A first example is shown in Figure 4.29. By considering a combination of services, based on the percentages shown in each block, the probability of success is 99%, which means a LowRisk. The diagram is also enriched with to additional blocks: FinancialGain and InvestmentDecision, lead the decision maker to take decisions about the quality of the services to be subscribed. In this case, as shown by the "InvestmentDecision" and "Financial income" blocks, it is convenience to invest (with a gain of €9850) by subscribing services with such quality parameters indicated, compared to not invest (€6940).

Conversely, considering a low level quality of the SMS Notification service, and by also subscribing a low level quality of the WebServiceProvider service, the level of risk spreads systematically on the Payments and Transactions services by influencing drastically the PEO service. In fact, the success rate drops to 63%, which means "HighRisk". See Figure 4.30.

An Agent-based approach

This second approach is centered on a reference framework, called ReActor, an object oriented framework based on discrete-events simulation [97]. The

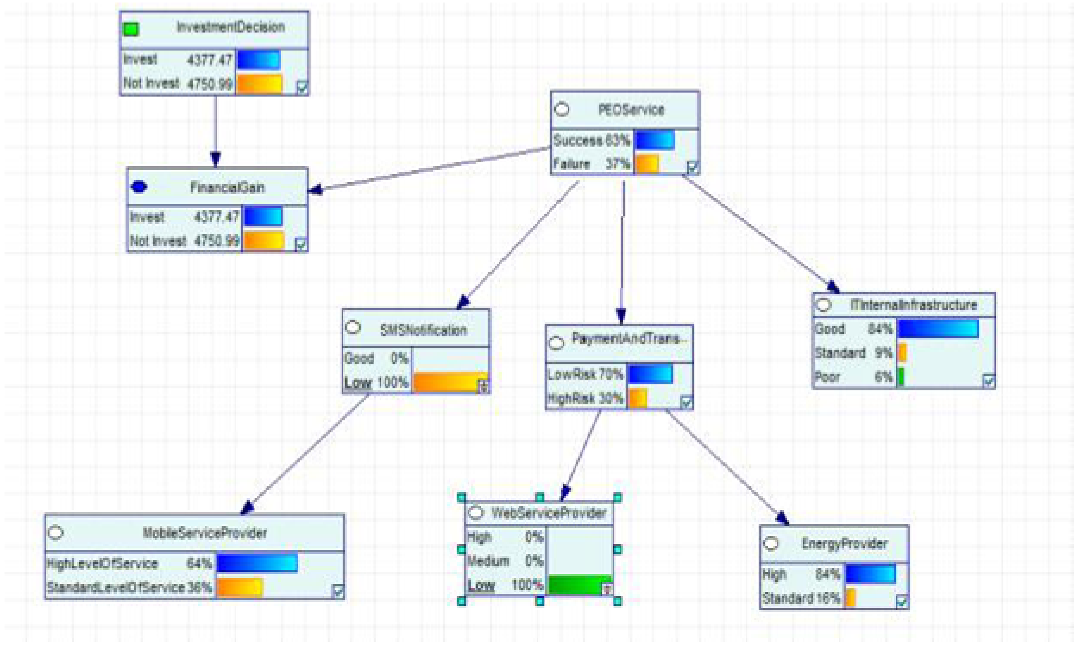


Fig. 4.30. High Risk of the PEO service

reference model adopted for the definition and the development of the agent-based simulator for the analysis of the systemic risk is represented in Figure 4.31. In particular for each static blocks represented in Figure 4.30, a specific ReActor entity is defined. Then a behavior is associated to each of them, based on the follow four main actor models:

- ServiceModel: this model is employed for services belonging in the specific scenario to be analyzed; its aim is to provide the service associated to it;
- AttackModel: this model is adopted for modeling attack scenarios and related typologies of attacks respect to a specific ServiceModel;
- RecoveryModel: it aims to model policies and countermeasures in order to make more resilient a specific service when some anomalies occur;
- ObserverModel: it is employed for monitoring specific properties of interest which are strictly related to a specific service; it aims to collect information of specific properties, locally at service level or globally at scenario level..

Such models have been implemented by extending the above mentioned agent-based framework by mapping them as agents, that is, autonomous entities each of which has its own behavior. In particular, the ServiceModel is mapped as ServiceAgent; the AttackModel as an AttackAgent; the RecoveryModel is mapped as a RecoveryAgent and the ObserverModel as an ObserverAgent.

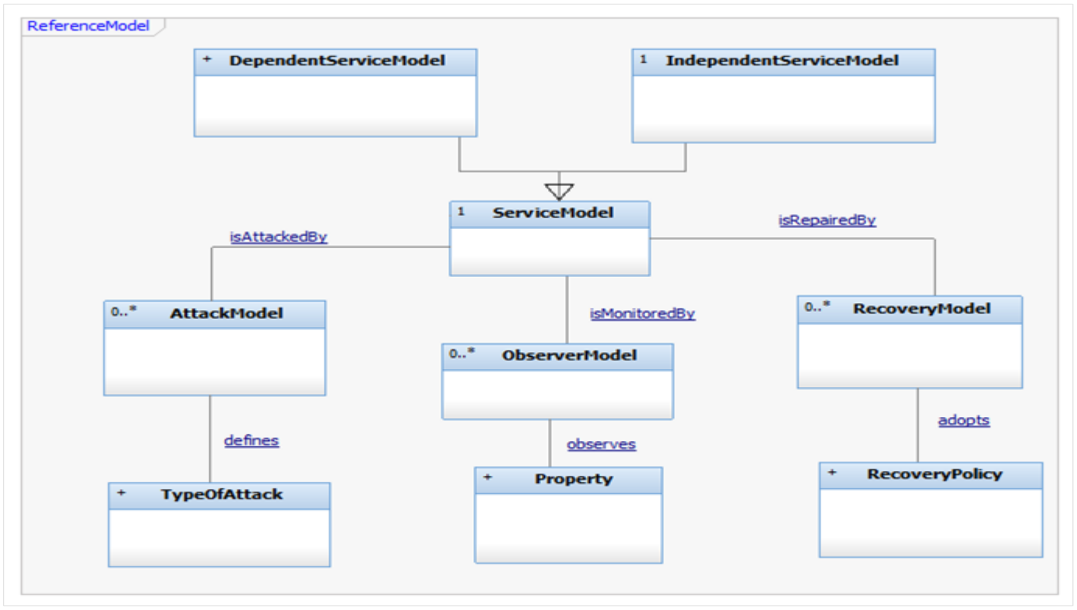


Fig. 4.31. High Risk of the PEO service

Such agents and their behaviors are achieved by implementing and extending the basic class ActorBehavior of the Reactor framework, which in turn, has been also defined as Observable. Consequently all agents that are introduced in the system, and that extend ActorBehavior, are potentially trackable. Whereas, the ObserverModel and, consequently, the ObserverAgent, has been marked as Observer, that is with the ability to monitor other agents. Finally, the behavior of each agent is characterized by different types of Message, that can respectively transmit, receive and handle in order to enable the communication with the other agents. As an example, the diagram in Figure 4.32 shows the behavior of the ServiceAgent, defined as a state machine.

In particular, when the simulation starts, the status of ServiceAgent becomes Working.This means that the ServiceAgent is doing its job/delivering the service correctly.When an anomaly occurs, the state Working can get two types of events: ServiceFailure and ServiceFailurePropagation. Such events change the status of ServiceAgent into NotWorking, that, in turn, is defined in terms of two sub-states DirectFailure and IndirectFailure. In particular, when the ServiceFailure event occurs, the status NotWorking declines into the state of DirectFailure. This means that the failure of the service was due to internal factors of the service. This condition triggers the propagation of the failure by a ServiceFailurePropagation event to the services that depend from the ServiceAgent; thismeans that a service of the system, could receive a ServiceFailurePropagation event, which turns its status into NotWorking and specifically into the IndirectFailure status. This implies that its failure

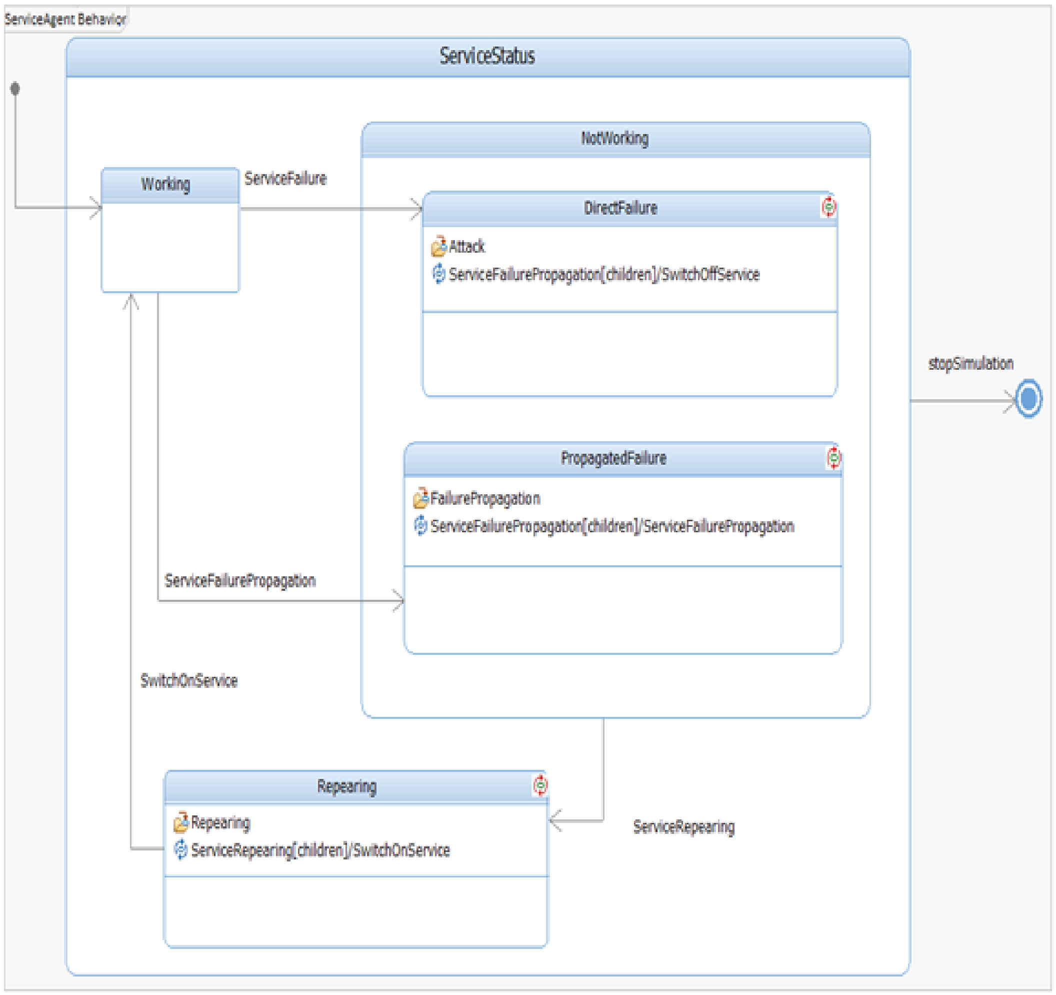


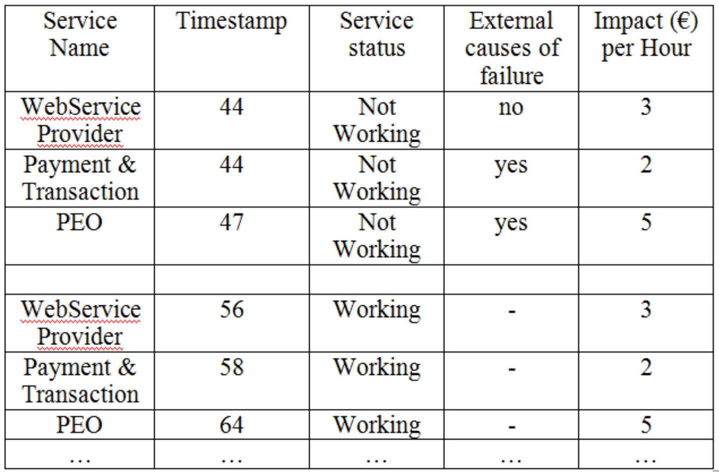
Fig. 4.32. High Risk of the PEO service

was due to a failure propagated by third parties on which it depends.Finally, from the NotWorking status, the ServiceAgent can receive a ServiceRepearing event that brings it into the Repearing status. This allows to recover/restore the ServiceAgent and propagate this information among the other services depending on it, so as to make them all Working again.

Results and Conclusions of the Case Study

From the analysis conducted on this case study, it is clear how the quality of services level and the involved system infrastructure (internal or third-party),

Table 4.16. STAKEHOLDERS, ROLES, GOALS AND DEPENDENCIES



Service Name	Timestamp	Service status	External causes of failure	Impact (€) per Hour
WebService Provider	44	Not Working	no	3
Payment & Transaction	44	Not Working	yes	2
PEO	47	Not Working	yes	5
WebService Provider	56	Working	-	3
Payment & Transaction	58	Working	-	2
PEO	64	Working	-	5
…	…	…	…	…

strongly influence the success or the failure for the delivery of a service. In this case the use of a low quality Notification service is critical. As a consequence, the choice of a good MobileServiceProvider, combined to a Medium/High quality of the WebServiceProvider is essential for making the system more resilient. Indeed, (i) in the first scenario, involving the deployment of services with a high level of reliability, or in the second scenario, combining medium-quality services, the system operates to keep resilient in presence of permanent failures, or temporary blackout, of some involved entities; (ii) instead, the second scenario highlights the high risk due to the strong dependence on entities that provide low robust / reliable services.

Whereas from the conducted study based software agents, other useful and more dynamic information are gathered from the simulation for each service involved (see Table 4.16); for example: if a service is available (working) or unavailable (not working), the time when the failure of a service happened (timestamps), if the cause of the failure is due to external factors, the impact (e.g. in terms of money) per unit of time (e.g. per hours).

Assessing the Systemic Risk is crucial for organizations so as it is that of the operative one.

Systems Engineering approaches and Simulation techniques can provide viable solutions to assess the Sistemic Risk.

The combination of the GOReM methodology and the RAMSoS method has been proposed for Systemic Risk Assessment through simulation: (i) Modeling of SoSs structure and behavior;(ii) Explicit representation of dysfunctional behavior; (iii) Evaluation of different risk scenarios through simulation; (iv) Quantitative and qualitative risk assessment also in combination with classical analysis techniques.

5

Long-lasting EIS with a new approach: ResDevOps

In this chapter the ResDevOps approach, aimed at reaching long-lasting EIS, as presented at the 24th International Conference on Requirements Engineering (RE'16), which was held in Beijing in 2016 September from 12-16, is presented.

It is worthwhile to note that the International Requirements Engineering Conference (RE) is one of the largest annual software engineering conferences. It has an 'A' rating from the Australian Ranking of ICT Conferences and an 'A1' rating from the Brazilian ministry of education(see IEEE Requirements Engineering RE in http://www.conferenceranks.com).

ResDevOps approach is aimed to extend the lifecycle of an Enterprise Information System (EIS) in this era, where agile development approaches are used with the attention to customer satisfaction, achieved primarily via continuous delivery and via rejection of any upfront documented requirements activity [16].

ResDevOps includes the usage of GOReM as methodology for the EIS Requirements Engineering and inter-operates with any agile development approach and with the trendy DevOps philosophy. The positive effects on improving the customer satisfaction, through a longer life of its costly, complex, ubiquitous and not easily substitutive EIS, is shown by some real use cases. In fact, the approach has been experimented in real industrial research projects that are here presented.

Note that the usage of GOReM is raccommended for its peculiar features discussed in Chapter 4, even though it is not mandatory and a different RE methodology might be used, provided that the idea behind ResDevOps approach is preserved.

5.1 Introduction to the problem statements

Software engineering aims to maximize customer satisfaction, by devising and exploiting suitable methodologies, processes, tools and frameworks for improving the software development activities. Lean Agile methodologies[98], the DevOps philosophy[99, 100], Agile Unified Process[101] and frameworks such as SAFe® (Scaled Agile Framework®)[102] and DAD (Disciplined Agile Development)[12] represent some successful efforts in this direction.

The rationale is that, if the customer is satisfied with the IT System a provider has developed, i.e. it well responds to the user's requirements, then the IT System will have a longer life: the customer saves and the provider gains. A longer life of an IT System or, more generally, of an Enterprise Information System (EIS), is then a goal for both the customer and the provider.

Today EISs are increasingly complex, pervasive and costly, so it is not feasible to substitute them in order to keep pace with innovations and ever-changing requirements. Therefore, achieving good evolvability and consequently a longer life, is mandatory.

Starting from some concrete experiences, it is argued that Requirements Engineering has still much room of improvement towards assuring longer life for EISs.

DevOps culture and tools already foster these objectives, by applying the agile approach to the iterative process for software application development. The resulting applications are concurrently integrated in the customer EIS, which then results enriched by new functionalities. However, agile-after-agile developments lead to an EIS that is: (i) not governed, because the different customer references lose the overall requirements control and (ii) out-of-date in its technology and in the supported requirements (both functional and non-functional) because, over-the-time, it becomes inadequate to support the actual business needs. Nowadays it is quite impossible to substitute an old unmanageable EIS as a whole, because of several reasons, where costs and user habits are only some important components. Then, while lean and agile methodologies, including DevOps approach, are very suitable and successfully useful, it is important, at the same time, to be able to control the overall EIS supported requirements and to continuously integrate it with research and innovation results. Embedding a continuous concurrent research process into the chain of a modern agile software engineering framework, where DevOps is a definitive positive component, is a way to achieve long-lasting EISs.

The Application Lifecycle Management (ALM) approach, described in 3.5.1, is, currently, the best known tentative to sketch a framework to supporting the management, development and operation of every application working in an Enterprise. However, (i) the ALM architecture refers to each application of the Enterprise and do not manage at all the EIS, composed by many applications, where integration, distribution and different versions of the same application, complicated the whole EIS; (ii) the development line could be whichever SDLC. Thus, in particular nowadays, the development line of ALM refers to whatever of the agile development methods, that reject any documented requirements for an application and, even worse, for the requirements supported by the EIS as a whole.

These considerations have lead to the definition of the ResDevs approach that, cooperating with DevOps, originated the ResDevOps framework described in this chapter.

For its aims, ResDevs needs a methodology to represent the EIS and to trace its evolutions, in terms of its requirements, implemented functionalities and involved scenarios. The choice was GOReM, the goal-oriented approach already discussed , whose initial unique aim was to avoid the waste of investment in large industrial research projects, often co-funded by government, where some confusion usually arises. In fact, in these projects, the objective to well managing elicitation, specification and analysis of requirements, is not properly attained, because of misunderstandings among project stakeholders that mainly have different languages. Thus, GOReM well applies to the numerous stakeholders present in an enterprise where a complex ubiquitous, pervasive, often distributed, multipurpose and highly interoperable EIS, operates.

5.2 ResDevOps = ResDevs+DevOps

In order to guarantee a long-lasting EIS, large enterprises should build and maintain up-to-date the requirements model of their internal information systems[16]. Also, on the basis of some real experiences, co-funded research projects, aimed to provide enterprises innovation, should not stop with the scheduled end of projects, but they should be embedded inside the EIS requirements, considered for innovation proposals and maybe, for future development projects. More generally, a research and innovation project, aimed to support the business context of the EIS, should be always kept alive. The goal of this project is to gather innovation thrusts and to transform them into new requirements that should be incrementally integrated in the current requirements model. It should be the main source of innovation proposals for the enterprise managers. Each new business scenario or application, embedded in the EIS, thanks to specific development projects, has to be the result of an update of the requirements model. New projects are generated either by requests of the EIS's users, asking for new functionalities based on their experience in the operating environment, or by specific improvement/innovation projects.

To support all these activities, the *ResDevs* approach is proposed. ResDevs is based on three main pillars:

1. an adequate *requirements model* of the EIS;
2. a *continuous research process*;
3. a support for the *incremental addition* of new requirements coming either from user needs or from innovation thrusts.

ResDevs, once in a while, generates new developments projects.

ResDevOps approach joins the just defined ResDevs process with the agile world of DevOps discussed in section 3.2.

In DevOps approach, each new project is developed by using agile methods. In particular, according to Fig.3.4: code is built, usually as result of *sprints*, implementing parts of the requirements defined by *user stories* [106, 107]; after the test activity, a release of the application is produced and deployed in the target IT system, where it is installed and put in operation; then, a monitoring activity of the released application detects the user satisfaction; each defect produces an improvement of the user stories that better fits the user needs; if small changes are needed, a new plan for coding, build, test and release activities is provided, in compliance with the agile approach, and the loop of Fig. 3.4 goes on. On the contrary, if important changes in the requirements need to be taken into account, the DevOps side resorts to the ResDevs side of the ResDevOps process.

ResDevOps could gather new ideas to solve the issue caused by the continuous user's feedbacks in the DevOps approach: it may happen that important requirements changes need considerable additional efforts and costs. Thus, whereas small changes in requirements will continue to be managed as usual, inside the agile with DevOps approach or the agile release train of SAFe®, big changes in requirements have to be carefully managed, by looking at the real implications on the overall EIS requirements model, calling for mindful decisions supported by the ResDevs approach.

Figure5.1 summarizes the ResDevOps approach, resulting in the integration of ResDevs infrastructure with DevOps infrastructure. It adds the possibility of:

1. Mastering the overall EIS requirements, by means of a set of models that guide the ResDevs infrastructure, concurrently with the ongoing development activities, [108, 109] managed by the DevOps infrastructure;
2. Studying and experimenting possible evolutions of the system subparts, by keeping active a group of cooperating stakeholders, including researchers, new-technology experts, end users and business managers, to evaluate the possibility to define new development projects.

The aim is to achieve a long lasting EIS, by continuously innovating and improving the functional/technological components of the EIS, by controlling the introduction of new requirements asked by the end users and new business development support asked by the EIS managers. All these activities, supported by the ResDevs infrastructure, may provide new scheduled developments inside the EIS managed, as usually, as agile development projects by the DevOps infrastructure.

The ResDevOps basic idea follows the Kaizen Japanese philosophy of continuous improvement. It is a way of thinking but it is also a rigorous and precise method, that uses statistical quality control and an adaptive framework. Managers and workers are focused on the idea of never being arrived. They want to take on every new challenge through a slow and continuous improvement of their business, of their supporting processes and of their IT system.

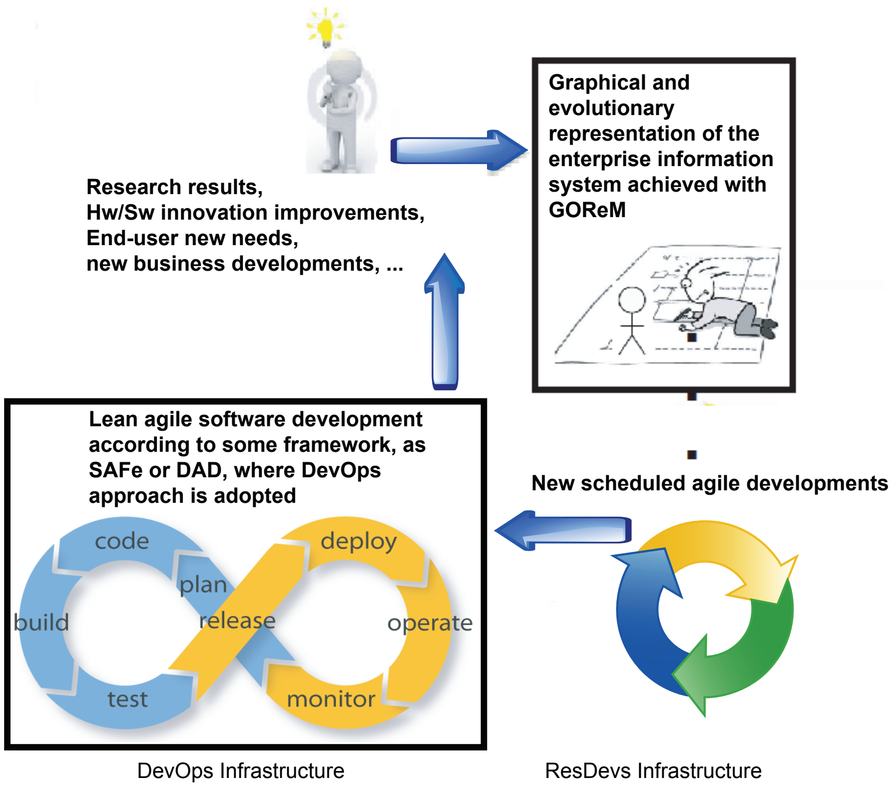

Research results,
Hw/Sw innovation improvements,
End-user new needs,
new business developments, ...
Graphical and evolutionary representation of the enterprise information system achieved with GOReM
Lean agile software development according to some framework, as SAFe or DAD, where DevOps approach is adopted
code
plan
deploy
build
release
operate
test
monitor
New scheduled agile developments
DevOps Infrastructure
ResDevs Infrastructure

Fig. 5.1. ResDevOps interoperation schema

In other words, continuous improvement in an Enterprise does not only mean to satisfy the current customer's requirements, actually the aim of DevOps, but also suggests to look forward to innovation through a continuous research process, managed by ResDevs.

As seen, in the previous chapter, GOReM methodology has been proven effective in mastering requirements of a large industrial research project concerning a scalable EIS of an enterprise. Therefore, it can be successfully employed as the grounding methodology for ResDevs. All the GOReM phases are iterative, traceable and ready to easily receive those changes that naturally arise during the EIS life.

It is important to understand the boundaries of the EIS, in terms of the involved context and business scenarios. Some application scenarios may be designed and proposed for new development projects and, if defined and accepted by the managers, they will be scheduled as new agile developments with the DevOps approach, where continuous integration, testing, delivery and users' feedbacks are compulsory features.

Research and innovation activities may start from a co-funded industrial research project or as an internal project responding to users or managers' needs. Project goals and stakeholders are identified and processed by the context modeling phase, and a set of business scenarios are identified. From each of them, application scenarios can be modeled, evaluated and possibly considered for the inception of new software development projects. ResDevs cycle concurrently goes on with its investigations and possibly generates new development projects.

ResDevOps has the great potential of ensuring the achievement of long-lived IT systems by obtaining: better return of investment for the customer; increase of software quality and user satisfaction; business improvement and provider's trust enhancement.

Ultimately, ResDevOps approach is the result of joining forces coming from both ResDevs and DevOps, even though they appear to have different goals and different speeds.

ResDevs needs that requirements specification is clearly documented and maintained up-to-date, as its main goal is to timely deliver new pieces of the whole EIS, thus allowing to have an always effective, modern and competitive EIS[110]. In this context, no development is needed, but the work of good researchers, modern technological enthusiasts, farsighted managers and good project evaluators and proposers. Here, budget, feasibility study, dynamic designing and risk management are important milestones in order to maintain previous investments and long lasting a complex EIS. DevOps is instead committed to the development of applications by using an agile and lean process. In fact, the continuous delivery and feedback are directed towards user satisfactions and then to achieve long lasting software. Software, which does not properly satisfy the end user, will be short lived.

5.2.1 ResDevOps Context Modeling

In this section, GOReM is applied for modeling the context where ResDevOps operates. ResDevOps itself uses GOReM in the activities occurring in the ResDevs infrastructure related to the EIS requirements model. Figure 5.2 shows the overall Stakeholder Diagram resulting from applying the GOReM's Context Modeling phase.

The box named '**Continuous Research with GOReM Methodology**' groups the stakeholders of GOReM itself used inside ResDevs: Context Modeler, Scenario Modeler and Application Modeler. Research Manager and Innovation Manager cooperate with GOReM stakeholders as partners in the research activities that span along the whole life of the EIS, by working on innovation of both technologies and functionalities. In addition, even DevOps Manager might cooperate with ResDevs when the changes of requirements asked by the customer are too costly to be managed by the usual DevOps loop and need some new project proposal formalization.

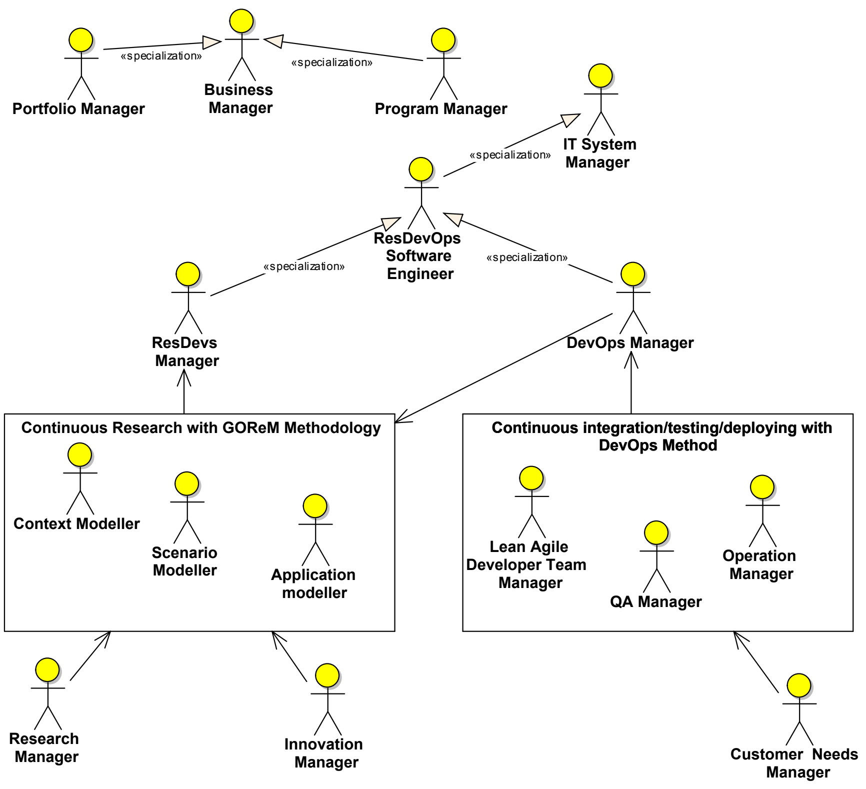


Fig. 5.2. Context Model: ResDevOps Stakeholder diagram

The box named '**Continuous Integration/Testing/Deploying with DevOps**', groups the DevOps stakeholders as represented in wikipedia where DevOps is the intersection of development (software engineering), technology operations and quality assurance (QA). Accordingly, Lean Agile Developer Team Manager, QA Manager and Operation Manager cooperate with the Customer's Needs Manager as partners in the success of an agile project development.

ResDevs Manager and DevOps Manager are respectively responsible of these two boxes and they are specialization of a generic ResDevOps Software Engineer, who has the control of the overall framework.

The IT System manager is the stakeholder who has the control of the overall EIS. He can decide, in agreement with the Business Manager, when and whether to submit part of the EIS under the innovation process. The Business Manager decides whether an innovative idea, coming from the Continuous Research process, can become a development project to be carried out in the DevOps framework and to which specific Program and Portfolio this new

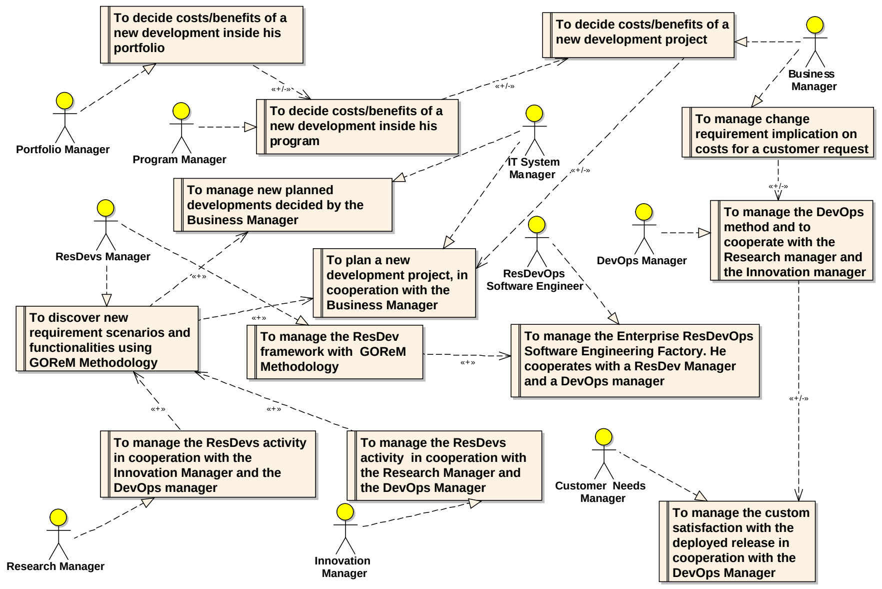

To decide costs/benefits of a new development inside his portfolio
To decide costs/benefits of a new development project
Business Manager
Portfolio Manager
Program Manager
To decide costs/benefits of a new development inside his program
To manage change requirement implication on costs for a customer request
IT System Manager
To manage new planned developments decided by the Business Manager
ResDevs Manager
To plan a new development project, in cooperation with the Business Manager
ResDevOps Software Engineer
DevOps Manager
To manage the DevOps method and to cooperate with the Research manager and the Innovation manager
To discover new requirement scenarios and functionalities using GOReM Methodology
To manage the ResDev framework with GOReM Methodology
To manage the Enterprise ResDevOps Software Engineering Factory. He cooperates with a ResDev Manager and a DevOps manager
To manage the ResDevs activity in cooperation with the Innovation Manager and the DevOps manager
To manage the ResDevs activity in cooperation with the Research Manager and the DevOps Manager
Customer Needs Manager
To manage the custom satisfaction with the deployed release in cooperation with the DevOps Manager
Research Manager
Innovation Manager
«+/-»
«+»

Fig. 5.3. Context Model: ResDevOps Dependency diagram

project may be included in agreement with the Program Manager and the Portfolio Manager, respectively.

Figure 5.3 shows the Softgoal Dependency Diagram, where boxes represent stakeholder's softgoals and links represent specific dependencies between them (i.e. contributes, denoted as +; hinders, denoted as -; contributes and/or hinders, denoted as +/-; include). Due to space limitation, the Softgoal Diagram in which they are defined, is not reported.

The ResDevOps Software Engineer manages the Enterprise Software Engineering factory with the contribution of the ResDevs Manager, who has, among others, the goal to manage the framework supporting the three phases of the GOReM methodology. The DevOps Manager's objective is to handle the DevOps loop of Fig. 3.4, but also to cooperate with the ResDevs stakeholders when the Customer's Needs Manager asks for significant changes of requirements. The ResDevs Manager, indeed, aims also to discover new requirement scenarios and new functionalities useful for the innovation and improvement of the EIS.

Whereas the other softgoals and their dependencies with the Stakeholders are described in Fig.5.3, it is highlighted the softgoals of the Business Manager. He is in charge of the evaluation of the costs/benefits of each new application scenario and he should decide whether this could become a new scheduled agile software development project. In addition, it is underlined the softgoals of Program Manager and Portfolio Manager, who decide about the inclusion of the new project inside their different businesses. Finally, the goal of the Customer's Needs Manager is to aliment the DevOps loop by requesting the adaptation of the previously established requirements, to be first suitably planned. We just recall that a new planning should imply a business evaluation by the Manager of the System deployment, who, in our case, is the DevOps Manager. He may interact with the Business Manager as the new plan could cause cost and scope variations.

5.2.2 Case Studies

The problem stated and the proposed ResDevOps approach, have been used in two large projects in [28] and [67].

In the first case study, related to the project DICET-INMOTO, GOReM was used to model the context of a Destination Management Organization (DMO) together with a set of scenarios and some application models. Some prototypes, related to the touristic events process management, were developed by OKT Srl, a SME involved in the project, together with an academic student for his final work-product, using an agile approach and DevOps philosophy. This is the first experience where the basic ResDevOps idea was initially born and it fitted the expectations of the project partners.

In the second case study, some contexts and some scenarios, related to the Cyber Security complex domain, have been developed with interesting findings related to the security of Cloud services[26] and the systemic risk for banks[29]. In these case studies, the ResDevs approach has been widely appreciated. The modeling and development of some prototypes for payment system are currently ongoing with the ResDevOps approach.

However, both industrial research projects represent the situation of new IT systems whose requirements have been defined starting from the stakeholders' needs.

Here, a third case study, referring an existing EIS to be modeled and then maintained with ResDevOps, is presented. It is part of an ongoing program inside the Herzum (www.herzum.com), a private enterprise (the provider) that is supporting the IT management of a big international group (the customer) working in the design and development of systems and high technological components.

The provider has done the following activities: first, he took over the customer's EIS and he has started to model a small subset of it (i.e. the part involved by the addressed user needs) by using GOReM. On this model the provider began a continuous process of refinement and evolutionary maintenance conforming to the described ResDevs activities, supporting both a better understanding of the existing EIS addressed part, and the identification/localization of some innovations, modeled as evolution of the initial modeled EIS requirements. Then, a well-planned and integrated new program, approved by the management staff of the customer, has been started at the scheduled time. The development, done at the provider side, has been carried out with the Scrum methodology and, with the full involvement of the customer, the DevOps philosophy has been followed. Lastly, the ResDevOps process has been completely applied.

Currently, the provider is continuing its activities of modeling and maintain the customer's EIS, enlarging, step-by-step the considered subpart, looking for and listening to every customer's need and working with the domain expert of the customer. Whenever a new requirement is pertinent with an ongoing scrum project (i.e. is a user story clarification), the DevOps team considers it. If not pertinent, the requirement is taken in charge by the ResDevs team which, eventually, consider to define a new agile development project. Both ResDevs and DevOps team include stakeholders coming from both the provider's and the customer's sides.

In the following, the case study is schematically shown, with the aim to give a better insight of the ResDevOps methodology. In particular, it is shown: (i) part of the context model, i.e. the **stakeholders**, (ii) part of a scenario model related to the overall organization of the Enterprise, in particular the **roles** played by stakeholders, and (iii) an application model, in terms of **actors** and **use cases**, for managing issues inside the enterprise, called **Application for issue management**. This application model shows how the different worldwide-distributed enterprise offices used to manage whichever issue up to some time ago. Then, with the help of the provider, the model has been evolved and innovated to better fit the customer's needs. Therefore, many pieces of the EIS initial architecture were changed and some others are still evolving.

The chosen case study shows the phases of ResDevOps employed to introduce a simple innovation step, consisting in allowing the EIS worldwide-distributed staff, to process, in a controlled way, the issues related to the customer's asset requests and assignments.

Figure 5.4 shows a sketched **Stakeholder Diagram** of the**Context Model for the EIS of the customer**. The customer is a ITC system provider, so the main stakeholders (i.e. UML actors with yellow head) are researchers, technicians (e.g. ICT developers, QA testers, etc.) and operation members, internal administrative personnel, external providers, customers and some managers as R&D managers, project managers, ICT managers, program managers, portfolio managers and generic business managers.

Figure 5.5 shows the main Softgoals of the Stakeholders and their dependencies.

Figure 5.6 shows a first **Role Diagram** for a Scenario Model, modeling the needs for a big number of employees, external consultants, customers and external providers to communicate inside the EIS fitting the Enterprise organization. The roles played by the stakeholders are represented by means of UML actors with white head. We called this scenario **Overall Enterprise Organizational Communication Management**.

Inside this scenario, many application models have been defined and their evolution is maintained. For the sake of briefness, in this case study it is

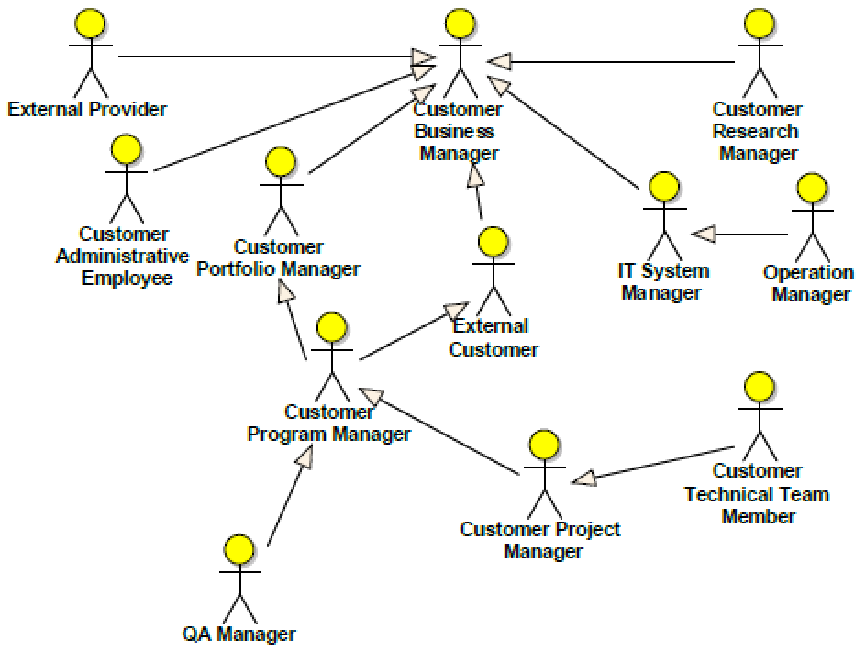

External Provider
Customer Business Manager
Customer Research Manager
Customer Administrative Employee
Customer Portfolio Manager
External Customer
IT System Manager
Operation Manager
Customer Program Manager
Customer Project Manager
Customer Technical Team Member
QA Manager

Fig. 5.4. Context Model for the EIS of the customer: Stakeholder Diagram

supposed that names given to stakeholders, roles, actors and goals are self-explanatory and easy to understand. Many GOReM diagrams, although important and actually necessary, are not reported here.

Figure5.7 shows how the enterprise answered to its needs to manage many different issues coming from worldwide actors of the enterprise. In particular, they needed an **Application for issue management**, where an **Issue submitter** actor used a **Issue Tracker** use case, having the functionality to track every actor request, to send a message to the manager able to handle its request. The manager who receives the request, arbitrarily process it and then answers the requester through the issue tracker.

The provider and the customer, by means of the ResDevs process, detected the need to innovate the issue management system.

In particular, the customer needs:

- to control the internal organization, so that requests reach the correct managers, as stated by the internal organization;
- that each request follows a predefined workflow of activities, specific for each kind of request, accordingly to a process defined by the organization.

All of this requires substituting the issue tracker use case with a new use case that, by means of a **help desk application**, is able to associate the

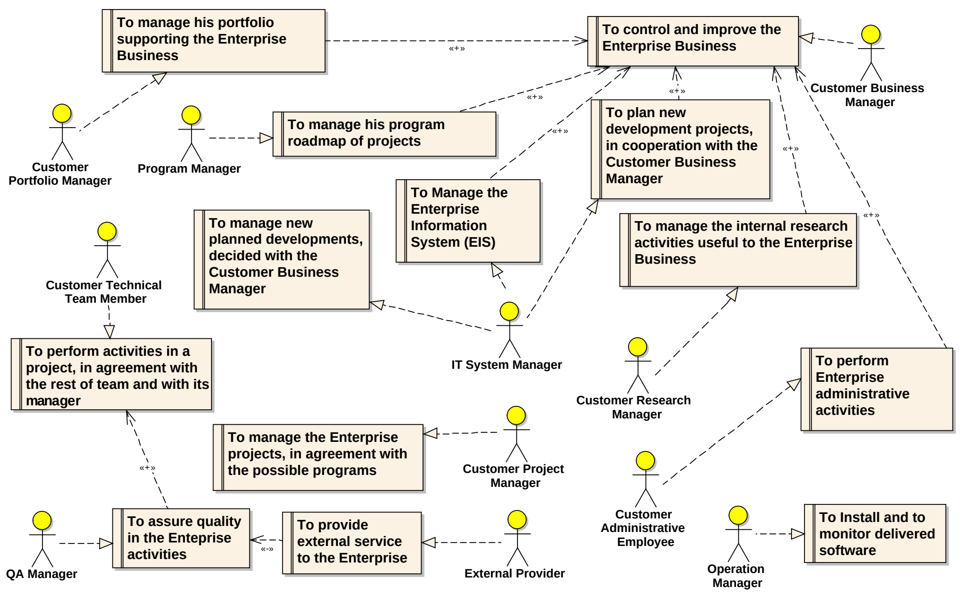

To manage his portfolio supporting the Enterprise Business
To control and improve the Enterprise Business
Customer Business Manager
Customer Portfolio Manager
Program Manager
To manage his program roadmap of projects
To plan new development projects, in cooperation with the Customer Business Manager
To Manage the Enterprise Information System (EIS)
To manage new planned developments, decided with the Customer Business Manager
Customer Technical Team Member
To manage the internal research activities useful to the Enterprise Business
IT System Manager
To perform activities in a project, in agreement with the rest of team and with its manager
Customer Research Manager
To perform Enterprise administrative activities
To manage the Enterprise projects, in agreement with the possible programs
Customer Project Manager
Customer Administrative Employee
To assure quality in the Enteprise activities
To provide external service to the Enterprise
QA Manager
External Provider
Operation Manager
To Install and to monitor delivered software
«+»
«-»

Fig. 5.5. Context Model for the EIS of the customer: Dependency Diagram

correct workflow to each kind of request to be managed by the enterprise in compliance with its organization.

The ability of managing different requests, with different workflows and activities, often without human intervention, allows to spare time for managers and to improve the internal process and the efficiency of the overall enterprise.

In particular, the customer needs the following requests handling:

- Asset Management: requests for asset, e.g. Mobile, Laptop, Personal Printer, Tablet, etc.
- Service Catalog: general purpose requests, e.g. Guest WiFi, Internet Access, VPN, Mailbox distribution list, HW booking, PC, Server Data Restore, Shared Stored Area, ...
- Support Request: whole enterprise support for some issues related to, e.g., Accounts, Applications, Email service, Facilities, Networks, Mobile, Hardware, Networks, Printers, ...
- ICT activity tracking: activity tracking support specific for the central IT division, e.g. Access Points, Applications, Backups, Computers, Data Center, LAN, Routers, Servers, Switches, UPS/Power Supply, ...

While the first three request handlers are related to the same "Overall Enterprise Organizational Communication Management" scenario, the fourth

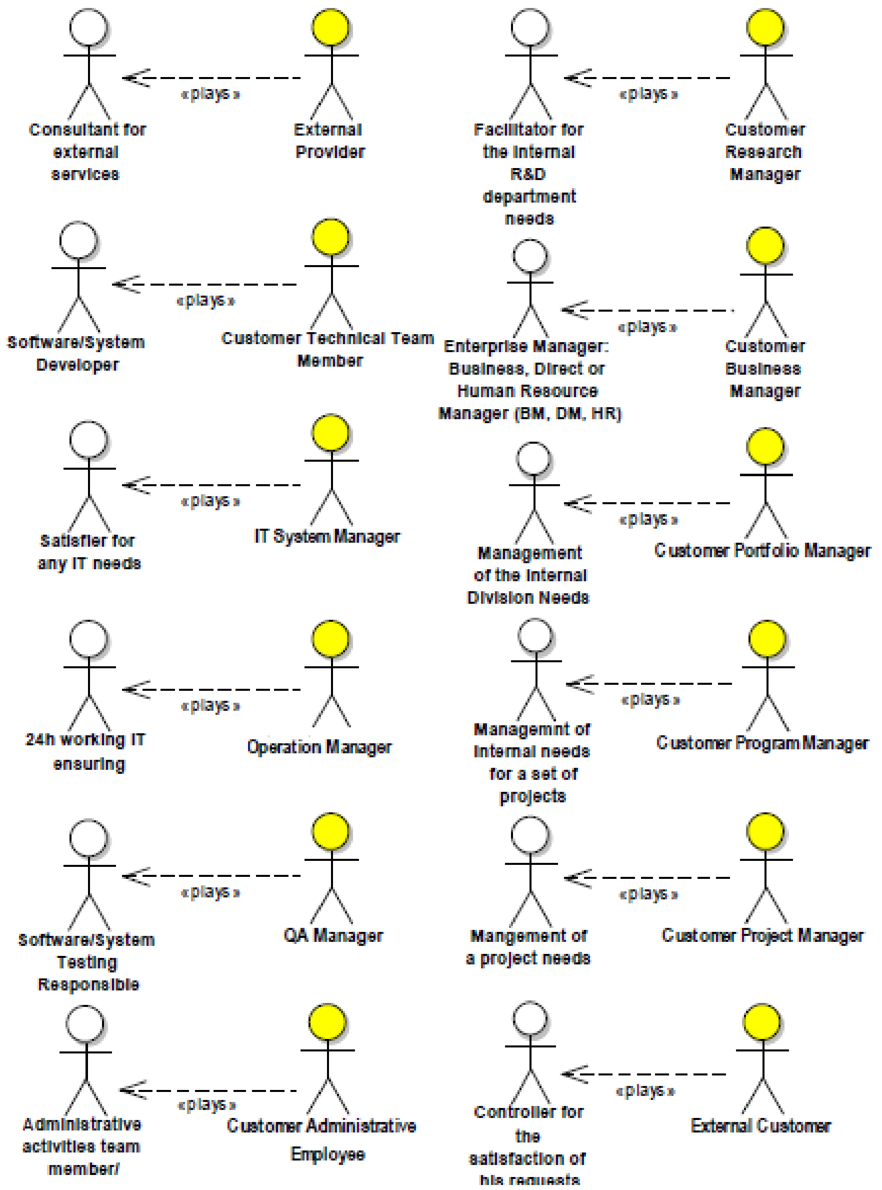

Consultant for external services
«plays»
External Provider
Facilitator for the internal R&D department needs
«plays»
Customer Research Manager
Software/System Developer
«plays»
Customer Technical Team Member
Enterprise Manager: Business, Direct or Human Resource Manager (BM, DM, HR)
«plays»
Customer Business Manager
Satisfier for any IT needs
«plays»
IT System Manager
Management of the internal Division Needs
«plays»
Customer Portfolio Manager
24h working IT ensuring
«plays»
Operation Manager
Managemnt of internal needs for a set of projects
«plays»
Customer Program Manager
Software/System Testing Responsible
«plays»
QA Manager
Mangement of a project needs
«plays»
Customer Project Manager
Administrative activities team member/
«plays»
Customer Administrative Employee
Controller for the satisfaction of his requests
«plays»
External Customer

Fig. 5.6. Scenario Model for Overall Enterprise Organizational Communication Management: Role Diagram

one belongs to a different scenario related to the request handler of the specific division related to ICT.

Figure 5.8 shows the Enterprise requirements at this point.

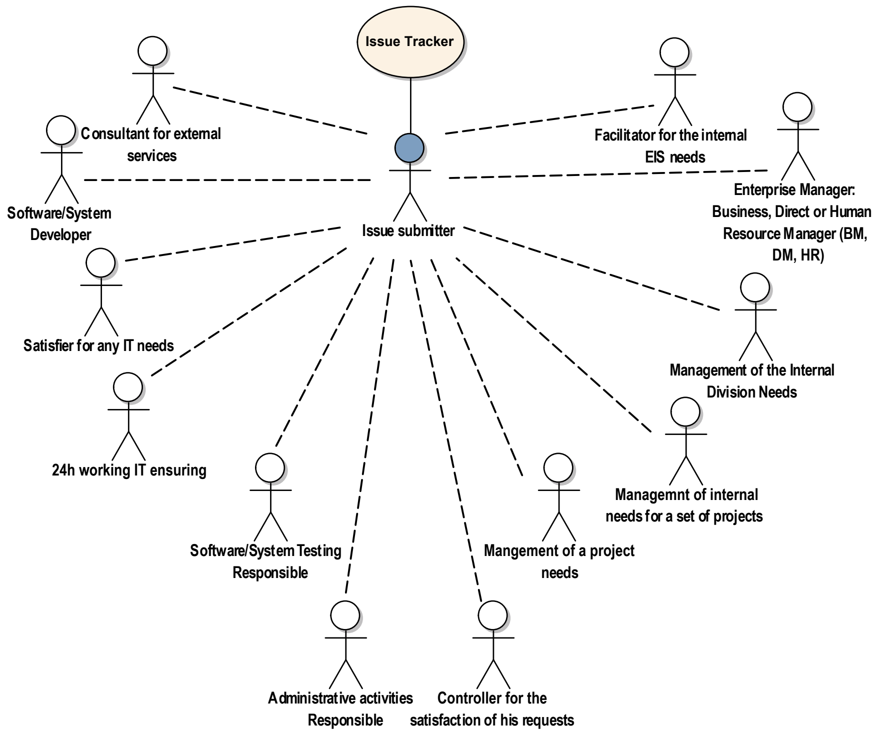

Issue Tracker
Issue submitter
Consultant for external services
Software/System Developer
Satisfier for any IT needs
24h working IT ensuring
Software/System Testing Responsible
Administrative activities Responsible
Controller for the satisfaction of his requests
Mangement of a project needs
Managemnt of internal needs for a set of projects
Management of the Internal Division Needs
Enterprise Manager: Business, Direct or Human Resource Manager (BM, DM, HR)
Facilitator for the internal EIS needs

Fig. 5.7. Application Model: Application for issue management – Role to Actor mapping and Issue Tracker Use Case

In this specific case, the provider approved a set of small development projects with the aim to incrementally introduce the specific request handling procedures. These procedures will gradually substitute the legacy system with a modern help desk, developed by using an on-the-market, powerful and mod-

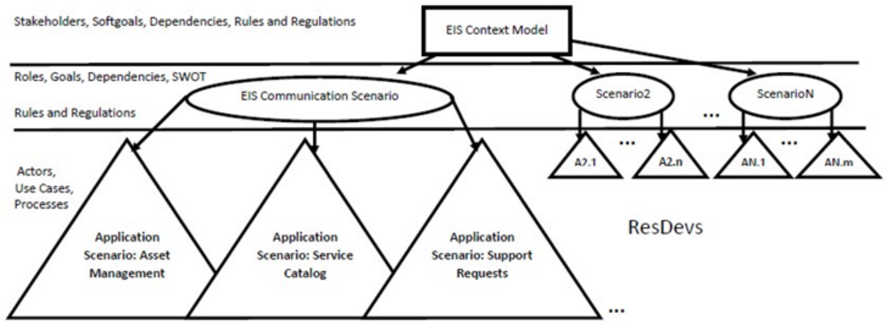

Stakeholders, Softgoals, Dependencies, Rules and Regulations
EIS Context Model
Roles, Goals, Dependencies, SWOT
Rules and Regulations
EIS Communication Scenario
Scenario2
ScenarioN
A2.1
A2.n
AN.1
AN.m
Actors, Use Cases, Processes
Application Scenario: Asset Management
Application Scenario: Service Catalog
Application Scenario: Support Requests
ResDevs

Fig. 5.8. EIS requirements model with GOReM

ern tool. In particular, the Atlassian (www.atlassian.com) tool-suite [111] has been used and many specific request management are currently on operation in the customer side.

The ResDevs process is evolving the Enterprise requirements related to the Application model **Application for issue management**, while the development team is writing stories and is developing every single project. Specifically, it is shown the new Application model related to the same Business Scenario **Overall Enterprise Organizational Communication Management**, but limited to the management of the customer issues for internal asset requests and assignments, e.g. mobile phones, SIM cards, Internet keys, personal computers, and so on.

In particular, the provider defined a new applicative scenario called **Application for asset management**. Figure5.9 shows the involved actors for each role in the new defined application model, i.e. Submitter for an asset request, DM (Direct Manager), HR (Human Resource) and Asset distributor.

Figure5.10 shows the main use cases for each actor: Submitter for an asset request may call the Help Desk use case for request an asset. DM and HR may evaluate the asset request, based on their own motivations. The actor who should distribute the assets may assign or, if the asset is not available, may reserve it and, when possible, he assigns the asset to the requestor. Figure 5.11 shows the process involved in this new application model: a request automatically reaches worldwide the proper DM who may approve the request and then asks to the HR; HR may approve and then asks to assign the asset to the requestor.

The software development factory is inside the provider who, by using a Scrum agile development process and the DevOps approach, has developed and put in operation some applications resulting from these projects.

Figure 5.12 shows the first version of the workflow defined for the "Help Desk tool" at the beginning of the project development.

Roughly speaking, this workflow specifies that each request for an asset will reach, automatically, the Direct Manager (DM) of the requestor, who can approve and automatically pass the request to the Human Resource (HR) Manager, who, in turn, can approve or not the request. If approved, the application automatically identifies who is able to assign the requested asset; he is asked to complete the request if and when possible (the request is in progress), and then to close it when resolved. In this case, the requestor will be notified that his request has been accepted and satisfied. If the request is not approved, the requestor is notified and he has the option to ask again.

This initial workflow was defined with the customer and implemented by the provider. Then, it was put in operation, in agreement with the operation team of the customer. As it naturally happens in the DevOps approach, many feedbacks came from the customer and, in accordance with the provider, small changes on the workflow of Fig.5.12 were made. The workflow currently in operation is different from the initial one, but it is not possible to show it

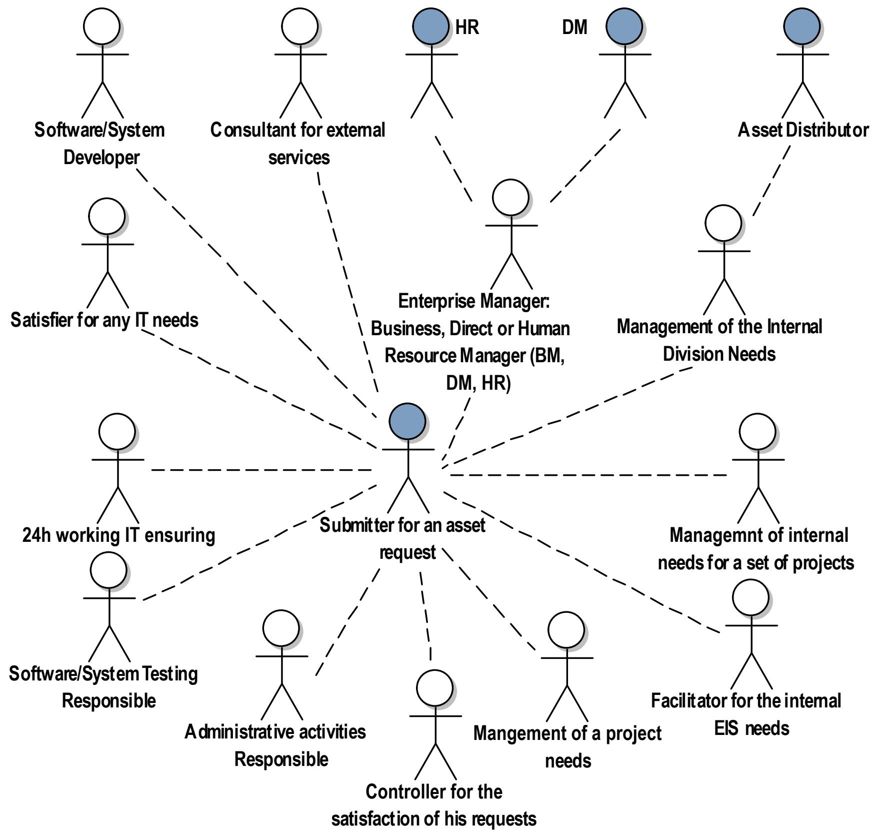

HR
DM
Software/System Developer
Consultant for external services
Asset Distributor
Satisfier for any IT needs
Enterprise Manager: Business, Direct or Human Resource Manager (BM, DM, HR)
Management of the Internal Division Needs
24h working IT ensuring
Submitter for an asset request
Managemnt of internal needs for a set of projects
Software/System Testing Responsible
Administrative activities Responsible
Controller for the satisfaction of his requests
Mangement of a project needs
Facilitator for the internal EIS needs

Fig. 5.9. New Application Model related to an Application for asset management: Actors involved by Roles

because the use of this information is restricted by a non disclosure agreement with the provider.

However, the Main Use cases of the Application model in Fig. 5.10, did not change, because the high level requirements remained the same. Every change occurred during the agile development, asked only for stories refinements.

Figure 5.13 shows the final EIS requirements model and the Service Desk Tool, resulting from the ResDevOps approach for the customer.

It is outlined the necessity to manage requirements as any others artifacts and to bind tools in operation with the configurations of requirements from which they was derived. In other word, in an EIS it should be clear for a tool the specific version of context, scenario and application, it refers to.

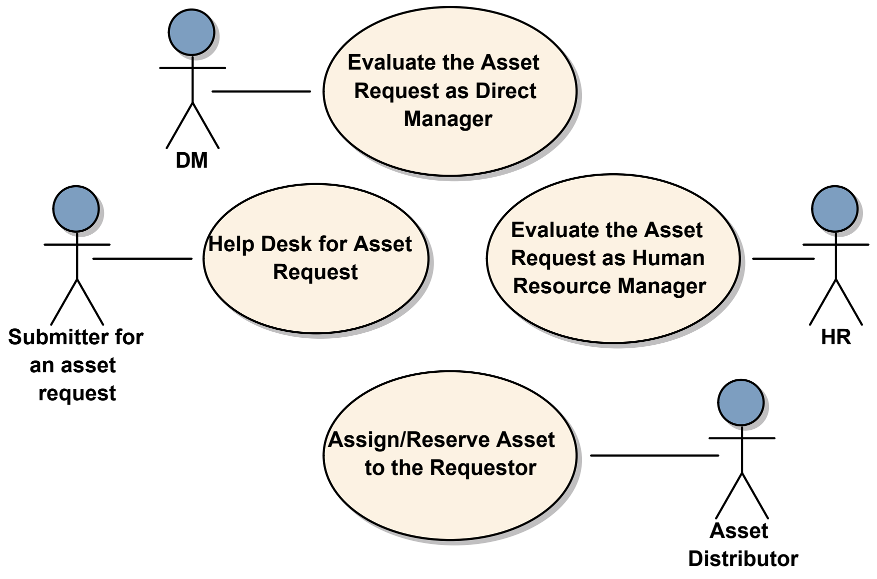

Evaluate the Asset Request as Direct Manager
DM
Help Desk for Asset Request
Submitter for an asset request
Evaluate the Asset Request as Human Resource Manager
HR
Assign/Reserve Asset to the Requestor
Asset Distributor

Fig. 5.10. New Application Model related to an Application for asset management:Main use cases

5.2.3 Conclusion and future work

ResDevOps framework suggests a software engineering working chain, as a possible solution to the problem statement of reaching a good customer satisfaction having an always up-to-date long-lived EIS.

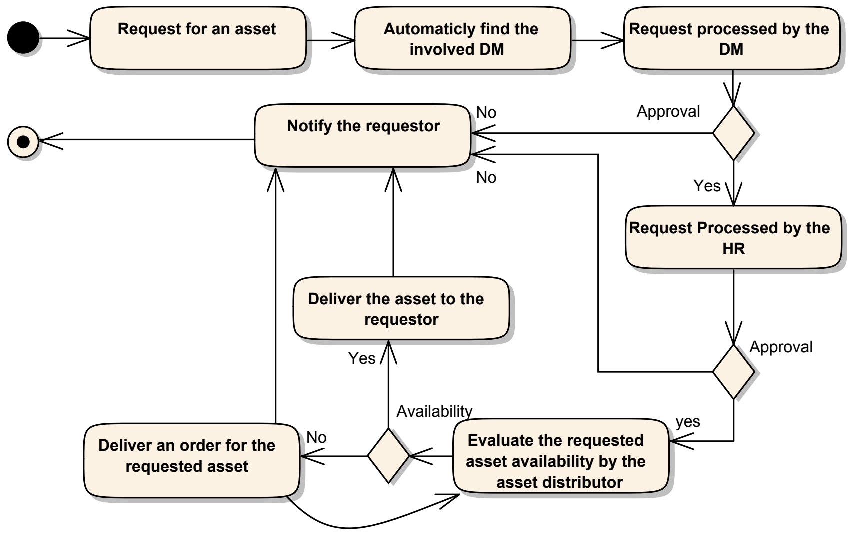

Request for an asset
Automaticly find the involved DM
Request processed by the DM
Approval
No
Notify the requestor
No
Yes
Request Processed by the HR
Approval
Deliver the asset to the requestor
Yes
Availability
yes
No
Deliver an order for the requested asset
Evaluate the requested asset availability by the asset distributor

Fig. 5.11. New Application Model related to an Application for asset management:The main process

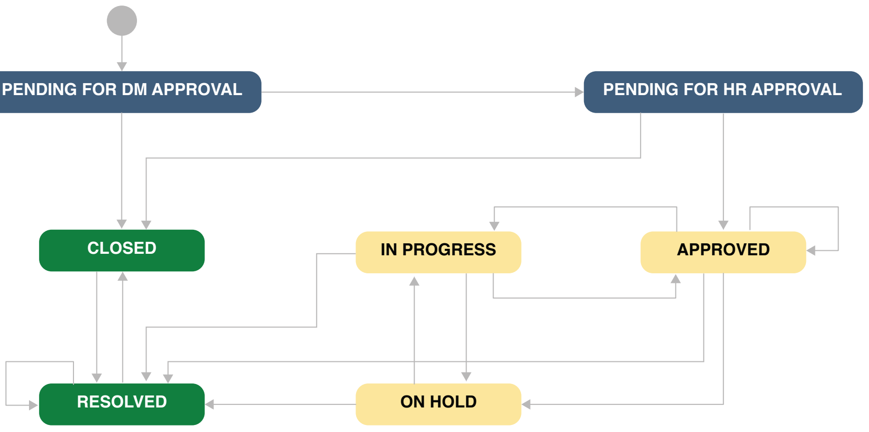


Fig. 5.12. The initial workflow implemented as part of Help Desk tool for the new application model

The overall framework is based on a ResDevs infrastructure that is managed with the GOReM methodology, whose clarity, easiness-to-learn, and expressiveness have been experimented in some large industrial research projects at University of Calabria. ResDevs concurrently cooperates with a software engineering framework based on agile methods and including a DevOps infrastructure. We have modeled the context of a possible ResDevOps framework by using GOReM itself.

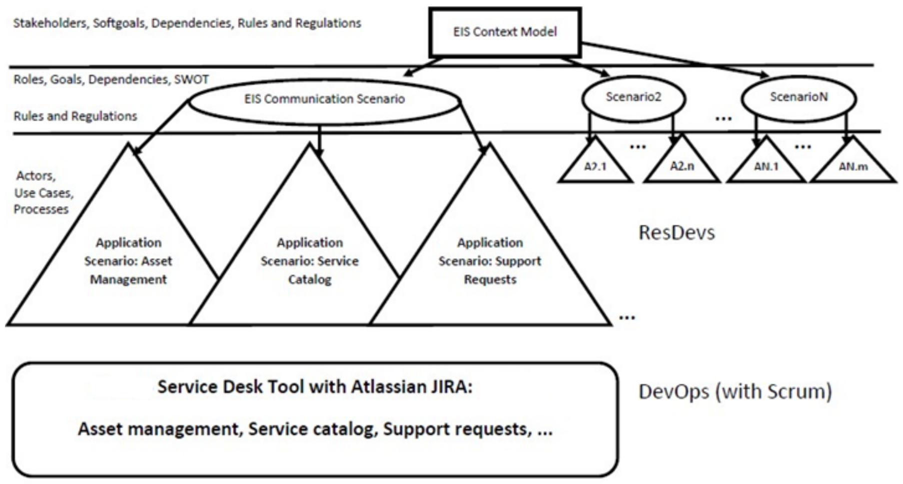

Stakeholders, Softgoals, Dependencies, Rules and Regulations
EIS Context Model
Roles, Goals, Dependencies, SWOT
Rules and Regulations
EIS Communication Scenario
Scenario2
ScenarioN
A2.1
A2.n
AN.1
AN.m
Actors, Use Cases, Processes
Application Scenario: Asset Management
Application Scenario: Service Catalog
Application Scenario: Support Requests
ResDevs
Service Desk Tool with Atlassian JIRA:
Asset management, Service catalog, Support requests, ...
DevOps (with Scrum)

Fig. 5.13. EIS requirements model and the Service Desk Tool

The main strength of applying the ResDevOps process is the ability to have a continuous control over the whole EIS requirements evolution along the time.

A potential weakness might be the extra effort of building the first GOReM model of the overall EIS, although the methodology is easy to apply and effective. In addition, a complete EIS model should be built incrementally, as shown in the case study.

A possible goal of future research could be the use of ResDevOps for extending an already existing framework, e.g. SAFe®. Also an implementation of ResDevOps as add-on of Atlassian suite should be interesting for improving the Atlassian ALM discussed in section 3.5.1 to manage an every open project ResDevs for each customer, for tracking the models' versions and for easily managing links among the ResDevs application models with the agile DevOps software releases that are in operation inside the customer environment.

Next sections are devoted to describe these kinds of extensions

5.3 ResDevs might extend some pre-existing frameworks

ResDevOps should be extended both in the direction of the management of activities, time-lines and costs of projects, programs and portfolios of a Enterprise, and for controlling what application is running in the customer operation environments with its location of hw and sw.

Framework as SAFe and DAD, previously presented, give the merited importance to these arguments and might be enriched with the ResDevs approach with the aim to benefit of the previously explained importance to have a longer life for the EIS of the customer.

The Atlassian suite supports, by means of its tools, both the agile ALM (Application Lifecycle Management) and DevOps approach. In addition, the Atlassian suite has a detailed support to whatever of its 'custom fields', that constituted its basic building blocks, where to putting precise information, such as the localization of the hw and sw inside the IT operation environment of the customer. Thus, also Atlassian suite would benefit from an addon implementing the ResDevs approach, allowing to managing the EIS requirements of the overall applications constituting the EIS and running on the IT customer operation environment.

5.3.1 Extending ResDevOps

Portfolio concept allows to manage projects development at 'Lean Agile Enterprise scale'.

Lean Enterprise scale gives a view of development teams and of the whole IT Enterprise as an event-driven system, where a product construction starts only when the order of a customer is arrived.

Agile Enterprise scale gives a view of development teams and of the whole IT Enterprise as an evolutionary delivery-driven system, where a product construction start with building and delivering the minimum possible part of the whole product, useful for having user/customer feedbacks on correct requirement gathering. Successive evolutions and increments, also with the canary deployment [113], are finalized to rapidly construct the whole product.

Lean Agile Enterprise scale gives a view of development teams and of the whole IT Enterprise as an event-and-delivery-driven system. This is scalable from small development teams and small IT Enterprises to big ones.

The adoption of this thinking needs adequate support tools.

It implies to provide executives with useful dashboards and reports, and introduces important cross-project features as Program and Portfolio concepts are.

Portfolios, Programs and Projects give to managers and executives specific frameworks for a lean agile control of the Enterprise, while, to development teams specific frameworks for lean agile software productions.

A project, such as a software project, an IT project, a business project, a product/application to be managed or the ResDevs side of ResDevOps framework seen as a project to continuously manage the overall EIS, must be supported by agile concepts, as versions and sprints, configurations or releases.

A project as ResDevs, which continuously maintain up-to-date the EIS, supports requirements models through the concepts of GOReM.

A software project, instead, has support for user stories, test cases, issue tracking, wiki spaces, and so on, because the development activity is based on some agile method together with the DevOps approach, in this case.

Project management is supported at any level, scaling from sprints plans, test plans, project plans to program plans, with support for financial information, legal contract information, workflows and permission government.

To introduce these concepts in addition to ResDevOps, it might be interesting to analyze how to introduce ResDevs inside a framework as SAFe and/or DAD which already support agile developments and DevOps in particular.

In addition, it is possible to extend with the ResDevs framework, the Atlassian suite (www.atlassian.com) that with JIRA and Confluence tools and many add-ons, is already able to support lean agile software development or, more generally ALM and DevOps. In particular, the Portfolio for JIRA Atlassian suite is able to add the previously described Portfolios concepts, well supporting project managers and executives with many useful reports, dashboards and tools, while the issue tracking of JIRA and its add-ons, allow to manage the Agile development and the DevOps approach.

In the following, both extensions are considered.

5.3.2 ResDevs inside SAFe and DAD

As already specified in a previous chapter, The Scaled Agile Framework, (SAFe) is a freely-revealed, on-line knowledge base of patterns for applying Lean and Agile development at enterprise scale. The SAFe website (www.scaledagileframework.com) allows users to browse the *SAFe Big Picture* to understand roles, teams, activities, and artifacts necessary to scale these practices to the enterprise.

SAFe realizes very well the thinking which is sketched behind the DevOps side of ResDevOps approach (i.e. the right side of figure 5.1), and also introduces very important and strategic figures and roles to manage all the previously specified Portfolio concepts for Project and Portfolio Management (PPM).

Going into the details of the architecture in figure 5.14, it is possible to suggest the introduction of the ResDevs side of the ResDevOps frameworks, to support the role of the "Lean-Agile Leaders" strictly bound to the overall EIS and to whom SAFe seems not providing sufficient practical directions.

In fact, "Lean-Agile Leaders" are the Enterprise's existing managers, leaders, and executives. Only they can change and continuously improve the EIS which support the Enterprise's business. As reported in the SAFe house of Lean, main values for Lean-Agile Leaders, starting from "respect fot people and culture", are (i) "flow", an incremental delivery value based on continuous feedbacks and adjustments, (ii) "innovation", providing time and space values for creativity, and (iii) "relentless improvement", facilitating better IT support of the business.

Concerning Disciplined Agile Delivery (DAD), which includes SAFe framework, Ambler in [12] says that: "Disciplined agile professionals will:

- Work closely with enterprise professionals, such as enterprise architects and portfolio managers.
- Adopt and follow enterprise guidance.
- Leverage enterprise assets, including existing systems and data sources.
- Enhance your organizational ecosystem via refactoring enterprise assets.
- Adopt a DevOps Culture.
- Share learnings with other teams.
- Adopt appropriate governance strategies including open and honest monitoring"

Thus, adding the ResDevOps approach might improve specifically the "Enterprise awareness", one of the key aspects of DAD framework.

5.3.3 ResDevOps with Atlassian tools

In this section the support for GOReM with the Atlassian JIRA tools is firstly presented. The EIS requirements for the third case study discussed in section 5.2, is thus shown using the GOReM approach as implemented in

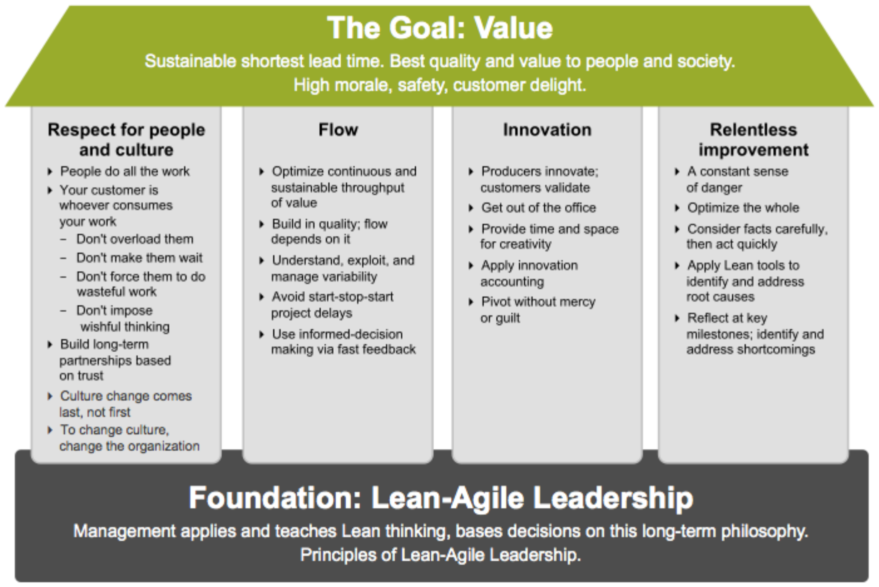

The Goal: Value
Sustainable shortest lead time. Best quality and value to people and society.
High morale, safety, customer delight.
Respect for people and culture
People do all the work
Your customer is whoever consumes your work
– Don't overload them
– Don't make them wait
– Don't force them to do wasteful work
– Don't impose wishful thinking
Build long-term partnerships based on trust
Culture change comes last, not first
To change culture, change the organization
Flow
Optimize continuous and sustainable throughput of value
Build in quality; flow depends on it
Understand, exploit, and manage variability
Avoid start-stop-start project delays
Use informed-decision making via fast feedback
Innovation
Producers innovate; customers validate
Get out of the office
Provide time and space for creativity
Apply innovation accounting
Pivot without mercy or guilt
Relentless improvement
A constant sense of danger
Optimize the whole
Consider facts carefully, then act quickly
Apply Lean tools to identify and address root causes
Reflect at key milestones; identify and address shortcomings
Foundation: Lean-Agile Leadership
Management applies and teaches Lean thinking, bases decisions on this long-term philosophy.
Principles of Lean-Agile Leadership.

Fig. 5.14. The SAFe House of Lean

JIRA. Finally, the usage of Portfolio for JIRA, a JIRA add-on, and other Atlassian tools are discussed, to show as the Atlassian platform is a good alternative to SAFe and DAD frameworks and might be enriched to support ResDevOps, because of the GOReM approach inclusion.

GOReM in Atlassian JIRA (GOReM-JIRA)

GOReM was developed in Atlassian JIRA using only basic JIRA concepts. In particular:

’Context Model’ is an issue type, and an issue of this type is associated to each defined Context Model.

A JIRA project ’GOReMStakeholders’ allows to create issues representing Stakeholders.

A JIRA project ’GOReMSoftgoals’ allows to create issues representing Softgoals.

To define a Context Model it is necessary to create a JIRA Project with the name of the Context Model and to associate (i.e. to link) to this project, the issue of issue type Context Model together with all the issues for the stakeholders and their softgoals.

’Stakeholders diagram’ in a context model is expressed by ’respond to’ and ’linked with’ links.

’Softgoals diagram’ in a context model is expressed by the predefined ’associates’ link.

'Dependency diagram' in a context model is expressed by 'hinders (-) or 'is hinder from' and 'facilitates (+)' or 'is facilitated from' and 'includes' or 'is included by' and 'facilitates/hinders' or 'is facilitate from/ is hinder from' links.

'Rules and Regulations diagram' is included, in this version, inside the attached comments of the Context Model issue.

Each 'Scenario Model' is a project with an issue of issue type Scenario Model. The issue is linked to all its issues 'roles' and 'goals', created inside the corresponding projects 'GOReMRoles' and 'GOReMGoals'. In addition, the issue of issue type Scenario Model is associated to the Context Model issue to which it belongs and vice-versa.

Each 'Role' in a Scenario Model is linked to a 'Stakeholder' of the corresponding context with the 'is played by' or 'plays' links. This constitutes the 'Role Diagram'.

'Goal Diagram' in the Scenario Model is expressed by associating 'Roles' to its 'Goals' with the predefined 'associates' link.

'Rules and Regulations' and 'SWOT analysis' are, in the current version, inside the comment of the corresponding issue Scenario model.

Each 'Application Model' is a project with an issue of issue type Application Model. The issue is linked to all issues 'actors' and 'use cases', created inside the corresponding projects 'GOReMActors' and 'GOReMUseCases'. In addition the issue of issue type Application Model is associated to the corresponding Scenario Model issue and vice versa.

Each 'Actor' in an Application Model is linked to its 'Role' of the corresponding Scenario Model with the predefined 'associates' link.

Each 'Actor' is associated also to their 'use cases' with the predefined 'associates' link.

The processes are, in the current version, inside the comment of the corresponding Application Model issue.

Case study implementation: EIS of a customer in JIRA

While the Context Model in figure 5.5 was implemented as-is in GOReM-JIRA, the Scenario Model related to the EIS Communication was implemented in the simplified (with respect to Figure 5.6), version shown in Figure 5.15.

The Scenario Model for the 'EIS Communication Management', sees the 'Manager', that is a 'Business Manager', as the 'Requester', i.e. whichever of the involved Stakeholders that plays this role, and the 'Manager of the Internal Division Needs', that is an 'Administrative Employee' Stakeholder, managing only a warehouse of assets for the corresponding division.

The Application Model for the 'Asset Management', sees as actor a 'Direct Manager (DM)' or a 'Human Resource Manager (HR)', both in charge to receive requests for assets from a 'Requester' and to evaluate each request

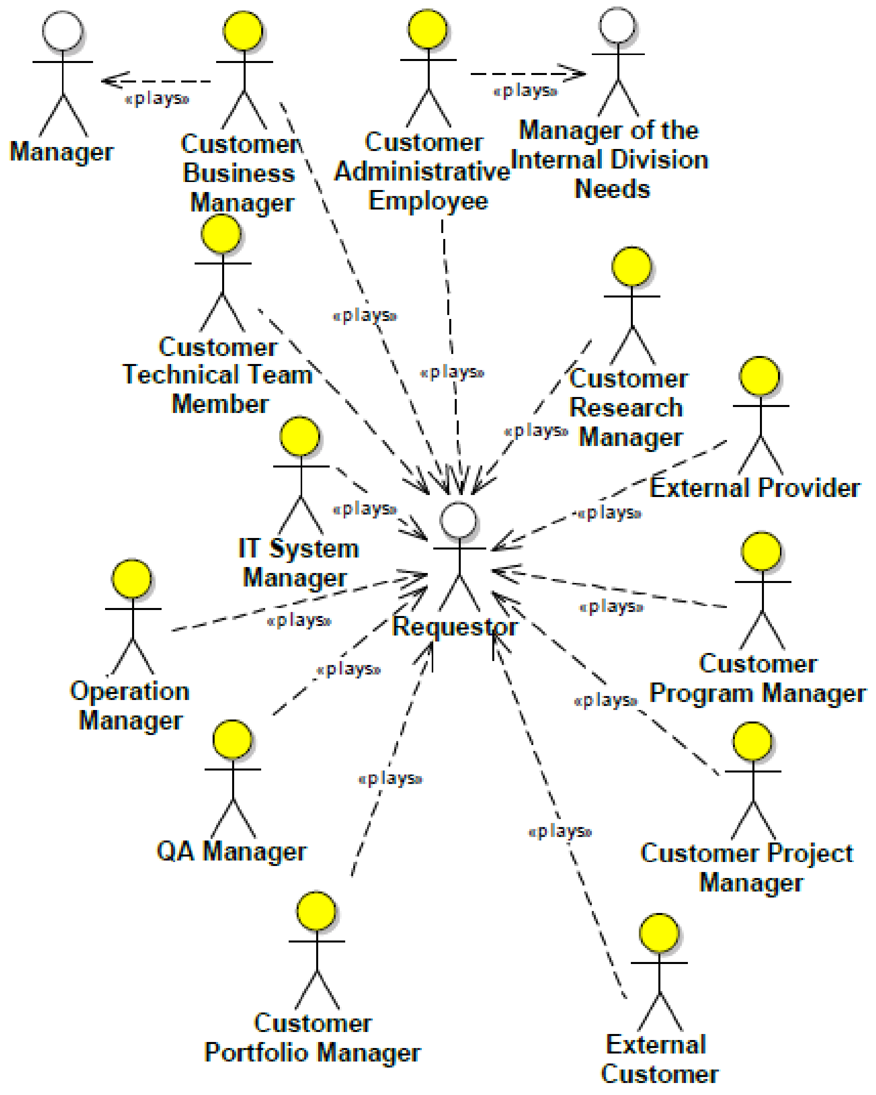

Manager
Customer Business Manager
Customer Administrative Employee
Manager of the Internal Division Needs
Customer Technical Team Member
Customer Research Manager
External Provider
IT System Manager
Requestor
Operation Manager
Customer Program Manager
QA Manager
Customer Project Manager
Customer Portfolio Manager
External Customer
«plays»

Fig. 5.15. Scenario Model for EIS Communication Management: Roles Diagram

based on their own function. The 'Requester' is in charge to submit the request to an 'Help Desk tool' and the actor 'Asset Distributor', actor for the Manager of the 'Internal Division Needs' who 'Assign or Reserve' assets to the 'Requester', if authorized to do so.

ResDevOps in JIRA

In the previous sections, the JIRA implementation of GOReM has been shown and a case study has been described.

That allows to implement the ResDevs side of the ResDevOps approach.

The DevOps approach is naturally supported with the Atlassian tools, allowing to support every agile development methods, by using JIRA Software for supporting the development projects and many DevOps Atlassian offered tools (see https://www.atlassian.com/devops/tools-and-technologies) supporting the DevOps philosophy, described in the section 3.2.

The integration between ResDevs and DevOps (as in 5.1) for having ResDevOps, remains to be specified.

To this aim, referring to Figure 5.13, in JIRA it is sufficient to add to each software project or application project realizing an Application model, an issue of the GOReM defined issue type Application Model, and to link it to the issue of the same type, which identifies the GOReM Application Model referred by the JIRA project and defining the requirements.

It is worth to note that the user stories inside the JIRA project realizing the Application Model, might be easily derived from the corresponding use cases. In this way, every use case in the Application Model, that is a high level use case telling 'what' and not 'how', could be translated into a set of user stories and implemented in the software JIRA project.

Every change to the user stories or to the JIRA workflow implementing processes, should be compared with the Application Model with the aim to understand if (i) it is necessary to provide a new version of the application model traced and added to the same Scenario Model, or (ii) if the Scenario Model is affected, so it is necessary to provide also a new (version) Scenario Model for the same Context Model or, (iii) if also the Context Model is affected and a new version (possibly improved) has to be provided.

As far as version and configuration management, it is necessary to deal with a configuration and version control tool. It could be the same used for the software projects implementing the application models. In fact, the Requirements Model for the EIS, as said in other parts of this thesis, is an artifact as any other software artifact [16].

In the Atlassian world, tools as GIT or Atlassian Bitbucket and Atlassian Stash, and/or other tools integrated in the Atlassian suite, may be configured for our aims.

Concerning the case study discussed in the previous section, the implementation in JIRA has been done by creating Projects and issues following the JIRA approach. In particular, the context model for the 'EIS of the customer', the 'EIS Communication' scenario model and the 'Asset management' application model with their realization is shown in the following. Finally a JIRA software project, named 'Asset Management' with the issue 'Asset Request' and its JIRA workflow is shown and linked to the requirements expressed in the corresponding Application Model.

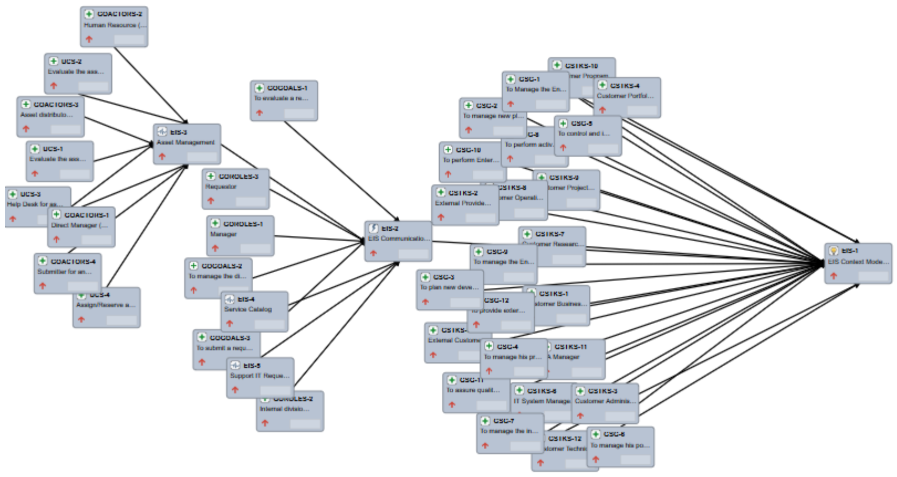
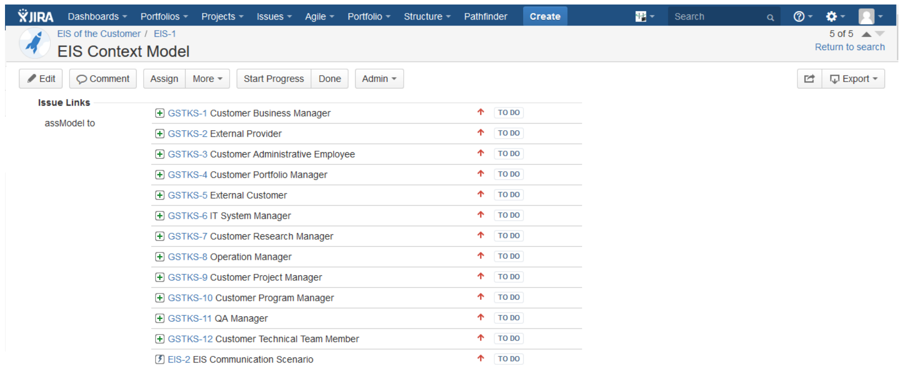


Fig. 5.16. GOReM in JIRA: Projects and issues, modeling the context, the scenario and some applications



EIS of the Customer / EIS-1

EIS Context Model

Issue Links

assModel to

GSTKS-1 Customer Business Manager	↑	TO DO
GSTKS-2 External Provider	↑	TO DO
GSTKS-3 Customer Administrative Employee	↑	TO DO
GSTKS-4 Customer Portfolio Manager	↑	TO DO
GSTKS-5 External Customer	↑	TO DO
GSTKS-6 IT System Manager	↑	TO DO
GSTKS-7 Customer Research Manager	↑	TO DO
GSTKS-8 Operation Manager	↑	TO DO
GSTKS-9 Customer Project Manager	↑	TO DO
GSTKS-10 Customer Program Manager	↑	TO DO
GSTKS-11 QA Manager	↑	TO DO
GSTKS-12 Customer Technical Team Member	↑	TO DO
EIS-2 EIS Communication Scenario	↑	TO DO

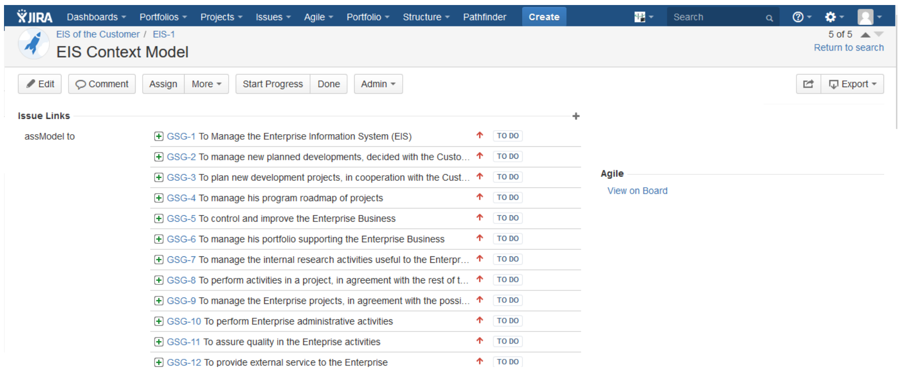
Fig. 5.17. GOReM in JIRA: Context Model: EIS for the customer and its Stakeholders issues

Figure 5.16 shows an overview of the models hierarchy.

Figure 5.17, shows the Context Model linked issues, representing the Stakeholders, while figure 5.18 shows the Context Model linked issues, representing Softgoals.

Figure 5.19 shows how the Stakeholders Dependency Diagram, of figure 5.4 appears in JIRA.

Figure 5.20 shows, as an example, the stakeholder 'Operation Manager' and its linked issues, that is: its Context Model 'EIS Context Model', an issue of the GOReM defined type 'Context Model', named 'assModel', which identify the model. In this case the Context model of the 'EIS for the customer' project (i.e. the overall EIS GOReM requirements) where the stakeholder is



EIS of the Customer / EIS-1

EIS Context Model

5 of 5

Return to search

Issue Links

assModel to	GSG-1 To Manage the Enterprise Information System (EIS)	↑	TO DO
	GSG-2 To manage new planned developments, decided with the Custo...	↑	TO DO
	GSG-3 To plan new development projects, in cooperation with the Cust...	↑	TO DO
	GSG-4 To manage his program roadmap of projects	↑	TO DO
	GSG-5 To control and improve the Enterprise Business	↑	TO DO
	GSG-6 To manage his portfolio supporting the Enterprise Business	↑	TO DO
	GSG-7 To manage the internal research activities useful to the Enterpr...	↑	TO DO
	GSG-8 To perform activities in a project, in agreement with the rest of t...	↑	TO DO
	GSG-9 To manage the Enterprise projects, in agreement with the possi...	↑	TO DO
	GSG-10 To perform Enterprise administrative activities	↑	TO DO
	GSG-11 To assure quality in the Enteprise activities	↑	TO DO
	GSG-12 To provide external service to the Enterprise	↑	TO DO

Agile

View on Board

Fig. 5.18. GOReM in JIRA: Context Model: EIS for the customer and its Sofgoals issues

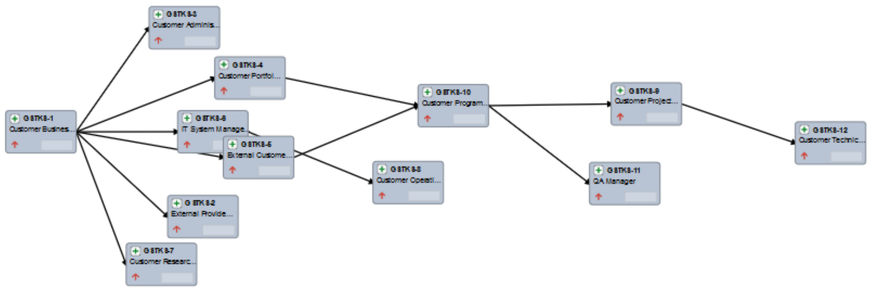
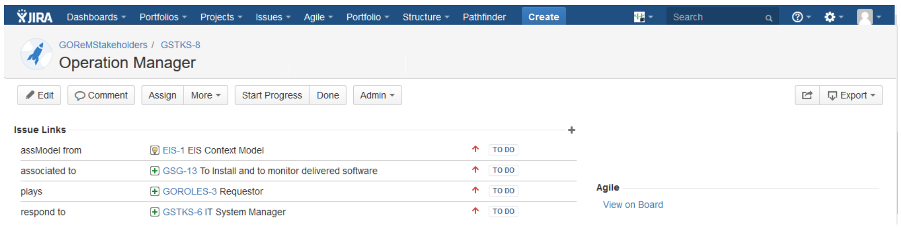


Fig. 5.19. GOReM in JIRA: Context Model: Stakeholder Dependency Diagram



GOReMStakeholders / GSTKS-8

Operation Manager

Issue Links

assModel from	EIS-1 EIS Context Model	↑	TO DO
associated to	GSG-13 To Install and to monitor delivered software	↑	TO DO
plays	GOROLES-3 Requestor	↑	TO DO
respond to	GSTKS-6 IT System Manager	↑	TO DO

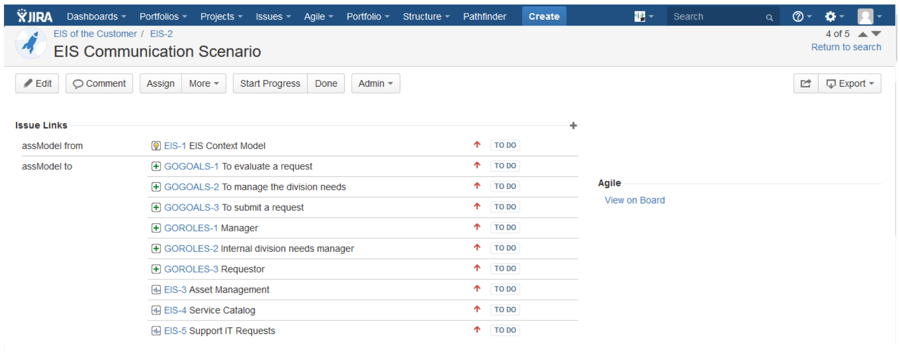
Agile

View on Board

Fig. 5.20. GOReM in JIRA: Context Model: Stakeholder and its linked issues

considered; the softgoal 'To Install and to monitor delivered software' that is a softgoal of this stakeholder, as shown in Figure 5.5. The 'plays' link associates this stakeholder to the role 'Requestor' played in the application model 'Asset Management' where the role belongs to and the link 'respond to' reaching the stakeholder 'IT System Manager', constituting its upper in the hierarchy of Figure 5.4.

Figure 5.21 shows the Scenario Model for the 'EIS Communication of the customer' with all its links i.e. roles and related goals. Rules and Regulations



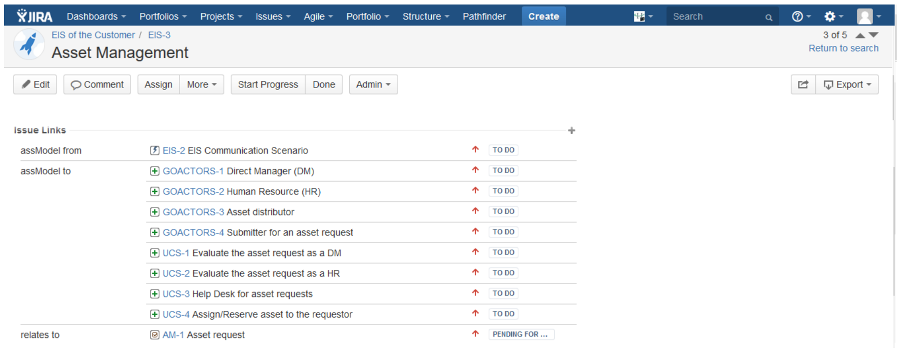
EIS of the Customer / EIS-2

EIS Communication Scenario

Issue Links

assModel from	EIS-1 EIS Context Model	TO DO
assModel to	GOGOALS-1 To evaluate a request	TO DO
	GOGOALS-2 To manage the division needs	TO DO
	GOGOALS-3 To submit a request	TO DO
	GOROLES-1 Manager	TO DO
	GOROLES-2 Internal division needs manager	TO DO
	GOROLES-3 Requestor	TO DO
	EIS-3 Asset Management	TO DO
	EIS-4 Service Catalog	TO DO
	EIS-5 Support IT Requests	TO DO

Fig. 5.21. GOReM in JIRA: Scenario Model: EIS Communication and its linked issues



EIS of the Customer / EIS-3

Asset Management

Issue Links

assModel from	EIS-2 EIS Communication Scenario	TO DO
assModel to	GOACTORS-1 Direct Manager (DM)	TO DO
	GOACTORS-2 Human Resource (HR)	TO DO
	GOACTORS-3 Asset distributor	TO DO
	GOACTORS-4 Submitter for an asset request	TO DO
	UCS-1 Evaluate the asset request as a DM	TO DO
	UCS-2 Evaluate the asset request as a HR	TO DO
	UCS-3 Help Desk for asset requests	TO DO
	UCS-4 Assign/Reserve asset to the requestor	TO DO
relates to	AM-1 Asset request	PENDING FOR ...

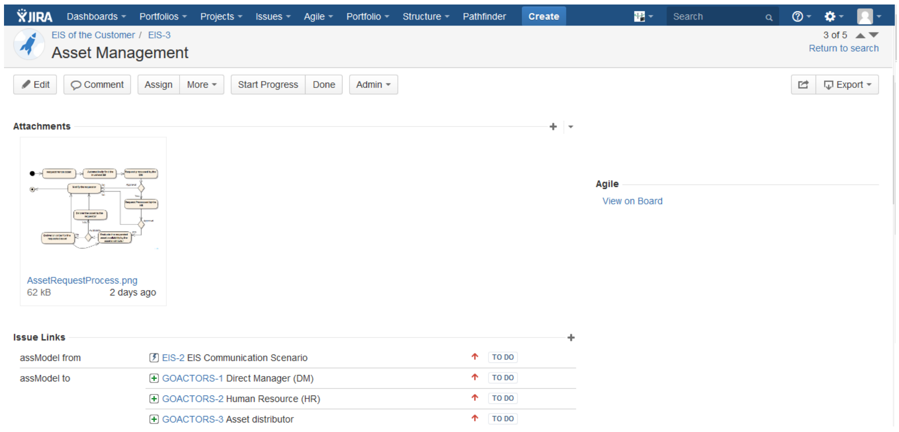
Fig. 5.22. GOReM in JIRA: Application Model: Asset Management and its linked issues

and SWOT in this version, are added as comments to the issue representing the scenario model. In addition the figure shows links to the Application Models 'Asset Management', 'Service Catalog' and 'Support IT Request'.

Concerning the 'Asset Management', chosen as an example, Figure 5.22 shows Actors and Use Cases, conforming to GOReM. The main Process is inserted in a comment of the issue representing the Application Model, as shown in Figure 5.23

This application model is linked to a software project in JIRA, called 'Asset Management', as shown by the link named 'related to'. This is the development project that implements this application model for 'Asset Management'. Figure 5.24 shows the project with only one issue, the Asset Request linked to (by 'related to') the application model, as shown in Figure 5.25.

Finally figure 5.26 shows the JIRA Workflow implementing the process described in the Application Model.


JIRA Dashboards Portfolios Projects Issues Agile Portfolio Structure Pathfinder Create Search

EIS of the Customer / EIS-3

Asset Management

3 of 5
Return to search

Edit | Comment | Assign | More | Start Progress | Done | Admin | Export

Attachments

AssetRequestProcess.png
62 kB 2 days ago

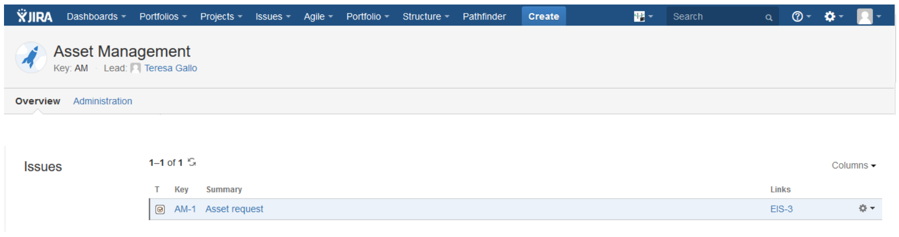
Agile

View on Board

Issue Links

assModel from	EIS-2 EIS Communication Scenario	↑	TO DO
assModel to	GOACTORS-1 Direct Manager (DM)	↑	TO DO
	GOACTORS-2 Human Resource (HR)	↑	TO DO
	GOACTORS-3 Asset distributor	↑	TO DO

Fig. 5.23. GOReM in JIRA: Application Model: Asset Management Process


JIRA Dashboards Portfolios Projects Issues Agile Portfolio Structure Pathfinder Create Search

Asset Management
Key: AM Lead: Teresa Gallo

Overview Administration

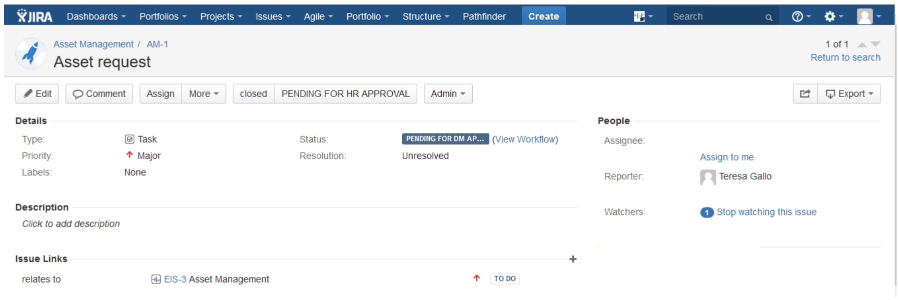
Issues

1–1 of 1

Columns

T	Key	Summary	Links
	AM-1	Asset request	EIS-3

Fig. 5.24. Asset Management JIRA SW Projects


JIRA Dashboards Portfolios Projects Issues Agile Portfolio Structure Pathfinder Create Search

Asset Management / AM-1

Asset request

1 of 1
Return to search

Edit | Comment | Assign | More | closed | PENDING FOR HR APPROVAL | Admin | Export

Details

Type:	Task	Status:	PENDING FOR DM AP... (View Workflow)
Priority:	Major	Resolution:	Unresolved
Labels:	None		

Description

Click to add description

Issue Links

relates to	EIS-3 Asset Management	↑	TO DO

People

Assignee:	
	Assign to me
Reporter:	Teresa Gallo
Watchers:	1 Stop watching this issue

Fig. 5.25. Asset Request related to the application model 'Asset Management'

For each asset request, the requestor creates and starts a new issue and Asset Management application project. The workflow allows this request to reach the Direct Manger of the requestor, and the process goes on.

Then, by maintaining up-to-date both the 'EIS for the customer' JIRA project, realizing ResDevs, as specified with GOReM, and the links among Application Models and software/application JIRA projects, implemented with

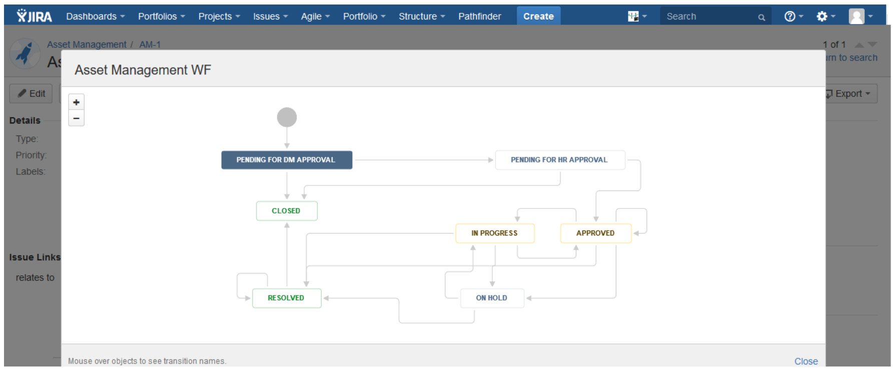


Fig. 5.26. Asset Request JIRA workflow

Scrum and DevOps approachs, as supported by the Atlassian JIRA tool set, the ResDevOps approach is completed.

ResDevOps in JIRA and the IT Management Business support

The previous section has shown how ResDevOps has been implemented using the Atlassian Tools framework. It is worth to note that the ResDevs side should be related not only to functional and non functional requirements but also to the providers, to the IT infrastructure were the applications are located and run (i.e. hw + S.O. and basic sw environments). So by saying IT Management, actually the whole IT department, with its systems (both hardware and software), is considered.

Then, as specified in the section 3.5, it is important to support the overall EIS requiremnts engineering in an Enterprise, that is the whole IT department, considered as a business supporting the enterprise business.

To this end, as said in 5.3.1, SAFe and DAD offer support for projects, programs and portfolios management in a completely agile development environment including DevOps, while Atlassian offers JIRA portfolios functionalities, allowing to support JIRA projects from the point of view of the responsibilities, budgets, elapsed times, resources, and so on.

Concluding, with the tools Atlassian JIRA and 'Portfolio for JIRA' (www.atlassian.com/software/jira/portfolio), ResDevOps applied to the EIS of the Enterprises may be managed as an always opened JIRA project. In fact, it is possible to allocate and control to the EIS requiremnts project, a number of resources and a (annual) budget, as well as to manage every new projects produced by the left side of Figure 5.1. In addition by linking the software projects issues to the use cases of the application models, it is possible to integrate the right side of the Figure 5.1.

Concluding, ResDevOps would enhance the Atlassian tool set with a comprehensive IT Management, by maintaining alive the overall EIS requirements, including control over costs, resources (i.e. Human resources, hardware and software), innovations, new businesses, new needs and planning. In addition, with Atlassian JIRA, it is easy to link application models to the corresponding implementation projects or application JIRA projects and, in a future, to link them with the operational environments (both hw and sw) where the final applications are running.

6

Conclusions

6.1 Main contributions

The research presented in this thesis aimed at contributing to fill the gap between:

- traditional methods for software development, which had been going in the direction of being very formal, with very long schedule and with a big distance between the understanding of the customer needs, usually collected in large documents of requirements, and the delivery of some software solution that (at least partially) satisfied the customer, and
- the current trend of the lean agile methods that, through looking for the customer satisfaction with the continuous delivery concept, not only needs a revolution on the kind of contract between provider and the customer, who with big difficulty appreciates the lack of a preventive definite cost for obtaining an application satisfying its needs, but also refuses to document and to maintain the requirements of what is developed.

This gap results in the issue that the customer could loose the control over the features of the single software in developing state, although with some good functionalities already tested, but he has no idea of the final state in term of what and when it will be reached. In addition, the customer could loose the control of the satisfied needs and of the location and configuration of its vital complex, often distributed, EIS. This because of the presence of many software providers working with lean and agile methods on the same EIS, a continuously changing set of integrated, often distributed applications or, generally speaking, an IT system.

The problem statement consists in the lack of a sufficient control over the overall EIS, in term of how it is composed, what are the boundaries of each subpart, what is doing each subpart, where it is running, for who and why.

he in-depth idea inside in this work, attributes the unmanageable complexity of EISs to the absence of sufficient care on requirements management activity.

Also considering the Scrum method, there are schools for becoming a Scrum Master, but less schools, best practices or methods to follow for the figure of the Scrum Product Owner, who should understand what the customer wants, and should write down the needs as user stories for the development team.

Requirements engineering, indeed seems to go on, in the tentative to include the concept of 'user story' used by the lean agile method for defining what the customer is looking for.

Requirements in agile are documented by user stories and by reading the developed code, but this is not sufficient for achieving both customer and provider satisfaction, whose work is related to the resulting complexity of the EIS.

In sum: there is a middle ground between one extreme, absurdly bureaucratic of the traditional software engineering methods, and the other, absurdly informal of the agile approach.

In agreement with [16], requirements should be treated as an asset of the Enterprise, exactly like other software components. Requirements continuously change and should be put under the control of configuration management tools. Relying exclusively on user stories and on code, as the only sources of requirements, it is not sufficient to maintain, over the time, a solid IT system. This is one of the main limitations of agile methods.

In this context, two main contributions have been provided:

- GOReM, a goal-oriented methodology for requirements engineering, with specific phases, allowing effective collaboration among stakeholders, easy to use because based on UML, able to give the merited importance to rules and regulations to be taken into account for non functional requirements, well separating the mastering concept exploration of a requirement from the design of a solution. This methodology allows to facilitate cooperation among great amount of stakeholders who must collaborate for the definition and realization of an IT system.
- ResDevOps approach that supports a long-lasting EIS, throughout a continuous requirements evolution and innovation support, possibly guided by GOReM, and controlled lean agile developments, in a continuous unceasing chain.

The experience gained during the definition of the above mentioned results, allowed to face more general issues concerning many different contexts, as Tourism, Cyber security, Simulation and Cloud Services, allowing to experiment the proposed concepts within real case studies.

The case studies and the definition of the previously addressed works have been published in important and specific domain conferences. In addition, a manuscript is in press on a international journal.

6.2 Future Work

The results presented in this thesis constitute a starting point for ongoing and future research activities.

As explained in previous chapter, ResDevOps might be better implemented as a plug-in of Atlassian tool set, allowing the ALM already supported tools to be more generally applied to an EIS and not only to every single application.

Better, but more ambitious, would be to work for adding ResDevOps in the DAD framework and/or for being considered in the SAFe framework, currently a focus of the Software Engineering Community.

Conclusion of this thesis is: currently, the main direction of Software Engineering, with the agile methods and SAFe framework, is towards *building an application right* for the current needs of the customer. However, Requirements Engineering, part of the Software Engineering, is fundamental for *building the right application*, usually running in a complex and distributed IT system, like whatever nowadays Enterprise Information System. This problem statement requires some additional work and visibility at the international level to persuade the Software Engineering Community to support this direction.

References

1. Hebert D. Benington. "Production of Large Computer Programs", paper adapted from a presentation at a symposium on advanced programming methods for digital computers sponsored by the Navy Mathematical Computing Advisory Panel and the Office of Naval Research in June 1956, 1956
2. Winston W. Royce. "Managing the Development of Large Software Systems", Proceedings, IEEE WESCON, August 1970, pages 1-9, 1970
3. B. W. Boehm. "Software Engineering", IEEE Trans Computers, Dec. 1976, pp. 1226-1241
4. P. Herzum, O. Sims. "Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise", Wiley, 2000
5. I.Sommerville. "Software Engineering". 10th edition, 2015.
6. B.Boehm. "A spiral model of software development and enhancement", Newsletter ACM SIGSOFT Software Engineering Notes, Volume 11 Issue 4, pp 14-24, 1986
7. Rational Software White Paper. "Rational Unified Process - Best Practices for Software Development Teams". TP026B, Rev 11/01, 2001.
8. Charles Edeki. "Agile Unified Process". International Journal of Computer Science and Mobile Applications, Vol.1 Issue. 3, September- 2013, pg. 13-17 ISSN: 2321-8363.
9. S.Ambler. "Agile modeling: Effective practices for extreme programming and the unified process". New York: John Wiley and Sons, Inc., 2002
10. K.Beck et al. "Manifesto for agile software development", 2001, Online: http://agilemanifesto.org/
11. S.Ambler. "The agile unified process", 2005, Online: http://www.ambysoft.com/unifiedprocess/agileUP.html
12. Scott W. Ambler, M. Lines. "Disciplined Agile Delivery (DAD): A Practitioner's Guide to Agile Software Delivery in the Enterprise", IBM Press, ISBN: 0132810131, 2012
13. A.Shalloway. "Lean Agile: An Overview of Agile Methods - White Papers essential", Sep 2010, www.netobjectives.com/files/resources/articles/OverviewOfLeanAgileMethods.pdf
14. M. W. Alford and J.T.Lawson. "Software Requirements Engineering Methodology (Development)", TRW Defense and Space Systems Group 7702 Governors Drive West Huntsville AL 35805, 1979.

15. Stuart R. Faulk. ”Software Requirements: A Tutorial”, Software Requirements Engineering (2nd ed.), IEEE Computer Society Press. ISBN 0-8186-7738-4., 1997 E. Clifton. ”Fault tree analysis - a history”. Proceedings of the 17th International Systems Safety Conference, pp. 1-9, Orlando (Florida, USA), August 16-21, 1999.
16. B.Meyer. ”Agile. The Good, the Hype and the Ugly”, Springler, ISBN-10: 3319051547, 2014
17. Scaled Agile Inc.”SAFe® 4.0 Introduction, Overview of the Scaled Agile Framework® for Lean Software and Systems Engineering”, White Paper, July 2016
18. Deloitte Consulting LLP. ”2016 Global Outsourcing Survey - Outsourcing accelerates forward”, June 2016 http://www2.deloitte.com/content/dam/Deloitte/us/Documents/process-and-operations/us-cons-sdt-gos-exec-summary-2016.pdf
19. A. Botchkarev, P. Finnigan. ”Complexity in the context of information systems project management” Organisational Project Management 2015, 2(1): 15-34, http://dx.doi.org/10.5130/opm.v2i1.4272
20. PMI. ”The View From Above:THE POWER OF PORTFOLIO MANAGEMENT”, white paper of the Project Management Institute , April 2013, http://www.pmi.org/-/media/pmi/documents/public/pdf/white-papers/portofolio-management.pdf
21. I.Sommerville. ”Software Engineering”.Ninth Edition Addison-Wesley, ISBN-10: 0-13-703515-2, 2011
22. T. Dingsøyr, S. Nerur, V. Balijepally, N. Brede Moe. ”A decade of agile methodologies: Towards explaining agile software development”, Journal of Systems and Software, Elsevier, Volume 85, Issue 6, June 2012, Pages 1213-1221
23. S. Citrigno, A. Furfaro, T. Gallo, A. Garro, S. Graziano, and D. Saccà. ”Mastering concept exploration in large industrial research projects” in INCOSE Italian Conf. on Systems Eng. (CIISE 2014), Rome, Italy, November 24-25 2014, pp. 26-37
24. A. van Lamsweerde. ”Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]”, in Proc. of 12th IEEE Int. Requirements Eng. Conf., Sep. 2004, pp. 4-7.
25. DCS.”District of cyber security,”, 2016, Online: https://www.distrettocybersecurity.it
26. A. Furfaro, T. Gallo, A. Garro, D. Saccà, and A. Tundis. ”Requirements specification of a cloud service for cyber security compliance analysis”, in Proc. of 2nd IEEE International Conference on Cloud Computing Technologies and Applications (CloudTech 2016), 2016.
27. A.Furfaro, T.Gallo, A.Garro, D.Saccà, A.Tundis. ”ResDevOps: A software engineering framework for achieving long-lasting complex systems”, Proc. of 24th IEEE International Requirements Engineering Conference, Bijing, China,, IEEE Computer Society Conference Publishing Services (CPS), 2016; 246-255, doi:10.1109/RE.2016.15.
28. DICET-INMOTO. ”ORganization of Cultural HEritage for Smart Tourism and Real-time Accessibility (OR.C.HE.S.T.R.A.)”, Project funded by the Italian Ministry of Education, University and Research (MIUR) - PON Project - Research and Competitiveness 2007-2013, 2015

29. A.Furfaro, T.Gallo, D.Saccà. "Modeling cyber systemic risk for the business continuity plan of a bank. in Proc. of the Int. Cross Domain Conference and Workshop (CD-ARES 2016), Salzburg, in Computer Science 9817 LN (ed.), Springer International Publishing Switzerland 2016, 2016; 158 - 174, doi:10.1007/978-3-319-45507-511.
30. S.Citrigno, A.Furfaro, T.Gallo, A.Garro, S.Graziano, D. Saccà, A.Tundis. "Extending GOReM Through the RAMSoS Method for Supporting Modeling and Virtual Evaluation of the Systemic Risk" Proceedings of the 2nd INCOSE Italia Conference on Systems Engineering, Turin, Italy, November 14-16, 2016, pp 16-25
31. D.CHAPPELL."WHAT IS APPLICATION LIFECYCLE MANAGEMENT?", COPYRIGHT © 2008 CHAPPELL & ASSOCIATES, December 2008, http://www.davidchappell.com/WhatIsALM–Chappell.pdf
32. W.J. Satzinger, R.B. Jackson, S.D.Burd. "Systems Analysis and Design in a Changing World", sixth edition, Course Technology, 2011
33. R.Phalnikar, V.S. Deshpande, S.D. Joshi. "Applying Agile Principles for Distributed Software Development", Proceeding of IEEE International Conference on Advanced Computer Control, pp 535-539, 2008
34. A.Qumer, B.Henderson-Sellers. "An evaluation of the degree of agility in six agile methods and its applicability for method engineering", Information and software technology, 50(4), 2008, 280-295
35. S. Rajagopalan and S.K. Mathew. "Choice of Agile Methodologies in Software Development: A Vendor Perspective," Journal of International Technology and Information Management: Vol. 25: Iss. 1, Article 3., 2016
36. K.Beck with C.Andres. "Extreme Programming Explained: Embrace Change", Second Edition, Addison-Wesley Boston, ISBN 0-321-27865-8, 2004
37.] N.Wirth. "A Plea for Lean Software", in IEEE Computer, Vol. 28, no. 2, February 1995, pp 64-68.
38. M.Poppendieck. "Lean Software Development", in Proceeding of 29th IEEE International Conference on Software Engineering, 2007
39. Atlassian Inc. "Kanban. How the kanban methodology applies to software development", official Atlassian site https://www.atlassian.com/agile/kanban
40. K. Hong. "DevOps (Development and Operations)", 2016, Online: http://www.bogotobogo.com/DevOps
41. S. W. Ambler, Guest Editor. "Disciplined Agile Delivery: The Foundation for Scaling Agile", Cutter IT Journal, Vol. 26, No. 11, November 2013
42. S. Joshi. "Five things to understand about enterprise devops solutions", 2015, Online: http://devops.com/2015/08/20/five-things-tounderstand-about-enterprise-devops-solutions/
43. J.L.Marechaux. "Top 10 reasons to use Bluemix and DevOps services", 2015 [Online]. Available: http://www.ibm.com/developerworks/library/ d-10-reasons-to-use-bluemix-devops-trs/index.html
44. C. Hooper. "Concurrent engineering: The foundation of DevOps", 2011, Online: http://www.charleshooper.net/blog/concurrentengineering-the-foundation-of-devops/
45. Wikipedia. "DevOps". [Online]. Available: https://en.wikipedia.org/wiki/DevOps
46. OMG. "(UML) Unified Modeling Language (UML®) v. 2.5", Object Management Group, 2015

47. A. Lapouchnian. "Goal-Oriented Requirements Engineering: An Overview of the Current Research", Technical report, Department of Computer Science, University of Toronto, 2005, Online: http://www.cs.utoronto.ca/ alexei/pub/Lapouchnian-Depth.pdf
48. D. Amyot, A. Shamsaei, J. Kealey, E. Tremblay, A. Miga, G. Mussbacker, M. Alhaj, R. Tawhid, E. Braun, and N. Cartwright. "Towards Advanced Goal Model Analysis with jUCMNav", Proc. of Fourth International Workshop on Requirements, Intentions and Goal in Conceptual Modeling (RIGiM 2012), 2012
49. I.M. Ramirez, M. Vergne, M. Morandini, L. Sabatucci, A. Perini and A.Susi. "Where Did the Requirement Come from? A Retrospective Case Study", Proc. of Fourth International Workshop on Requirements, Intentions and Goal in Conceptual Modeling (RIGiM 2012), 2012
50. A. Sutcliffe and P. Sawyer. "Requirements Elicitation: Towards the Unknown Unknowns", Proc. of the 21st IEEE International Conference on Requirements Engineering (RE'13), Rio de Janeiro, Brazil, pp. 92 - 104, 2013
51. A. van Lamsweerde. "Goal-oriented requirements engineering: a guided tour", Proc. of the 5th IEEE International Symposium on Requirements Engineering, Toronto, pp. 249 - 262, 2001
52. D. Amyot. "jUCMNav v7.0.0", Online: http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome
53. L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. "Non-Functional Requirements in Software Engineering", Kluwer, 2000
54. E. Yu. "Towards modelling and reasoning support for early-phase requirements engineering", Proc. of the Third IEEE International Symposium on Requirements Engineering, pp. 226-235, 1997
55. A. van Lamsweerde and E. Letier. "From object orientation to goal orientation: A paradigm shift for requirements engineering", Proc. of Radical Innovations of Software and Systems Engineering in the Future, ser. Lecture Notes in Computer Science, 2004
56. A. van Lamsweerde. "Requirements engineering in the year 00: A research perspective", Proc. of the 22nd International Conference on Software Engineering, (ICSE - 00). New York, pp.5-19, 2000
57. D. Quartel, W. Engelsman, H. Jonkers, and M. van Sinderen. "A goal oriented requirements modelling language for enterprise architecture", Proc. of IEEE International Enterprise Distributed Object Computing Conference (EDOC 09), pp. 313, 2009
58. D. CHAPPELL. "WHAT IS APPLICATION LIFECYCLE MANAGEMENT?", COPYRIGHT 2008 CHAPPELL & ASSOCIATES, SPONSORED BY MICROSOFT CORPORATION, 2008
59. Wikipedia. "Microsoft Visual Studio", Online: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio#Application_Lifecycle_Management
60. MSDN. "DevOps and Application Lifecycle Management". Microsoft, 2016, online https://www.visualstudio.com/en-us/docs/vsts-tfs-overview
61. Sanjay. "Atlassian Tool Suite for ALM" March 29, 2016
62. C. Rolland, J.Horkoff, E. Yu, C.Salinesi and J. Castro. "Preface to the Proceeding of Fourth International Workshop on Requirements, Intentions and Goal in Conceptual Modeling"(RIGiM)", 2012

63. Z.151 ”User Requirements Notation (URN) Language definition”, Recommendation ITU-T Z.151 (10/2012),Online: https://www.itu.int/rec/T-REC-Z.151-201210-I/en, 2012
64. K.W. Engelsman and R. Wieringa. ”Goal-Oriented requirements engineering and enterprise architecture: two case studies and some lessons learned”, Proceedings of the 18th international conference on Requirements Engineering: foundation for software quality (REFSQ’12), pp. 306-320, 2012
65. R. Ali, F. Dalpiaz and P. Giorgini. ”Reasoning with contextual requirements: Detecting inconsistency and conflicts”, in Journal Information and Software Technology, Volume 55 Issue 1, January, pp. 35-57, 2013
66. ACI Informatica. ”DICET-INMOTO - ORganization of Cultural HEritage for Smart Tourism and Real-time Accessibility (OR.C.HE.S.T.R.A.)” - Project funded by the Italian Ministry of Education, University and Research (MIUR) - PON Project - Research and Competitiveness 2007-2013, 2015.
67. Poste Italiane. ”District of cyber security”, Online: https://www.distrettocybersecurity.it, 2016
68. J. Lind. ”Issues in agent-oriented software engineering”, in AgentOriented Software Engineering, Lecture Notes in Computer Science, Eds. Springer Berlin Heidelberg, vol. 1957, pp. 45-58, 2001
69. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. ”Tropos: An agent-oriented software development methodology”, In Autonomous Agents and Multi-Agent Systems, vol. 8, no. 3, pp. 203-236, 2004
70. F. Zambonelli, N. R. Jennings, and M. Wooldridge. ”Developing multiagent systems: The Gaia methodology”, ACM Trans. Softw. Eng. Methodol., vol. 12, no. 3, pp. 317-370, 2003
71. P. Caire, N. Genon, P. Heymans and D. L. Moody. ”Visual Notation Design 2.0: Towards User Comprehensible Requirements Engineering Notations”, Proc. of the 21st IEEE International Conference on Requirements Engineering (RE’13), Rio de Janeiro, Brazil, pp. 115 - 124, 2013
72. OMG. ”Business Process Model And Notation TM (BPMNTM) v. 2.0”, Object Management Group, Spec. formal/2011-01-03, 2011. [nline: http://www.omg.org/spec/BPMN/2.0/
73. T. Hill and R. Westbrook. ”SWOT analysis: It’s time for a product recall”, Long Range Planning, vol. 30, no. 1, pp. 46 - 52, 1997.
74. OMG. ”SysML V1.4 Specification Release”, Online: http://www.omgsysml.org/, 2016
75. DICET-INMOTO. ”Rapporto di ricerca dell’attività A2 1.2 Allegato4”, ACI Informatica, 2014
76. D. Getz. ”Progress in Tourism Management Event tourism: Definition, evolution and research”, ScienceDirect, Tourism Management, pp. 403-428, 2008
77. ECB. ”Business continuity oversight expectations for systemically important payment systems (SIPS)”, Report of European Central Bank, 2006, On line: https://www.ecb.europa.eu/pub/pdf/other/businesscontinuitysips2006en.pdf
78. CPSS and IOSCO. ”Principles for financial market infrastructures”, Press release ISBN 92-9197-108-1, Bank for International Settlements Committee on Payment and Settlement Systems (CPSS) and IOSCO Technical Committee, 2012
79. D. Goldsmith, M. Siegel. ”Systematic approaches to cyber insecurity”, Technical report, MIT Sloan School of Management1 - ECIR Working Paper, 2012

80. DTCC. ”Cyber Risk a Global Systemic Threat: A White Paper to the Industry on Systemic Risk”, White paper, Depository Trust & Clearing Corporation (DTCC), October 2014.
81. DTCC. ”Systemic risk barometer: results overview”, 2016 Q1. Press release, Depository Trust & Clearing Corporation (DTCC), 2016
82. D. Rissman. ”US regulators issue guidance on disaster recovery and business continuity planning for hedge funds”, 2013, Online: http://aceits.net/us-regulators-issueguidance- on-disaster-recovery-and-business-continuity-planning-for-hedge-funds/
83. P. Sommer. ”Reducing systemic cybersecurity risk”, Oecd/ifp project on future global shocks, Information Systems and Innovation Group, London School of Economics and Ian Brown, Oxford Internet Institute, Oxford University, 2011. Online: https://www.oecd.org/gov/risk/46889922.pdf
84. R. Tendulkar. ”Cyber-crime, securities markets and systemic risk”, Joint staff working paper, IOSCO Research Department and World Federation of Exchanges, 2013 Online: http://www.iosco.org/research/pdf/swp/Cyber-Crime-Securities-Markets-and-Systemic-Risk.pdf
85. BankofItaly. ”Guidelines on business continuity for market infrastructure”, Guidelines, Bank of Italy, 2014. Online: https://www.bancaditalia.it/compiti/sispaga-mercati/codise/Guidelines business continuity market infrastructures.pdf
86. Weforum. ”The global risks report 2016”, Insight report, 11th World Economic Forum, 2016, Online: http://www3.weforum.org/docs/Media/TheGlobalRisksReport2016.pdf
87. E. Cerutti, S. Claessens, P. McGuire. ”Systemic risks in global banking: What available data can tell us and what more data are needed?”, Working Paper 18531, National Bureau of Economic Research, November 2012, Online: http://www.nber.org/papers/w18531
88. A.Furfaro, A. Garro, A. Tundis. ”Towards Security as a Service (SecaaS)”, On the modeling of Security Services for Cloud Computing 2014 Int. Carnahan Conf. on Security Technology (ICCST), pp 1-6, 2014
89. F. Buccafurri, L. Fotia, A. Furfaro, A. Garro, M. Giacalone, A. Tundis. ”An Analytical Processing Approach to Supporting Cyber Security Compliance Assessment”, Proceedings of the 8th International Conference on Security of Information and Networks, ACM, pp. 46-53, 2015
90. G. Naacke, R.Tendulkar. ”Cyber-crime, securities markets and systemic risk”, OICV-IOSCO and WFE, TechReport, OICV-IOSCO and WFE, 2013
91. F. Liu, J. Tong, J. Mao, R.Bohn, J.Messina, L. Badger, D. Leaf. ”NIST Cloud Computing Reference Architecture” National INstitute of Standards and Technology, TechReport, 2011
92. CSA. ”Security Guidance for Critical Areas of Focus in Cloud Computing V3.0”, Cloud Security Alliance, 2011,
93. CSA. ”SecaaS — Defined Categories of Service 2011”, Cloud Security Alliance, 2011
94. D. Swift. ”Successful SIEM and Log Management Strategies for Audit and Compliance”, SANS Institute, 2010
95. A. Garro, A Tundis. ”On the Reliability Analysis of Systems and SoS: the RAMSAS method and related extensions”, IEEE Systems Journal (IJS), vol. 9 (1), pp. 232-241, 2015.

96. A. Garro, A Tundis. "Modeling and simulation for system reliability analysis: The RAMSAS method", Proc. 7th International Conference on System of Systems Engineering (SoSE), 2012
97. F. Cicirelli, A. Furfaro, L. Nigro. "A DEVS M&S framework based on Java and actors", Proc. of 2nd European Modelling and Simulation Symposium, pp. 337-342, Barcelona (Spain), October 4-6, 2006
98. A. Shalloway. "An Overview of Agile Methods", 2012, Online: http://www.netobjectives.com/files/resources/articles/ OverviewOfLeanAgileMethods.pdf
99. K. Hong. "DevOps (Development & Operations)", 2016, Online: http://www.bogotobogo.com/DevOps/DevOps_Jenkins_Chef_Puppet_Graphite_Logstash.php
100. S. Joshi. "Five things to understand about Enterprise DevOps Solutions", 2015, Online: http://devops.com/2015/08/20/five-things-to-understand-about-enterprise-devops-solutions/
101. C. Edeki. "Agile Unified Process International Journal of Computer Science and Mobile Applications", 1, pp 13-17, 2013
102. I. OREN. "New Essential SaFe Guidance Article", 2016, Online: http://www.scaledagileframework.com/new-essential-safe-guidance-article/
103. S. Agrawal. "CA Technologies transforms DevOps with Application Centric Infrastructure", 2014, Online: http://blogs.cisco.com/datacenter/ca-technologies-transforms-devops-with-application-centric-infrastructure
104. G. Menzel. "DevOps - The Future of Application Lifecycle Automation", 2014, Online: https://www.capgemini.com/blog/capping-it-off/2014/12/devops-the-future-of-application-lifecycle-automation
105. J. L. Marechaux. "Top 10 reasons to use Bluemix and DevOps Services", 2015, Online: http://www.ibm.com/developerworks/library/d-10-reasons-to-use-bluemix-devops-trs/index.html
106. V. G. K. Pammi. "Agile User Stories – The Building Blocks for Software Project Development Success", 2013, Online: https://www.scrumalliance.org/community/articles/2013/september/agile-user-stories
107. G. Lucassen, F.Dalpiaz, J. M. E. van der Werf, S. Brinkkemper. "Forging high-quality User Stories: Towards a discipline for Agile Requirements", In Proc. IEEE 23rd Int. Requirements Engineering Conf. (RE), pp 126-135, 2015
108. A. Gunasekaran. "Concurrent engineering: a competitive strategy for process industries", Journal of the Operational Research Society, 49, pp 758-765, 1998
109. C. Hooper. "Concurrent Engineering: The Foundation of DevOps", 2011, Online: http://www.charleshooper.net/blog/concurrent-engineering-the-foundation-of-devops/
110. J. Humble, J. Molesky, B. O'Reilly. "Lean Enterprise – How High Performance Organizations Innovate at Scale", O'Reilly Media, 2014
111. T. Demos, D. Macmillan. "Australian Software-Tools Maker Atlassian Planning", U.S. IPO, 2015, Online: http://www.wsj.com/article_email/australian-software-tools-maker-atlassian-planning-u-s-ipo-1443213301-lMyQjAxMTE1NDI3NTAyNjU5Wj
112. GeNIe (Graphical Network Interface). Available: http://genie.sis.pitt.edu/
113. Ed. Snible. "Active Deploy & Canary Advisor", InterConnect2016, IBM Research, February 21-25, Las Vegas, 2016

114. M.R V. Chaudron, W. Heijstek, A. Nugroho. "How effective is UML modeling? An empirical perspective on costs and benefits", Softw Syst Model (2012) 11:571?580, © Springer-Verlag 2012
115. L.Kuzniarz, M.Staron, C.Wohlin. "An Empirical Study on Using Stereotypes to Improve Understanding of UML Models", Proceedings of the 12th IEEE International Workshop on Program Comprehension (IWPC?04), 2004
116. A. Tundis, S. Citrigno, T. Gallo, A. Garro, S. Graziano, D. Saccá, M. Mühlhäuser. "Systemic Risk Modeling and Evaluation through Simulation and Bayesian Networks", ARES'17, Reggio Calabria, Italy, August 29-September 01, 2017 (accepted)