Business Modeling and Requirements in RUP: A Dependency Analysis of Activities, Tasks and Work Products

Carina Campos[1], José Eduardo Fernandes[2(✉)], and Ricardo J. Machado[3]

[1] Dept. de Sistemas de Informação, Universidade do Minho, Guimarães, Portugal
carina.camposl3@gmail.com
[2] Polytechnic Institute of Bragança, Bragança, Portugal
jef@ipb.pt
[3] Centro ALGORITMI, Escola de Engenharia, Universidade do Minho, Guimarães, Portugal
rmac@dsi.uminho.pt

Abstract. Most artifacts developed during the requirements engineering process relate themselves in different ways. In order to understand in detail how they affect each other during the software development process, it is relevant to identify their interdependencies. This paper presents a systematization of the existing interdependencies between the different elements of the Rational Unified Process (RUP) in the Business Modeling and Requirements disciplines. This work, which highlights knowledge about the different interdependencies and traceability of RUP elements, is useful to avoid unconscious decisions during software the development process and also, to detect potential problems due to the violation of the existing interdependencies.

Keywords: Requirements engineering · Interdependencies · Traceability · Artifacts · RUP

1 Introduction

Software systems, one of the most complicated things developed by humankind, became an essential part of our everyday lives and from which we are completely dependent on. Their existence is so pervasive that, nowadays, society expects them to assist us either in critical activities or in our everyday activities, to have high quality, to provide exciting functionalities, to be reliable, and to be produced at low cost.

Given the increasing demand and complexity of software systems, the software engineering discipline has accumulated, over the last years, an extensive scientific body of knowledge related to theories, methods, approaches, and tools needed to construct software systems [1]. To cope with the growing complexity and diversity of engineering problems, the adoption of systematic and disciplined approaches to deal with requirements and their problems has become crucial [1]. The study and research about requirements is extensive in literature, covering diverse areas, such as the design of

O. Gervasi et al. (Eds.): ICCSA 2016, Part V, LNCS 9790, pp. 595–607, 2016.
DOI: 10.1007/978-3-319-42092-9_45

requirements modeling languages [2] or model transformation techniques and code generation [3]. Recently, Fernandes and Machado [1] addressed the essence, issues, and techniques for requirements in engineering projects. Requirements engineering, in the context of the development of a system through an engineering project, embodies a set of activities that permits eliciting, negotiating, and documenting the functionalities and the restrictions of that system [1]. As an engineering discipline, is closely related to the concept of project; it is throughout the project that the engineer applies his technical and scientific knowledge, to solve the problems and to achieve the objectives which he is confronted with [1]. In turn, the concept of project is closely related to the concept of process.

Most individual requirements developed during the requirements engineering process relate to and affect each other in different ways and thus cannot be treated in isolation [4, 5]. The fact that the requirements relate to and affect each other makes it necessary to identify and manage the requirements interdependencies in order to avoid potentially costly mistakes during the system development.

Requirements interdependencies are not a problem by themselves, but they influence the number of development activities and decisions made during the software engineering process [6]. Traceability is the basis for studying the requirements interdependencies during the development process [7] since it allows identifying and justifying the artifacts that implement the requirements initially formalized.

Software development produces various kinds of artifacts. The artifacts, such as requirements, do not exist in isolation; instead they relate to and affect each other [8]. During the development of solutions and also during the exploration phase for maintenance issues, frequently arises the need to introduce several changes to the project decisions previously established. These changes should be clearly identified to ensure the complete identification of the artifacts involved in the changes. To this end, it is necessary to have knowledge about how the different artifacts relate among them since it facilitates the identification of the artifacts affected.

RUP is a process that provides the best practices and guidelines for successful software development [9]. This work, in the context of Business Modeling and Requirements disciplines of RUP, analyzes and systematizes the traceability and the interdependencies that may occur between the various elements during software development projects.

This paper has the following structure: Sect. 2 presents the importance of dealing with the interdependencies and the traceability during the software development; Sect. 3 describes the interdependencies and the traceability between the different elements of the RUP; Sect. 4 presents the conclusions.

2 Interdependencies and Traceability

Requirements traceability is an issue that for long time is investigated and discussed, such as in [10]. Dahlstedt and Persson [7] refer to traceability as “a basis for addressing the requirements interdependencies”. According to Genvigir [11] and Zou et al. [12], traceability is intimately associated to the software production process, specifically to

the requirements and to the ability to establish links between these requirements and other artifacts that satisfy them.

Sánchez et al. [13] mention that the requirements traceability aims to help determine the impact of changes in the conception phase of software, to support their integration, preserve the knowledge and assure the quality and correction of the global system.

Requirements traceability is as a quality factor [6, 14–16]. Actively supporting traceability in a software development project can help ensuring other qualities of software, such as adequacy and understandability [15].

On the other hand, neglecting the traceability can lead to less maintainable software and to failures due to inconsistencies and omissions [15]. Dömges and Pohl [17] refer to neglecting the traceability or capture insufficient and/or unstructured traces leads to decrease in system quality, causes revisions, and hence, increases project costs and time.

Aizenbud-Reshef et al. [18] refer that, from the perspective of requirements management, traceability facilitates the interconnection of requirements to their origins and reasons. Additionally, it allows capturing the information needed for understanding the evolution of requirements and for verification of requirements fulfillment. Complete traceability allows calculate more accurately the costs, as well as to determine lists of changes, without depending on the programmer knowledge of all the areas that these changes affect [18]. All these reasons make crucial to implement traceability practices throughout the software development.

It is essential to identify and manage the interdependencies that occur throughout the system development in order to, if needed, in any context, to properly consider related artifacts and as such, to avoid potentially costly mistakes by neglecting either those relations or eventually relevant artifacts As mentioned earlier, through traceability, it is possible to manage these interdependencies; hence, traceability is fundamental to the development process.

Several works further develop traceability practices and theories [19, 20]. Marques et al. propose a traceability representation language [21] that provides “abstractions to requirements, artifacts and trace links as well as queries, through which trace links can be searched, retrieved and filtered”. In addition, they also propose a requirement traceability process [22] specifying its workflow, actors, responsibilities and inputs/outputs. Rempel and Mäder [23] review the elements involved in establishing traceability in a development project and derive a quality model for the systematic assessment of requirements traceability. To facilitate traceability in the model centric paradigm, Badreddin and Sturm [24] call for representing requirements as first class entities aiming to enable software developers, modelers, and business analysts to manipulate requirements entities as textual model and code elements. In this context, they propose a Requirement-Oriented Modeling and Programming Language (ROMPL). Soonsongtanee and Limpiyakorn [25] present an approach to enhancing the requirements traceability matrix with UML state diagrams to describe the traceability states of associated requirements or life-cycle work products. Regarding requirements interdependencies, Berrocal et al. [26] present a set of profiles to allow designers to explicitly model interdependencies between elements in BPMN 2 and UML 2 Use Case diagrams. They also define ATL transformations to automatically derive these relationships from the business specification to the requirements models.

The purpose of dealing, systematically, with requirements interdependencies improves the decisions made during software development as well as to detect the potential problems that may arise because of the requirements interdependencies [6]. Managing requirements interdependencies consists in identify, store and maintain information about how the requirements relate to and affect each other [6].

Maintain traceability of the requirements interdependencies is essential in order to support various situations and activities in the system development process [7]. Traceability should be included and treated along the development projects, thus representing, an asset to their success. Knowing the whole story of the artifacts, as well as their interdependencies, will enable easier identification and management of existing interdependencies from the early stages of development. Therefore, this knowledge minimizes problems that may arise during the software development process.

3 Interdependencies and Traceability in RUP

RUP aims to ensure the production of quality software that meets the needs and expectations of its users in a predictable schedule and cost [9]. RUP guidelines entail several elements such as activities, tasks, roles and work products. Throughout the development process, at several moments, RUP elements become interconnected; by this way, a simple change in an element causes various subsequent adjustments in others. Therefore, the knowledge of existing interdependencies between the various elements is particularly useful since it allows easier identification of elements affected during a change.

As Dahlstedt and Persson [7] refer, is essential to maintain the traceability of interdependencies since it allows to know, in detail, how the elements relate, as well as to support various situations and activities in the software development. Through the traceability of various elements of RUP, it is possible to easier identify and manage the interdependencies that may occur between elements. For those practitioners that adopt RUP guidelines, it is useful to understand the interdependencies that may exist between the various RUP elements.

3.1 Dependency Analysis of Activities and Tasks

RUP is organized in various disciplines and phases. However, the study mentioned in this paper focuses in two transitions (see Fig. 1): (1) from the Business Modeling discipline to the Requirements discipline, at Inception phase; (2) from the Inception to the Elaboration phase, within the Requirements discipline.

To facilitate an overview and analysis of all tasks of RUP (for Business Modeling and Requirements disciplines), the conduction of an initial RUP review allowed the construction of information presented in Tables 1 and 2.

These tables show the different tasks of the disciplines of Business Modeling and Requirements, the activities associated with these tasks, the phase where they are performed, and the roles responsible for them. Table 1 details the activities, tasks,

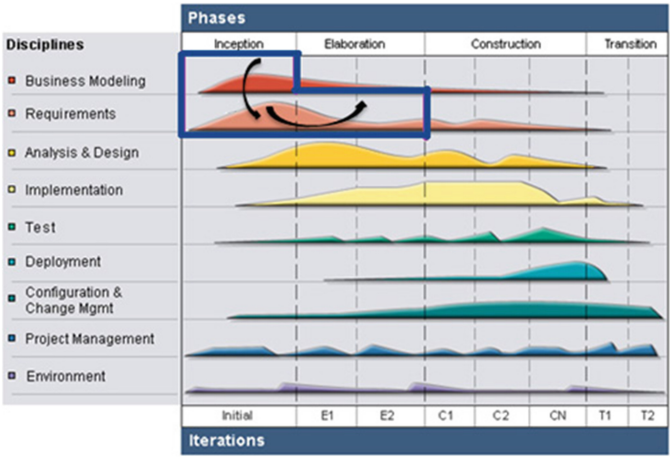

Fig. 1. Positioning of the study in the RUP (Based on [27]) (Color figure online)

Table 1. Activities, tasks, phases and roles of the business modeling discipline.

	Activities					Tasks	Phases				Role main
	B1 Assess Business Status αβ	**B2** Describe Current Business β	**B3** Define Business β	**B4** Explore Process Automation β	**B5** Develop Domain Model α		Inception	Elaboration	Construction	Transition	
Business Modeling	x					Assess Target Organization - **ATO**	B1				Business-Process Analyst
	x	x	x		x	Business Architectural Analysis - **BAA**	B1, B2, B5	B2, B3	B3	B3	Business Architect
			x			Business Operation Analysis		B3		B3	Business Designer
			x			Business Operation Design		B3	B3	B3	Business Designer
			x			Business Use-Case Analysis		B3	B3	B3	Business Designer
	x	x	x		x	Capture a Common Business Vocabulary - **CCBV**	B1, B2, B5	B2, B3	B3		Business-Process Analyst
				x		Construct Business Architectural Proof-of-Concept		B4			Business Architect
				x		Define Automation Requirements		B4	B4	B4	Business Designer
			x			Define Business System Context		B3			Business-Process Analyst
			x		x	Detail a Business Entity - **DBE**	B5		B3	B3	Business Designer
			x			Detail a Business Use Case			B3		Business Designer
			x			Detail a Business Worker			B3	B3	Business Designer
		x	x			Find Business Actors and Use Cases - **FBAUC**	B2	B2, B3	B3		Business-Process Analyst
	x	x	x			Identify Business Goals - **IBG**	B1, B2	B3			Business-Process Analyst
	x	x	x		x	Maintain Business Rules - **MBR**	B1, B2, B5	B2, B3	B3	B3	Business-Process Analyst
			x			Prioritize Business Use Cases		B3	B3		Business Architect
			x		x	Review the Business Analysis Model - **RBAM**	B5	B3	B3	B3	Technical Reviewer
			x			Review the Business Use-Case Model		B3	B3		Technical Reviewer
	x	x	x			Set and Adjust Objectives - **SAO**	B1, B2	B3			Business-Process Analyst
			x			Structure the Business Use-Case Model		B3	B3		Business-Process Analyst

	Activities
Classic RUP Lifecycle	α
Business Modeling Lifecycle	β
Classic RUP Lifecycle + Business Modeling Lifecycle	αβ

phases, and roles of the Business Modeling discipline considering both processes of Classic RUP Lifecycle and Business Modeling Lifecycle.

The column Activities presents the five activities performed in this discipline. The activities performed in the Classic RUP Lifecycle process are signaled in the table by an α, the activities performed in the Business Modeling Lifecycle process are signaled by an β and the activities performed in both processes are signaled in the table by αβ.

Table 2. Activities, tasks, phases and roles of the Requirements discipline.

	Activities						Tasks	Phases				Role main
	R1	R2	R3	R4	R5	R6						
	Analyze the Problem	Understand Stakeholder	Define the System	Manage the Scope of the	Refine the System	Manage Changing Requirements		Inception	Elaboration	Construction	Transition	
	αβ	αβ	α	α	α	α						
Requirements	x	x	x				Capture a Common Vocabulary - **CCV**	R1α, R2α, R3			R1β, R2β	System Analyst
					x		Detail a Use Case - **DUC**		R5			Requirements Specifier
					x		Detail the Software Requirements - **DSR**		R5			Requirements Specifier
	x						Develop Requirements Management Plan - **DRMP**	R1α				System Analyst
		x	x		x		Develop Supplementary Specifications - **DSS**	R2α, R3	R5		R2β	System Analyst
	x	x	x	x			Develop Vision - **DV**	R1α, R2α, R3, R4			R1β, R2β	System Analyst
		x					Elicit Stakeholder Requests - **ESR**	R2α			R2β	System Analyst
	x	x	x				Find Actors and Use Cases - **FAUC**	R1α, R2α, R3			R1β, R2β	System Analyst
		x	x	x		x	Manage Dependencies - **MDep**	R2α, R3, R4	R6	R6	R6	System Analyst
				x			Prioritize Use Cases - **PUC**	R4				Software Architect
						x	Review Requirements - **RReq**		R6	R6	R6	Technical Reviewer
						x	Structure the Use-Case Model - **SUCM**		R6	R6	R6	System Analyst

	Activities/ Phases
Classic RUP Lifecycle	α
Business Modeling Lifecycle	β
Classic RUP Lifecycle + Business Modeling Lifecycle	αβ

Column Tasks exposes all tasks practiced in the Business Modeling discipline. Only the tasks with a gray background were studied, since the other stand in phases that are outside the scope of our study. The intersection of column Activities with the lines of column Tasks indicates (through an 'x') the tasks included in the activities.

Column Phases presents which phases include the different tasks and activities. The abbreviations B1, B2, B3, B4 and B5 (for the various activities) associate tasks and their activities to the several phases. Column Role main refers which are the roles responsible for performing the different tasks. The activities with blue background (Assess Business Status, Describe Current Business and Develop Domain Model) and the phase (Inception) refer to the activities and the phase studied in Business Modeling discipline.

Table 2 presents the activities, tasks, phases, and roles of the Requirements discipline. Column Activities presents the six activities practiced in this discipline. As before, in the table, α signals activities performed in the Classic RUP Lifecycle process, β signals activities executed in the Business Modeling Lifecycle process, and αβ signals the activities performed in both processes.

Column Tasks exposes all tasks practiced in the Requirements discipline. In this discipline, all tasks have a gray background since they are in the phases of the scope of this study and as such, covered by this study. An 'x' at the intersection of column Activities with the lines of column Tasks indicates the tasks practiced in the activities.

Column Phases presents tasks and activities performed in the different phases. Abbreviations R1, R2, R3, R4, R5 and R6 (for the various activities) associate the tasks and their activities to phases where they are performed. In the intersections, we use α for Classic RUP Lifecycle, β for Business Modeling Lifecycle and αβ for both processes. The intersections show the process where the tasks and their activities are performed.

As in the previous table, the column Role main refers to the roles responsible for performing the different tasks.

The activities with blue background (Analyze the Problem, Understand Stakeholder Needs, Define the System, Manage the Scope of the System, Refine the System Definition and Manage Changing Requirements) and the phases (Inception and Elaboration) refer to the activities and the phases studied in Requirements discipline.

The information provided in these tables is useful throughout the software development because it allows to perceive how activities, tasks, and roles relate in a particular discipline and phase.

3.2 Dependency Analysis of Work Products and Tasks

The elaboration of the previous two tables allowed to perceive the tasks and activities covered in the disciplines and phases considered in this study. Tables 3 and 4 have the purpose of clarifying the interconnection of all work products of both disciplines to their respective tasks.

The first column of Table 3 shows all the work products of the Business Modeling discipline and the second column presents all the tasks. Only the tasks with a gray background were analyzed because the other tasks are in phases that are not within the scope our study.

The intersection of these two columns depicts the work products consumed and produced in the various tasks. These intersections use the terms IN, OUT and I/O: the term IN is used to refer work products consumed by the associated task; the term OUT represents the work products produced by the task in question; the term I/O represents that the work products are both consumed and produced by the task in question. Besides these terms, the term IN* refers to work products that are an optional entry of the associated task; these work products are not necessarily consumed in the task. In the tables, the use of colors facilitate the identification of terms IN, OUT, and I/O: the term IN is represented by the green color, the term OUT by the red color and the term I/O by the yellow color.

The first column, Table 4 shows all the work products of the Requirements discipline and in the second column presents all the tasks. All the tasks of this discipline were analyzed because all the tasks are in phases that are within the scope our study. The intersection of these two columns depicts which work products are consumed and produced in the various associated tasks. These intersections use the terms IN, OUT and I/O, which were previously defined.

Tables 3 and 4 show the work products produced and consumed by the different tasks. The information available in these tables allows the identification of existing interdependencies between tasks and work products that are produced and consumed.

Figures 2 and 3 present two graphical representations that were developed to enhance the perception of the information contained in the previous tables; i.e., all the existing interdependencies between the activities and the tasks and work products of a given phase and discipline. This visualization facilitates the analysis of the existing interdependencies along the development process, thus allowing for a better understanding and management.

Table 3. Work products of the tasks of the Business Modeling discipline.

Discipline	Business Modeling																			
Tasks / Work Products	Assess Target Organization - **ATO**	Business Architectural Analysis - **BAA**	Business Operation Analysis	Business Operation Design	Business Use-Case Analysis	Capture a Common Business Vocabulary - **CCBV**	Construct Business Architectural Proof-of-Concept	Define Automation Requirements	Define Business System Context	Detail a Business Entity - **DBE**	Detail a Business Use Case	Detail a Business Worker	Find Business Actors and Use Cases - **FBAUC**	Identify Business Goals - **IBG**	Maintain Business Rules - **MBR**	Prioritize Business Use Cases	Review the Business Analysis Model - **RBAM**	Review the Business Use-Case Model	Set and Adjust Objectives - **SAO**	Structure the Business Use-Case Model
Business Analysis Model - **BAM**		OUT	I/O	IN	OUT				I/O								IN			
Business Architectural Proof-of-Concept - **BAP-C**		IN *			IN *		OUT													
Business Architectural Document - **BAD**		OUT			IN		IN									I/O				
Business Deployment Model - **BDM**		OUT		OUT			IN													
Business Design Model - **BDesM**		OUT		OUT			IN													
Business Glossary - **BGl**						OUT		IN												
Business Goal - **BG**											IN			OUT						
Business Rule - **BR**															OUT					
Business Use Case Model - **BUCM**			I/O	IN	IN				I/O				OUT			I/O		IN		I/O
Business Vision - **BV**		IN				IN							IN	IN	IN	IN			OUT	
Supplementary Business Specification - **SBS**					IN				IN		OUT		OUT							
Target-Organization Assessment - **TOA**	OUT							IN											IN	

OUT	Output
IN	Input
I/O	Input/Output
IN *	The work product is an optional input of the associated task

Table 4. Work products of the tasks of the Requirements discipline.

Disciplines	Requirements												
Tasks / Work Products	Capture a Common Vocabulary - **CCV**	Detail a Use Case - **DUC**	Detail the Software Requirements - **DSR**	Develop Requirements Management Plan - **DRMP**	Develop Supplementary Specifications - **DSS**	Develop Vision - **DV**	Elicit Stakeholder Requests - **ESR**	Find Actors and Use Cases - **FAUC**	Manage Dependencies - **MDep**	Prioritize Use Cases - **PUC**	Review Requirements - **RReq**	Structure the Use-Case Model - **SUCM**
Glossary - **Gl**	OUT											OUT
Requirements Attributes - **ReqA**		OUT	OUT		OUT	OUT	OUT	OUT	OUT	OUT		OUT
Requirements Management Plan - **RMP**				OUT					I/O			
Software Requirement - **SofR**			OUT							OUT	IN	
Software Requirements Specification - **SRS**			OUT									
Stakeholder Requests - **SR**					IN	IN	OUT	IN				
Storyboard - **St**							OUT					
Supplementary Specifications - **SS**					OUT							OUT
Use-Case Model - **UCM**								OUT		IN		I/O
Vision - **Vi**			IN			OUT			OUT			

OUT	Output
IN	Input
I/O	Input/Output
IN *	The work product is an optional input of the associated task

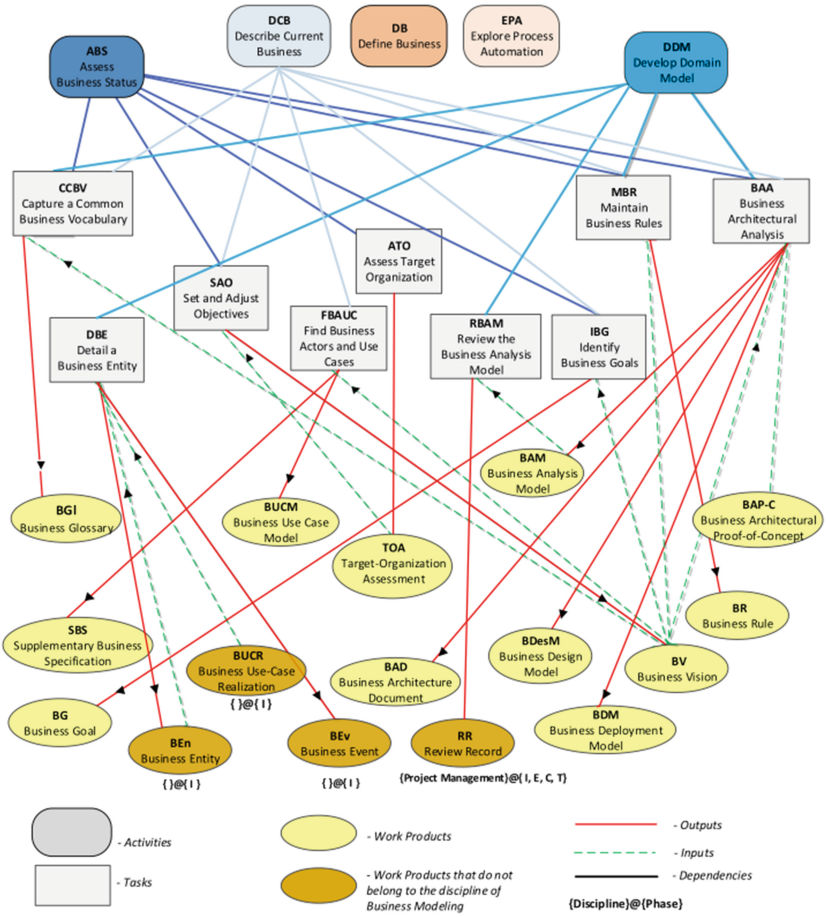

Fig. 2. Scheme Business Modeling@Inception (Color figure online)

The representation of the Tables 1 and 3. This representation refers to the Business Modeling discipline in the Inception phase. It presents the five activities belonging to this discipline. These activities interconnect to their tasks; two of these activities have no associated tasks because they are outside the scope of this study.

Each task has its associated work products. These work products may be consumed in the associated task (inputs, graphically represented by arrow green) or may be produced by that task (outputs, graphically represented by arrow red). The work products represented in yellow refers to work products belonging to the Business Modeling discipline.

The work products, represented in orange, despite being work products produced and consumed in this discipline|phase, do not belong directly to work products defined by RUP for this discipline. For these work products (in orange), a description below

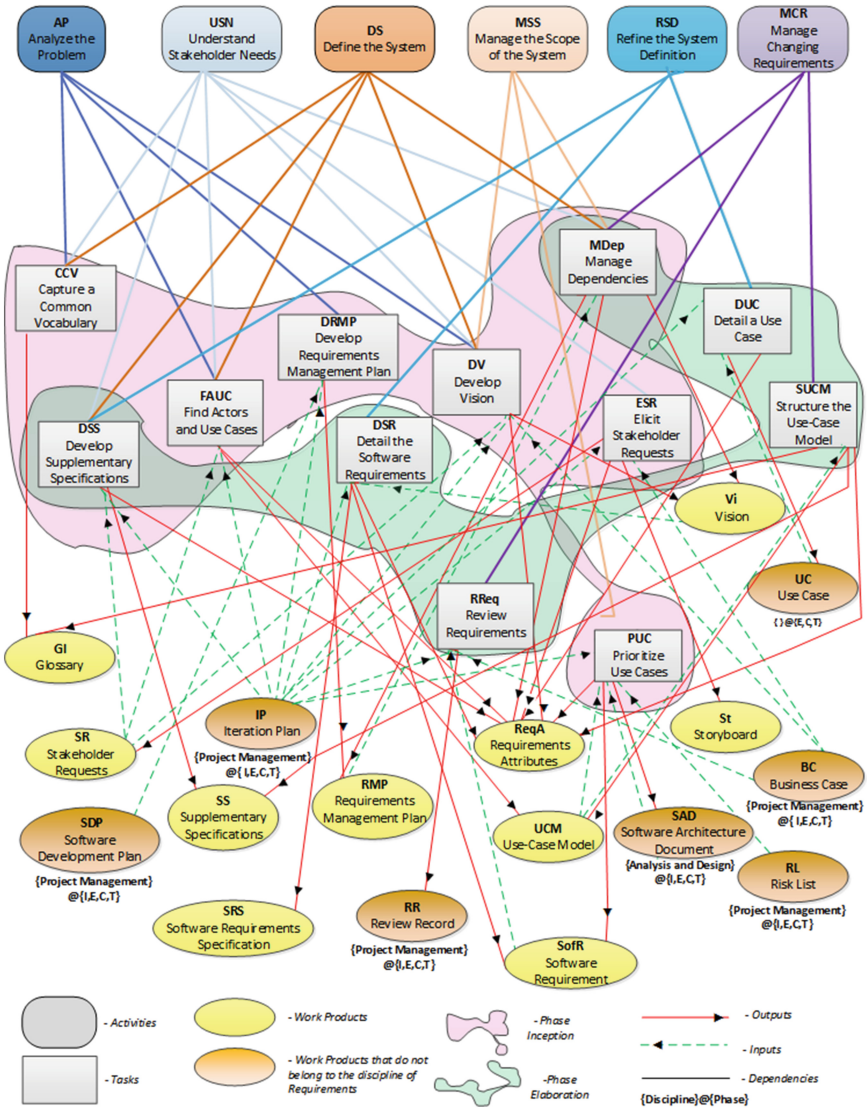

Fig. 3. Scheme Requirements@Inception, Elaboration

them indicates the discipline and the phase to which they belong; some of those do not have associated discipline because, in concrete, they do not belong to any.

The representation of the Fig. 2 is based on information gathered in the Tables 2 and 4. This representation refers to the Requirements discipline in the Inception and Elaboration phases. It presents the six activities belonging to this discipline, as well as its interconnected tasks. All these activities have associated tasks because all of them

are within the scope of this study. Each task has its associated work products. These work products may be consumed in the associated task (inputs, graphically represented by arrow green) or may be produced by that same task (outputs, graphically represented by arrow red). The work products represented in yellow refers to work products belonging to the Requirements discipline.

The colored background areas allow perceiving that two tasks are handled in both phases, thus verifying that there are interdependencies between the phases. The tables built facilitate the identification of interdependencies, not only among activities, tasks, phases and the roles, but also among tasks and work products, of the disciplines under consideration.

These tables, as well as the graphical representations allow analyzing the traceability of various elements of RUP, as well as easily identifying all the existing interdependencies between those elements. This becomes particularly useful since it allows knowing in detail how the various elements of the RUP process are related.

The information provided in these tables and representations, improve the practitioner's capacity in dealing with the impact of changes and in supporting better development decisions.

This systematization of the interdependencies is also useful to compare a particular method/process model with the RUP since it allows knowing in detail how the RUP is organized. The study of traceability and of the interdependencies between the various elements of the RUP may be extended to all disciplines and phases that compose this process. The expansion of the study will allow detailing how the various elements are related throughout the whole RUP process.

4 Conclusions

RUP is a process that provides best practices and guidelines for successful software development. However, this does not provide any information that enables for easy identification of traceability and existing interdependencies between the various elements that constitute it. Throughout the software development, this can become a problem since there is no explicit native information on RUP documentation on the inter-relation of RUP elements.

Our work produced several tables and graphical representations in order to highlight how the various RUP elements are related. These tables and graphical representations allow, from the initial phases of development, an easier identification of the various interdependencies and the traceability among elements, as well as to provide a deeper knowledge about the organization of RUP. This is quite advantageous since it is possible to avoid unconscious decisions during the development process as well as to detect early potential problems due to the existing interdependencies.

Acknowledgments. This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

1. Fernandes, J.M., Machado, R.J.: Requirements in Engineering Projects. Springer, Cham (2016)
2. Ivan, J.: The Design of Requirements Modelling Languages. Springer, Cham (2015)
3. Smialek, M., Nowakowski, W.: From Requirements to Java in a Snap. Springer, Cham (2015)
4. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Nattoch Dag, J.: An industrial survey of requirements interdependencies in software product release planning. In: Fifth IEEE International Symposium on Requirements Engineering, pp. 84–91. IEEE Press (2001)
5. Regnell, B., Paech, B., Aurum, A., Wohlin, C., Dutoit, A., Nattoch Dag, J.: Requirements mean decisions! – research issues for understanding and supporting decision-making in requirements engineering. In: First Swedish Conference on Software Engineering Research and Practice (SERP 2001), pp. 49–52 (2001)
6. Dahlstedt, Å.G., Persson, A.: Requirements Interdependencies – State of the Art and Future Challenges. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements. LNCS, pp. 95–116. Springer, Heidelberg (2005)
7. Dahlstedt, Å.G., Persson, A.: Requirements interdependencies - molding the state of research into a research agenda. In: Ninth International Workshop on Requirements Engineering: Foundation for Software Quality, pp. 55–64 (2003)
8. Heindl, M., Biffl, S.: A case study on value-based requirements tracing. In: 10th European Software Engineering Conference, pp. 60–69. ACM, New York (2005)
9. Kruchten, P.: Tutorial: introduction to the rational unified process. In: 24th International Conference on Software Engineering (ICSE 2002), pp. 703–703. ACM, New York (2002)
10. Gotel, O.C.Z.: An analysis of the requirements traceability problem. In: 1st International Conference on Requirements Engineering, pp. 94–101. IEEE Press (1994)
11. Genvigir, E.C.: Um Modelo para Rastreabilidade de Requisitos de Software Baseado em Generalização de Elos e Atributos. Instituto Nacional de Pesquisas Espaciais (2009)
12. Zou, X., Settimi, R., Cleland-Huang, J.: Improving automated requirements trace retrieval: a study of term-based enhancement methods. Empirical Softw. Eng. **15**(2), 119–146 (2010)
13. Sánchez, P., Alonso, D., Rosique, F., Álvarez, B., Pastor, J.A.: Introducing safety requirements traceability support in model-driven development of robotic applications. IEEE Trans. Comput. **60**(8), 1059–1071 (2011)
14. Ramesh, B., Jarke, M.: Toward reference models for requirements traceability. IEEE Trans. Softw. Eng. **27**(1), 58–93 (2001)
15. Winkler, S., Pilgrim, J.V.: A survey of traceability in requirements engineering and model-driven development. Softw. Syst. Model **9**(4), 529–565 (2010)
16. Spanoudakis, G., Zisman, A.: Software traceability: a roadmap. In: Chang, S.K. (ed.) Handbook of Software Engineering and Knowledge Engineering, vol. 3, pp. 395–428. World Scientific Publishing, Singapore (2005)
17. Dömges, R., Pohl, K.: Adapting traceability environments to project-specific needs. Commun. ACM **41**(12), 54–62 (1998)
18. Aizenbud-Reshef, N., Nolan, B.T., Rubin, J., Shaham-Gafni, Y.: Model traceability. IBM Syst. J. **45**(3), 515–526 (2006)
19. Huang, J., Gotel, O., Zisman, A. (eds.): Software and Systems Traceability. Springer, London (2012)
20. Turban, B.: Tool-Based Requirement Traceability Between Requirement and Design Artifacts. Springer, Wiesbaden (2013)

21. Marques, A., Ramalho, F., Andrade, W.L.: TRL: a traceability representation language. In: 30th Annual ACM Symposium on Applied Computing, pp. 1358–1363. ACM, New York (2015)
22. Marques, A., Ramalho, F., Andrade, W.L.: Towards a requirements traceability process centered on the traceability model. In: 30th Annual ACM Symposium on Applied Computing, pp. 1364–1369. ACM, New York (2015)
23. Rempel, P., Mäder, P.: A quality model for the systematic assessment of requirements traceability. In: 23rd IEEE International Requirements Engineering Conference (RE), pp. 176–185. IEEE Press (2015)
24. Badreddin, O., Sturm, A.: Requirement traceability: a model-based approach. In: 4th IEEE International Model-Driven Requirements Engineering Workshop (MoDRE), pp. 87–91. IEEE Press (2014)
25. Soonsongtanee, S., Limpiyakorn, Y.: Enhancement of requirements traceability with state diagrams. In: 2nd International Conference on Computer Engineering and Technology (ICCET), pp. V2-248–V2-252. IEEE Press (2010)
26. Berrocal, J., Alonso, J.G., Chicote, C.V., Murillo, J.M.: A model-driven approach for documenting business and requirements interdependencies for architectural decision making. IEEE Lat. Am. Trans. **12**(2), 227–235 (2014)
27. IBM, Rational Method Composer (version 7.1). http://www-03.ibm.com/software/products/en/rmc