Combining Models, Diagrams and Tables for Efficient Requirements Engineering: Lessons Learned from the Industry

Christophe Ponsard [1], **Robert Darimont** [2], **Arnaud Michot** [1]

1. *CETIC Research Centre*
   Rue des frères Wright 29/3, 6041 Gosselies, Belgique
   {christophe.ponsard,arnaud.michot}@cetic.be

2. *Respect-IT S.A.*
   Place des Peintres, 7/201, 1348 Louvain-la-Neuve, Belgique
   robert.darimont@respect-it.be

ABSTRACT. *Requirements Engineering (RE) involves a number of artifacts of different nature (models, structured text, tables, diagrams). Capturing and maintaining the relationships between those artifacts is currently not an easy task but definitely helps in both improving the quality and level of automation of requirements specification. Through several industrial applications, this paper looks at how the interactions between RE artifacts happens in practice and how to provide better tool support for them. Our experimental framework used KAOS model-based goal-oriented approach to investigate different industry use scenarios like identifying concepts within source documents or producing requirements/evaluation/cost estimates tables from the RE model. As a result of our findings, we have implemented related extensions to a goal-oriented RE tool.*

RÉSUMÉ. *L'ingénierie des exigences (IE) implique de nombreux artefacts de nature diverse tels que modèles, texte structur'e, diagrammes et tables). Capturer et maintenir les relations entre ces artefacts reste une tâche difficile mais qui a un impact véritable à la fois sur la qualité et le niveau d'automatisation de la production d'un cahier des charges. Sur base de nombreuses applications industrielles, ce papier décrit comment les interactions entre ces différents artefacts s'organisent en pratique et comment leur fournir un meilleur soutien outillé. Notre cadre expérimental s'est appuyé sur la méthodologie orientée but KAOS afin d'étudier différents scénarios industriels tels que l'identification de concepts dans les documents sources ou la production de tables d'exigences/evaluation/estimation de coûts. Le résultat de cette étude à mené à l'implémentation d'extensions dans un outil d'ingénierie d'exigence.*

KEYWORDS: *Requirements Engineering, goal-orientation, industry case studies, tool support*

MOTS-CLÉS : *Ingénierie des exigences, orientation buts, cas d'étude industriels, support outillé*

1. Introduction

Requirements Engineering (RE) is a process involving many activities such as elicitation, analysis, specification, validation and management as shown in Figure 1. Several artifacts are involved in this process either as input, output or internal part of the process (Abran *et al.*, 2001)(Sommerville, 2010). Those artifacts can take various forms in order to best support the related activity. The elicitation phase typically relies on interview transcripts whiteboard pictures or existing documentation from which requirements are identified. The analysis phase will structure those requirements either directly in a text processor or using an explicit model supported by a dedicated tool. The specification phase will produce system and/or software requirements specification usually taking the form of well structured documents according to company specific templates or standards like IEEE830 (IEEE, 1998). Those documents are generally not only composed of text but also typically rely on diagrams and tables. In order to convey the information in the most convenient format for its accurate description and understanding. A number of diagram notations has been agreed upon and standardised for this purpose, for example UML use cases/sequence diagrams/statecharts, SysML requirements diagrams, Business Process Modelling Notation (BPMN). Likewise, tabular notations are very popular in order to list requirements together with their characteristics. Dedicated tabular notations are also widely used to precisely describe specific aspects of the system such as its behaviour. For example state transition table, SCR tables, or IO tables.

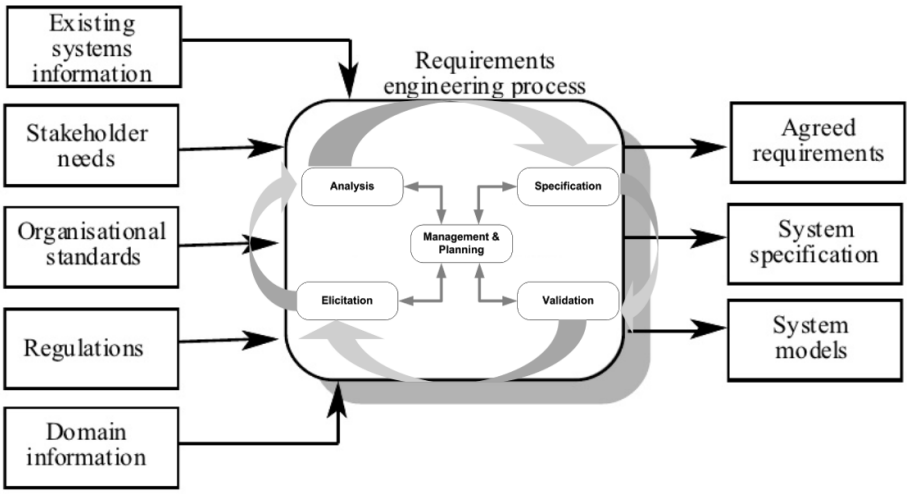

Figure 1. Requirements Engineering Process (adapted from Sommerville)

Across the different activities of the requirements engineering process, several artifacts of various forms are produced, processed, transformed and combined often iteratively. For example considering the validation of the first version of the requirements specification, some new requirements might be identified and need to be included in the requirements model. From there, specific diagrams and tables need to

be updated and re-included in an updated version of the requirements specification. Although a number of existing tools can trace the impact of such changes, they are often mostly limited to the management of documents and some kind of tables (IBM, 2002)(Dassault Systems, 2010). However they fail to address a number of scenarios, like involving diagrams, language translation, and performing quantitative processing like costs estimates.

The purpose of this paper is to present a consolidated view of interaction scenarios involving those different RE artifacts. Those scenarios were identified after conducting several real world requirements specification projects together with a systematic review of popular diagrammatic and tabular notations used in the field. Those experimentations also drove the development of specific extensions to a model based toolset supporting the KAOS goal oriented requirements engineering method (Lamsweerde, 2009)(Respect-IT, 2005). In order to ease the application of what we learned to other methods and tools, our paper describes our implementation using some abstractions for the requirements meta-model, document structure and way to query information.

Table 1. Overview of the Industrial Case Studies

Year	Domain	Location	Purpose	#reqs	#pages	Notations used
2004	Accessibility	Belgium	Domain model	185	108	goal diagrams, structured text, ranking table
2006	Grid computing	Europe	SRS	52	137	Structured text, req. tables, sequence diagrams
2007	Child care	Brussels	As-Is	126	182	BPMN, statecharts, tables (high level goals)
2009	Parliaments	Belgium	Call for tender	187	127	Structured text, req. tables, goal diagrams
2010	Banking	Brussels	internal templates	N/A	N/A	Structured text, context/goal/BMPN/information diagrams
2011	Smart Cards	Brussels	test plan	N/A	230	Command tables, Finite State Machines
2012	Child care	Brussels	Call for tender	173	267	Structured text, Event Process Chains, Use cases, req tables
2013	Electronic Nurse record	Brussels	Call for tender	223	151	Structured text, process models, req tables and diagrams
2014	Cloud computing	Europe	SRS, arch.	200	187	Structured text, req. tables, sequence diagrams
2014	Accounting	Belgium	SRS	210	170	Structured text, req tables and diagrams

As a support for our survey and experimentation work, our paper relies on about ten large requirements specifications involving big industrial players in Belgium and at international level. Table 1 give a overview of the size and contest of each case study. For confidentiality reasons, only the application domain and geographic location is mentioned. Our paper will also focus more specifically on three of those cases in order to illustrates some scenarios: the Belgian parliamentary administrations, an industrial Cloud interoperability project and a smart cart development project.

This paper is structured as follows. In section 2, we state the problem by performing a survey of popular ways to describe artifacts used within the requirements engineering process. Specific scenarios are identified and specified as user stories. In section 3, a consolidated view of all those user stories is proposed and some abstraction in order to be able to describe their relations in an implementation indepen-

dent way is introduced. In section 4, we describe the concrete implementation of the prototype we used to carry out or experimentations. In section 5, we illustrate our experimentation on a number of scenarios using excerpts of real world requirements specifications. In section 6, a critical discussion is carried out to point the achievements and areas of improvements, also in the light of related work. Finally in section 7, we draw some conclusions and discuss future work.

2. Survey of Artifacts used in the Requirements Engineering Process

This section reviews different kinds of artifacts of different forms already mentioned in the introduction. The identification is strongly based on the artifacts used in our industrial experience as described in Table 1. It is consolidated with literature material. The goal is not to be exhaustive in identifying those artifacts but rather to identify interesting ways those artifacts combine and interact within the requirements engineering process. For this purpose, we will make use of a user story template of the following form which will enable to capture both the interaction requirement but also the key actor and the underlying purpose.

As $< agent >$, **I need** $< interaction\_requirement >$ **so that** $< goal >$

2.1. Document Structure

Having a well-defined structure for the Requirements Document (RD) is important both from the RD user and RD producer. Standardised structures have been published as international standard such as the IEEE830, now replaced by the (IEEE, 1998; ISO/IEC/IEEE, 2011) or public templates like Volere (Atlantic Guild, 2014). Specific translation constraints may also apply to multilingual requirements documents. This results in the following user stories:

– US1: **As** RD User, **I need** to have a agreed well-structured document **so that** I can easily find my way in it and have confidence about the document quality.

– US2: **As** RD Producer, **I need** to have an agreed well-structured document **so that** I can make sure everything is covered and structured according to domain usage.

– US3: **As** RD Translation Manager, **I need** to extract textual output in a specific table format **so that** the translation process can be easily managed.

2.2. Diagram Notations

Diagram notations are very popular in software and system engineering and several notations have been standardised like UML for modelling software related aspects (OMG, 2011b), SysML for system-wide modelling (OMG, 2011b) or BPMN for modelling business processes (OMG, 2011a). Specific diagram notations of those standards are useful for requirements engineering such as the UML use case, UML and SysML state diagrams and BPMN. SysML also includes a simple requirements

diagram. However more elaborated requirements models are supported by specific methods such as i*/URN (Yu, Mylopoulos, 1997), KAOS (Lamsweerde, 2009) or GSN (Kelly, Weaver, 2004). Figure 2 illustrates a part of the goal model for a Cloud Management System used as case study in section 5. It shows the refinement of a goal into requirements and their assignment to specific software components.

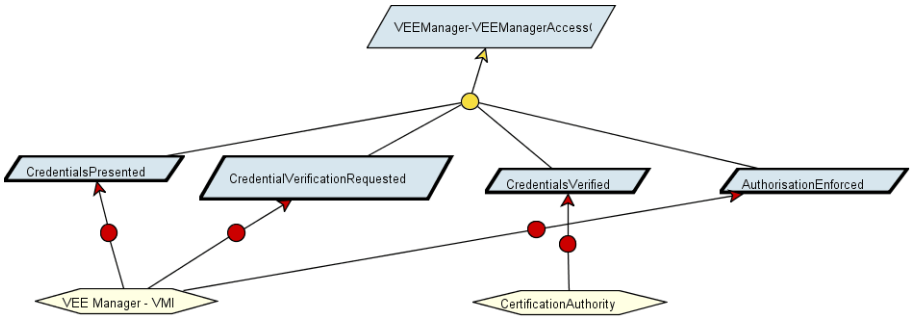

Figure 2. Example of Goal Model for a Cloud Management System

Considering the availability of such models it is also important in order to ease the way they are inserted in text documents. It also helps to automate the generation of specific byproducts, for example if the diagram is providing a specific view on information available in the requirements engineering model. This results in the following user stories:

– US4: **As** RD Producer, **I need** to be able to easily insert and update manually maintained figures in dedicated part of the RD **so that** the update effort is low.

– US5: **As** RD Produced, **I need** to be able to automate the generation of the diagrams that can be derived from information available in the RE model **so that** the update is efficient and consistency is guaranteed.

2.3. Tabular Notations

Tabular notations are very popular in requirements engineering and specification because they carry a lot of information in one structural form that the human reader can easily scan, understand and check. Just like the case of diagrams, an number of standard tabular notations are commonly used. The most common form of table is the requirements table listing requirements by tailing specific attributes such as their ID, short name, definition, priority, etc. Table 2 shows a typical requirements table found in the CCDS case.

Table 2. Example of Requirements Table

Id	Name	Definition	Resp.	Prior.	Ref.
R22	Initial Risk Assessment Provided	The Reservation Scheduling maintains for each running job a risk data structure that was initialised during the negotiation.	Risk Assessment	High	P1
R23	Dynamic Change in Risk Reported	The Extended Monitoring process monitors critical parameters about the resources and reports dynamic changes.	Extended Monitoring	High	P3

More specific notations are also widely used, such as state transition tables which describes state diagrams or SCR tables based on the Parnas four variables model or IO tables (Herrmannsdorfer *et al.*, 2008) (Hummel, Thyssen, 2009). Those tabular notations are more specific to the specification of control systems. Notice also that such tables can be redundant with diagrams and should be consistent with them. An example of a state-transition table extracted from the EMV case is shown in Figure 3.

	Selected	Initiated	Online	Script
GetProcessingOptions	Initiated if SW = 9000 Selected if SW != 900	NOTALLOWED	NOTALLOWED	NOTALLOWED
InternalAuthenticate	NOTALLOWED	Initiated	NOTALLOWED	NOTALLOWED
PinChangeUnblock	NOTALLOWED	NOTALLOWED	NOTALLOWED	Script
PutData	NOTALLOWED	NOTALLOWED	Online	Script
ReadRecord	Selected	Initiated	Online	Script
Select	Selected	Selected	Selected	Selected
UpdateRecord	NOTALLOWED	NOTALLOWED	Online	Script
Verify	NOTALLOWED	Initiated	NOTALLOWED	NOTALLOWED
GetData	Selected	Initiated	Online	Script
GetChallenge	Selected	Initiated	Online	Script
GenerateAC	NOTALLOWED	Online if SW = 9000 and SW = ARQC Script if SW = 9000 and SW != ARQC Selected if SW != 9000	Script if SW = 9000 Selected if SW != 9000	NOTALLOWED
ApplicationUnblock	NOTALLOWED	NOTALLOWED	Online	Script

Figure 3. Example of State-Transition Table for the EMV Smart Card Specification

This results in the following user stories:

– US6: **As** RD Producer, **I need** to be able to automatically generate requirements tables from the RE model **so that** the document can easily be updated.

– US7: **As** RD Producer, **I need** to be able to maintain the consistency between table, RE model and related diagrams **so that** the document consistency is guaranteed.

3. Consolidated Interaction Scenarios and Problem Abstraction

3.1. Summary table

Based on the previous section, we identified four main artifact types which are: RE models, text documents, diagrams and tables. Using the identified user stories we can populate a table showing how each kind of artifact can interact with another one in an input/output relationship. Furthermore, we can also systematically question all the gaps to identify interesting missing interactions. Scenarios identified during that "gap analysis" mainly relate to model transformations, to the use of table to automate large update of specific model attributes (e.g. validated priority information) and to more quantitative reasoning typically carried out through some spreadsheet output.

The final result of this process is shown in Table 3 which references the above user stories. In this table, the lines are the source artifacts while the columns are the target artifacts. Previously identified user stories are referenced using their identification number.

Table 3. Input/Output scenario combinations

In/Out	RE Model	Text Document	Diagram	Table
RE Model	Export (package, variant)	US1/2: explanation Concept reference	US5: generate goal breakdown Responsibility...	US6:Req. table Traceability Matrix US3: translation export
Text Document	Page reference Concept creation	US4: illustration Extraction	–	–
Diagram	(Model editing)	Referenced concepts	(Format conversion)	US7:Content listing
Table	Automate update (batch import)	US3: translation import	Layout view from query	Cost estimates

3.2. Problem Abstraction

In order to ease the description of how the above interaction scenarios can be best supported, we introduce some abstractions for the description of the different kinds of artifacts as well as some operations that can be carried out on them. This will also enable a better reuse of the experiment approach using different tools.

RE Models. The requirements engineering model is always structured according to some underlying meta-model. Such a meta-model will typically capture concepts like goals, requirements, software components (or agents), and processed information (e.g using a class model). In the scope of this paper, we will use the KAOS meta-model which is depicted in Figure 4.

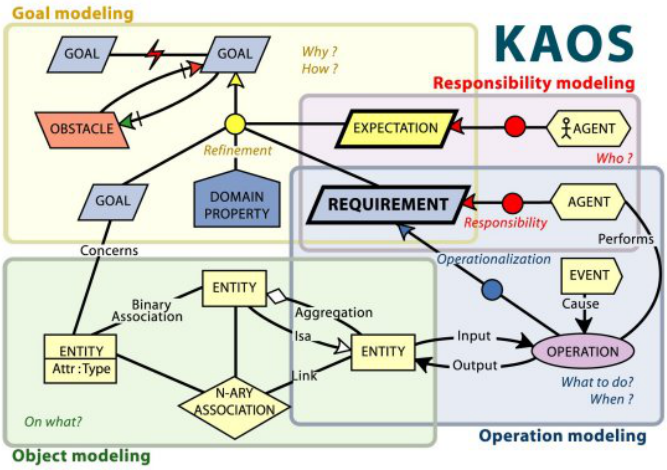

Figure 4. The KAOS Meta-model

Documents. We consider documents as a linear structure composed of a sequence of sections. Each section can start with a (sub)title and combine contents composed of texts, diagrams or tables as described by the following simple BNF grammar.

```
Document_Structure = Section*
section  = [title], Content*
Content  = Text | Diagram | Table
```

A typical operation is the generation of a concrete document based on a given document structure:

```
GENERATE Document_Structure AS Target_Format
Target_Format = Text | Table
```

Diagrams can either be manually produced or automatically generated. An manually produced diagram has just some given reference whereas an automatically generated diagram results from the layout of concepts extracted from the RE model typically using some query which produces the list of identifiers to show in the diagram. A standard query language is OCL but in the scope of this paper we will use the OQL query language as it is the query language supported in our prototype and is quite similar to SQL (ODMG, 1998). Based on the identifier list, an adapted layout algorithm should be applied using a reliable layout engine (like GraphViz (AT&T Research, 2014)).

```
LAYOUT { Model_Concepts } AS Layout_Type
Layout\_Type = Hierarchical | Orthogonal | Circular
Model_Concepts = QUERY ON Model
```

Tables are also expressed as queries over the model as for diagrams. Those queries can specify a number of attributes that will be arranged in columns where each query result will form a line.

4. Prototype Description

4.1. Existing Tool Base

The prototype we used for our experimentation was implemented on top of the Objectiver goal oriented requirements engineering tool (Respect-IT, 2005).

Objectiver has a full support of the KAOS meta-model. It also includes a document meta-model as well as a report model. This was mapped on simple structure text in versions 2 and extended to OpenOffice (ODT format) in version 3. This also means that ODT documents can embed live references to model concepts using an URL of the type: "obj:\\unique_id". Thus a goal or requirement can be directly created from a source document or cross-referenced in some output documents. Those references are consistently and efficiently updated, including the production of a **[deleted concept]** reference in case a concept was removed from the model.

4.2. Extensions

A first extension we developed was adding spreadsheet support to the tool. We used the ODS format to keep it consistent with ODT textual format and ease the integration of both kind of content. ODS documents have a more limited meta-model than ODT which is mainly directed towards using it as output. For example, it is also not possible to embed concept reference within an ODS.

A second major work was on the report generator. This component takes as input a report instantiating the report model and produces either a linear textual document in the open document format or spreadsheet binder in the open spreadsheet format. The UNO API is used to produce those ODS based either Open Office or Libre Office (SUN Microsystems, 2009). Depending on the target format, the behaviour of the generator will be different. For example a query will resolut in the production of an additional spreadsheet tab in an ODS binder while it will result in the inclusion of the table in an ODT document.

Specific information is also stored back into the model for example the page where a requirement is first described is available from the model and will enable the production of traceability tables precisely pointing to that requirement.

Depending on the target format there might be some limitation for example as a binder are composed of a sequence of spreadsheets only the leaf content of the report structure is considered.

5. Experimentation

This section reports on experimentations on the interaction scenarios described in the previous section. The results of some scenarios are also reused in other scenarios to further demonstrate the chaining possibilities. In order to be concrete, we rely on the following three specific cases extracted from the list of presented earlier.

– The Parliamentary Access Control eXchange System (PACXS) enabled to share access control credential across different parliamentary assemblies in Belgium in order to share infrastructure costs and ease the authentication of deputies involved in multiple assemblies (CETIC, Respect-IT, 2009).

– a Cross Cloud Deployment System (CCDS) developed by the PaaSage FP7 project and synthesizing the requirements of four supporting case studies (PaaSage Consortium, 2013).

– The (public) specification of a smart card security module and of the agent interaction with the Europay-Mastercard-VISA environment (EMVCo, 2011) which was fed into a toolset dedicated for test plan development (Devos *et al.*, 2012).

5.1. Requirements tables

A requirements table can easily be generated using an OQL query. Depending on the structure of the report, requirements tables will be presented using different decompositions. They can follow either a goal breakdown structure, an agent breakdown structure or some other structure as recommended by standards such as the IEEE830 or ISO29148 (IEEE, 1998; ISO/IEC/IEEE, 2011). This will simply affect the way requirements are being fitted in the query. For example the following query returns all the requirements that are present in a specific goal diagram. The output of this query is similar to Table 2.

```
SELECT req.id, req.name, req. informalDef, req.priority
FROM Diagram AS d, d.elements AS g, g.concept AS req, req.directIsOf as t
WHERE d.name="MyGoalDiagram" and t.name="Requirement"
```

5.2. *Generating the Traceability for Requirements Assignment to Agents*

Producing a traceability matrix of how requirements(expectations) are assigned to specific software components (human agents) is important for different reasons:

– all requirements should be assigned to exactly one responsible agent

– figuring the agent load is important to assess the effort to develop it (for software) or the validate how the load can be managed by a responsible human agent

Such a traceability matrix can be produced by the simple following query with requirements as rows and agents as columns.

```
SELECT row.id, col.id
FROM Responsibility AS res, res.parent AS row, res.sons.son AS col
WHERE col.Category="Software"
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		IT-Manager	EndUser	Application-developper	SpeculativeProfiler	CrossCloudDeploymentManage	ExecutionWare	System-administrator	MetaDatabase	StochasticReasoner	HumanAnalyst	Adapter		TOTAL
71	CloudServiceIntegratedWithCustomerApplication	0	0	0	0	0	0	0	0	0	1	0		1
72	WorkflowKnown	0	0	0	0	0	0	0	0	0	0	0		0
73	DeploymentLocationPreferencesSpecified	0	0	0	0	0	0	0	0	0	1	0		1
74	DataReplicationDefined	0	0	0	0	0	0	0	0	0	0	0		0
75	NearOptimalDeploymentCalculated	0	0	0	0	0	0	0	0	1	0	0		1
76	DatabaseScalabilityDefined	0	0	0	0	0	0	0	0	0	1	0		1
77	MinCloudAdministrativeOverhead	0	0	0	0	0	0	0	0	1	0	0		1
78	LegacyApplicationsDeployed	0	0	0	1	0	0	0	0	0	0	0		1
79	TargetDeploymentEnvMappedToCloudProviders	0	0	0	0	0	0	0	0	1	0	0		1
80	CrossCloudDeployment	0	0	0	0	0	0	0	0	0	0	0		0
81														
82	TOTAL	0	0	0	7	7	0	0	0	34	17	3		

Figure 5. Responsibility Traceability Matrix for the CCDS Case Study

The resulting ODS document is shown in Figure 5 for our CCDS case. A total column is produced to easily check for the above constraints. In this case some requirements are not yet assigned to a responsible agent (see right column). Some identified agents have also currently no assigned requirements, while other are quite complex like the $StochasticReasoner$.

5.3. Generating Agent Assignment Diagrams

The DIAGRAM query to produce the list responsibility of responsibility is quite similar as the one for the traceability matrix except that it is filtering on a single agent and just returns a list of identifiers.

```
SELECT req.id
FROM Responsibility AS res, res.parent AS req, res.sons.son AS ag
WHERE ag.name="CrossCloudDeploymentManager"
```

A circular layout is then applied to this list to produce the diagram shown in Figure 6. Notice that for agents with a lot of responsibilities (e.g. more than 20) such diagrams become overloaded and too large so it is advised to keep to a table layout in that case.

```
LAYOUT { DIAGRAM } AS Circular
```

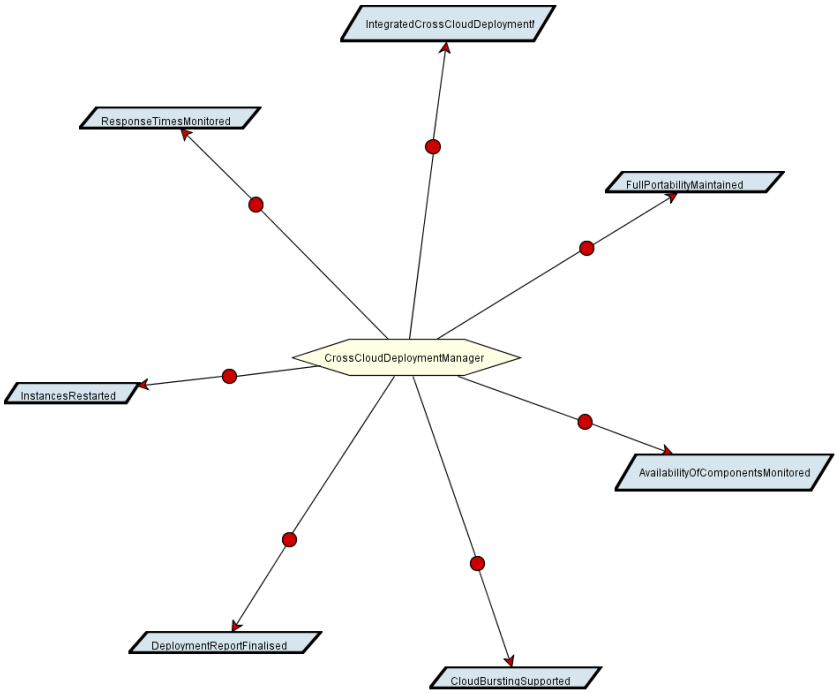

Figure 6. Responsibility Assignment for the Cross Cloud Deployment Management

5.4. Generating Evaluation Questionnaires

Another interesting scenario is to consider the procurement process, where candidates have to be evaluated against their ability to fulfil the requirements with their solution. Nice evaluation grids can be produced by grouping requirements by agents,

querying standard attributes like name or priority but also more specific attribute related to the actual reference of the requirement in the specification such as its number and page in the document.

```
SELECT req.numberInReport, req.name,
       req.priority, req.pageInReport, ag.name
FROM Responsibility AS res, res.parent AS req, res.sons.son AS ag
ORDER BY ag.name ASC
```

By using stylesheets and some macros, it is then possible to produce the kind evaluation questionnaire shown in Figure 7 where specific columns should be filled by the applicants. The returned results can then merged and compared in specific consolidation spreadsheets.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1					Reference	Name	Page	Module	Coverage	Parametrization	Standard development	Specific development	Availability	Comments
2									NWLT	-/1	-/1	-/1	Qi-YY	
49				2.1.4		Absences and overtime managed	36							
50				2.1.4.1		Absences managed	37							
51				G8.		Global administration of the absence management system supported	41							
52				R49.		Typology of absence types and status managed	42	Absence-Overtime Module	N					
53				R50.		Absences visualizable	42	Absence-Overtime Module	N					
54				R51.		Absence status overwritten by HR officers	42	Absence-Overtime Module	N					
55				R52.		Public holidays calendar managed	42	Absence-Overtime Module	N					

Figure 7. Example of Evaluation Questionnaire

5.5. Generating Effort Estimation Spreadsheets

Our final scenario is to perform some effort estimates and compare different possible options which can be specifying as variants in the RE Model. In the scope of this paper, we will consider that a specific effort field associated with each requirement is available. This can be filled in by some export. It can also be produced by an effort estimation method able to compute estimates from requirements. An example of such a method is the COSMIC Function Points (Dumke, Abran, 2010). The method requires all requirements to be enforcing using Create/Read/Update/Delete (CRUD) operations and is outside of the scope of this paper.

In order to produce the effort estimation, we will combine the following three spreadsheets using a tabular ODS output.

– EFFORT: a simple query to retrieve the requirement.effort

– ALLOCATION: the previously computed traceability matrix of Requirements vs Agents

– COMPUTATION: a specific spreadsheet performing the processing to compute module costs and applying styles on it

```
GENERATE [EFFORT,ALLOCATION,COMPUTATION] AS Tabular
```

The resulting spreadsheet is shown in Figure 8. The process was applied on two different variant of the system which should be specified as additional filters earlier in the process and that we will not detail here.

Req \ Module	Agent5	Agent3	Agent1
Requirement1	0	10	0
Requirement2	0	0	20
Requirement3	35	0	0
Requirement4	0	40	0
Requirement5	50	0	0
Requirement6	0	0	75
Requirement7	0	0	70
Requirement8	80	0	0
COST	165	50	165
TOTAL COST			380

Req \ Module	Agent5	Agent3	Agent1
Requirement1	0	10	0
Requirement2	0	0	20
Requirement3	35	0	0
Requirement4	0	40	0
Requirement5	50	0	0
Requirement6	0	0	75
Requirement9	100	0	0
COST	185	50	95
TOTAL COST			330

Figure 8. Effort Estimation for two Variants

6. Discussion and Related Work

We point discuss here some strong point emerging from our experiments as well as areas of improvement. We also bring some related work in the picture of this discussion to compare with other approaches and toolsets.

6.1. Strong points

– **Rich Model.** Using an underlying RE model provides rich traceability mechanisms and enables the automatic generation. This can be supported by most tools supporting a model-based approach like DOORS (IBM, 2002), Reqtify (Dassault Systems, 2010), Enterprise Architect (Sparx Systems, 2014) or Eclipse-based tools like TopCased (TopCased, 2011). Like in our case, the majority of tools provide some query mechanisms including predefined reports and finer grained queries (like OCL queries on Eclipse EMF models or SQL queries on the underlying database in Enterprise Architect).

– **Flexibility across artifacts.** The experience with the scenarios revealed a lot of flexibility resulting from the combination of interesting way to combine partial results, resulting in interesting chains of artifacts of different kinds. For example addressing the problem of reporting a concept page in a table revealed easy to support through minor evolution of the tool and relying on all the available transformation processes. In order to enable this, the tool should support a rich set of interconnection across different artifacts. With respect to this, most tools have limitations: DOORS and Reqtify are mainly targeting textual and table report generation but not copying with diagrams. Moreover Reqtify is only "readonly" and is not designed to reprocess its own outputs. Enterprise Architect fully supports the interplay of documents and diagrams. It also supports tabular views (e.g. on diagrams) and for import/export.

– **Ease of Combination** A key point to carry out our experimentations was the support for scenarios combination. In our case the main enabler was a visual report

functionality able to include all the manipulated artifacts and provide a specific behavior depending on the output format (e.g. including a query in a report will generate a formatted table in a document or a spreadsheet tab depending on the output format). Several tools only provide a report generator that are limited to the generation of standard template or require a lot of effort to customise their templates. As an alternative to a visual reporting tool, a good scripting API can also be efficient however it requires more effort to learn and to deploy.

6.2. Areas of Improvement

– **Connection with other models.** The richness of interaction scenarios also depends on the model. In our case it is rather limited to the goal-oriented RE. Other tools like cover larger domains such as design UML (Sparx Systems, 2014) or software product lines (University of Ottawa, 2001). The richer the model is, the greater the number of interesting interactions will be. Of course a tool must stay limited in scope and tool interconnectivity is also important (see later point).

– **Model Maintenance.** As it all model-based approaches corrections have be applied to the model and require regeneration. Tasks like locating and performing a correction in the model should be very easy to perform to avoid user rejection and will result in direct correction in documents that are not any more synchronised with the other artifacts. A simple recommended feature to reduce some corrections is to have a spell checker at the source of text capture.

– **Roundtrip Update.** More powerful roundtrip features could be investigated in order to be able to directly update from change in the documents. Most tools support the tagging of requirements description and attributes and support the synchronisation back to the model. However processing tables and diagrams from the target documents embedding them is more complex and often requires ad-hoc parsing. There is currently no easy solution for handling this at least considering traditional office suites.

– **Online tool.** The toolset we used as well as a number of major RE tools are still often based on a fat client architecture possibly connected to a database server. Nowadays toolsets are quickly evolving moving online in a SaaS mode either explicitly dedicated to requirements like (TraceCloud, 2014) or as evolution of other tools like bugtracker or wikis, e.g. JIRA and Confluence (Atlassian, 2014). Major office suite are already available online e.g. (Google, 2007), as well as diagramming toolsets like (Gliffy, 2009) (also available as Confluence plugin).

– **Interconnection protocol.** Tool integration is best achieved through an open interaction protocol like OSLC (Arwe, 2013). This protocol is now supported by several RE tools like DOORS (Jazz) or Enterprise Architect. However OSLC is based on the linked data principle and only provides the mechanisms to define resources in terms of RDF properties and query mechanisms. Achieving effective integration requires some work that could however be guided by our abstractions.

7. Conclusion and Perspectives

In the paper, we captured and documented a number of interesting interaction scenarios across several of artifacts types found in the requirements engineering process. This task relied on a careful review of a set of large requirements specification projects we carried our over the past ten years. We then looked at how well they were supported in a current state of the art RE tool. We prototyped a number of extensions to be able to cope with limitations and more importantly to experiment with combination and chaining of such scenarios in order to show interesting value added.

Our approach was proved positive, although some threats related to model-based engineering should be addressed. Interesting opportunities were also identified with the development of web-based solutions. We also tried to make our work reusable by others by describing our approach in abstract terms. We believe it is quite easily to transpose in a toolset providing a good extension API either closed (e.g. Enterprise Architect) or Open (e.g. Eclipse tools based on EMF like Papyrus, TopCased).

The developed extensions are part of the new V4 release of the Objectiver tool. Our future work is to support spreadsheets as input and to reference concepts within them. We are also studying more complex quantitative analysis of goal models. Finally we also plan to move to web-based tools and improve our tool interconnectivity using OSLC.

Acknowledgements

This work was funded by the Walloon Region INOGRAMS project (grant number 7171). We warmly thanks our industrial partners for sharing their cases.

References

Abran A., Bourque P., Dupuis R., Moore J. W. (Eds.). (2001). *Guide to the Software Engineering Body of Knowledge - SWEBOK*. Piscataway, NJ, USA, IEEE Press.

Arwe J. (2013, May). *Open Services for Lifecycle Collaboration, Core Specification Version 2.0.* http://open-services.net/bin/view/Main/OslcCoreSpecification.

Atlantic Guild. (2014). *Volere Requirements Template, Edition 17.* http://www.volere.co.uk/template.htm.

Atlassian. (2014). *JIRA and Confluence tools.* https://www.atlassian.com/.

AT&T Research. (2014). *Graphviz - Graph Visualization Software.* http://www.graphviz.org.

CETIC, Respect-IT. (2009, January). *Parliamentary Access Control eXchange System - Domain Description, Requirements Specification and Functional Specification.*

Dassault Systems. (2010). *Reqtify.* http://www.3ds.com/products-services/catia/capabilities/catia-systems-engineering/requirements-engineering/reqtify.

Devos N., Ponsard C., Deprez J.-C., Bauvin R., Moriau B., Anckaerts G. (2012). Efficient reuse of domain-specific test knowledge: An industrial case in the smart card domain. In *ICSE 2012*, p. 1123-1132. IEEE.

Dumke R., Abran A. (2010). *COSMIC Function Points: Theory and Advanced Practices*. Auerback Publications, Taylor & Francis LLC.

EMVCo. (2011, November). *EMV Integrated Circuit Card Specifications for Payment Systems - V4.3.* http://www.emvco.com/specifications.aspx?id=223.

Gliffy. (2009). *Diagrams Made Easy.* http://www.gliffy.com.

Google. (2007). *Google Docs.* http://www.google.com/docs/about.

Herrmannsdorfer M., Konrad S., Berenbac B. (2008, March). Tabular Notations for State Machine-Based Specifications. *CrossTalk The Journal of Defense Software Eng.*, No. 8.

Hummel B., Thyssen J. (2009). Behavioral Specification of Reactive Systems Using Stream-Based I/O Tables. In *Proc. of the 7th IEEE International Conference on Software Engineering and Formal Methods*, pp. 137–146. Washington, DC, USA, IEEE Computer Society.

IBM. (2002). *Rational DOORS.* http://www.ibm.com/software/awdtools/doors.

IEEE. (1998). *830-1998 - Recommended Practice for Software Requirements Specifications.*

ISO/IEC/IEEE. (2011). *29148-2011 - Systems and software engineering - Life cycle processes - Requirements engineering.*

Kelly T., Weaver R. (2004). The Goal Structuring Notation - A Safety Argument Notation. In *Proc. of dependable systems workshop on assurance cases.*

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

ODMG. (1998). *Object query language user manual v5.0.*

OMG. (2011a). *Business Process Model and Notation - version 2.0.* http://www.omg.org/spec/BPMN/2.0.

OMG. (2011b). *Unified Modelling Languages - 2.4.* http://www.omg.org/spec/UML.

PaaSage Consortium. (2013). *D6.1.1 Initial Requirements.* http://www.paasage.eu/images/documents/PaaSage-D6.1.1-Initial-Requirements.pdf.

Respect-IT. (2005). *Objectiver Requirements Engineering Tool.* http://www.respect-it.com.

Sommerville I. (2010). *Software engineering* (9th ed.). Harlow, England, Addison-Wesley.

Sparx Systems. (2014). *Enterprise Architect.* http://www.sparxsystems.com.au.

SUN Microsystems. (2009, April). *OpenOffice.org 3.1 Developer's Guide.* https://wiki.openoffice.org/w/images/d/d9/DevelopersGuide_OOo3.1.0.pdf.

TopCased. (2011). *Open Source Toolbox for Critical Systems.* http://www.topcased.org/.

TraceCloud. (2014). *Requirements Management - Agile, Waterfall, Change Control.* https://www.tracecloud.com.

University of Ottawa. (2001). *jUCMNav: Juice up your modelling.* http://jucmnav.softwareengineering.ca/twiki/bin/view/ProjetSEG/WebHome.

Yu E. S. K., Mylopoulos J. (1997, April). Enterprise modelling for business redesign: The i* framework. *SIGGROUP Bull.*, Vol. 18, No. 1, pp. 59–63.