Using Rational Unified Process in an SME – A Case Study

Geir Kjetil Hanssen¹, Hans Westerheim¹, and Finn Olav Bjørnson²

¹SINTEF ICT, N-7465 Trondheim, Norway

{geir.kjetil.hanssen, hans.westerheim}@sintef.no ²Norwegian University of Science and Technology, N-7491 Trondheim, Norway bjornson@idi.ntnu.no

Abstract. The Rational Unified Process (RUP) is a comprehensive software development process framework emphasizing use-cases, architecture focus and an iterative approach. RUP is widely known and many organizations have tried to adopt it. Being a framework, RUP has to, in some way, be tailored to the specific context of use, no software development project is alike. This paper presents a case study of a Norwegian SME that tried to adopt RUP in the simplest way, by introducing the methodology by providing comprehensive documentation and some simple training. Our study shows that the use of RUP had some positive effects but also that the use has been scattered. Interviews with users of RUP show that there is a great need of better training and practical support in getting most value out of RUP. The key message is that if you consider taking RUP into use you have to invest resources in it. Training and support are key success factors.

1 Introduction

The Rational Unified Process (RUP) is a software development process framework consisting of a more or less complete set of process elements for software development projects [1]. RUP defines a software development project as a set of disciplines, e.g. requirements handling, implementation etc., running from start to end through the whole project life cycle divided in a set of project phases. A project is performed by a group of actors, each having one or more well defined roles. Each role participates in one or more activities producing one or more artifacts. A discipline can run in iterations, that is, repetitions within a phase. Activities, roles and artifacts are the basic process elements of RUP. RUP is a prescriptive and plan driven methodology. As RUP is a comprehensive framework covering most aspects of a software development process it means that it in some way must be adapted to the situation of use, either ad-hoc for each project or in advance to produce a company wide standard.

In this paper we present a case study that describes the use of RUP in a company where no restrictions or guidelines were put on the use of RUP. The project managers and senior developers were given courses in RUP, and RUP Online (an electronic process guide on web) was purchased and installed. No common guidance for the use of RUP in projects was given. The company had no defined goals for introducing RUP; it was basically based on a belief that RUP would increase the professionalism in the company. The study has been conducted within a smaller Norwegian software development company. Three researchers have followed the company during a period of three years. This paper describes the experience from using RUP and derives some key conclusions that may be of use for others considering the use of RUP.

The paper has the following structure:

- The research method is described (collection of empirical data and data analysis).
- The research context of the case study, that is, the company, is described.
- The results part documents information and data collected. This includes descriptions of the usage of RUP as well as elements laying the ground for the forthcoming tailoring of RUP. Further on the results from the analysis of four projects and five interviews are documented.
- A discussion trying to clarify the key points from the analysis and giving a conclusion.

2 Research Method

2.1 Data Collection

The study has taken the form of a case study [2]. The research has been conducted by three external researchers mainly using project managers and software developers from the company as a source for data.

The first set of data was collected by interviewing project managers representing four projects. Prior to this series of interviews, the researchers prepared a spreadsheet that had a row for each role, activity and artifact described by RUP, grouped by the disciplines defined by RUP. A column was allocated to each project. The researcher conducting the interview asked the project manager about the use of every single role, activity and artifact in the actual project. If the element was used in the project as described by RUP, the actual cell was colored. If the item was used as described by RUP, but changed or replaced by a tailored element, the cell was colored and a comment was written about the change from the original item description in RUP. If the element was not used at all in the project, the cell was left blank.

The second set of data was collected by the means of semi-structured interviews with five other employees (each having experience with RUP from several various projects). The respondents had the following main responsibilities: 1) developer, 2) developer/project manager, 3) developer/project manager, 4) project manager/requirements engineer and 5) customer contact. Prior to the interviews, the researchers developed an interview guide. The guide consisted of questions with focus on their personal experience with using RUP across multiple projects to document a broader experience. The guide was open, allowing the respondents to freely discuss their experience.

These interviews were recorded and transcribed by the researchers. The transcriptions were reviewed by the interviewed objects, and possible corrections and clarifications were made.

2.2 Data Analysis

The spreadsheet documenting the use of RUP and the transcribed interviews were basis for the data analysis. From the beginning it was clear that the researchers were to use qualitative data analysis methods due to the nature of the data collected [3, 4].

Analysis of the spreadsheet. The spreadsheet was printed in 25% size of its actual size. This was done to get an *overview* of the RUP usage. The overview gave a clear visual picture of what parts of RUP which were really used. The comments in the spreadsheet were read through. The researchers tried to match comments with the non-use, to see if there might be some statements supporting the lack of usage. The RUP usage for the projects were also compared to the project definition, the scope of the project and the type of customer as a starting point for an understanding of the actual use and non-use of RUP elements.

Analysis of the interviews. The researchers used the constant comparison method [3] to identify the factors affecting the use of RUP among the project managers and senior developers. All the transcriptions were printed out, and each of the researchers got one copy each. The single transcriptions were read individually, and the researcher tagged statements in the documents which said something about use or non-use of RUP, reasons for using or not using RUP, and also positive and negative aspects with RUP itself. Then the researchers had a common work shop where the individual tagging were put onto a white board, and then compared. The comparison was the basis for a common summary of the interviews.

The main motivation for selecting this approach to data collection and data analysis was that the researchers did not have any pre-information about the use of RUP, and therefore no assumptions or hypothesis to test, thus a qualitative approach seemed appropriate. This motivation was supported by the relative low number of data points available from such a small company.

3 Research Context

The company described in this case is today a Norwegian software consultancy company with 50 employees, located in two different geographic offices.

They are mainly developing software systems with heavy back-end logic and often with a web front-end, typically portals. However, they also develop lighter solutions with most emphasis on the front-end.

The company acts as an independent software supplier, though there are close relationships to the biggest customers. Of the 50 employees today, 35 are working as software developers. Java and J2EE are used as the main development platform. The domain of which the company develops software is mainly for the banking and finance sector, as well as for public sector. The company has run 50 development projects within the bank and finance sector the last twelve years, and about 30-40 projects within the public sector the last 15 years.

Four employees are certified RUP-mentors acting as advisors in other SWorganizations, in addition to this they also used to run training courses in RUP as part of their partnership with Rational (now IBM Rational).

During the work described in this paper the company was declared bankrupt, and then restarted with new owners, but with the same employees. The data collection (using the spreadsheet) took place before the bankruptcy; the interviews took place about six months after the company was restarted.

4 Results

4.1 Interview Round 1: Documenting the Use of RUP

The four projects investigated had a scattered use of RUP. Interviewing the project leaders we documented the projects per phase to see which process elements were used and which were not and the corresponding reasons for that. In the following we present a summary per phase for each project (named project A to D).

The business modeling discipline

Project A was about porting functionality, no new functionality was introduced, thus not needing business modeling. For project B the customer had provided a business use case that was sufficient. Project C was developing software to be integrated with other systems. The business modeling discipline was used to clarify these interfaces. Project D had a business modeling discipline although it was not performed exactly as described by RUP.

The requirements discipline

Project A used the discipline partly to specify requirements for how to join the user interface of several systems. The other three projects used the requirements discipline quite extensively.

The analysis and design discipline

The elements in the discipline were partly used for all four projects. However there was a lot of adoption to the project context.

The implementation discipline

The use of the process elements in the implementation discipline was scattered. Although all four projects used it, project A used it briefly, project C and D used it extensively.

The test discipline

Project B and D had an extensive use of the process elements in the test discipline while the two other projects did not follow RUP for testing.

The deployment discipline

Project A had deployment activities but these were not done according to RUP. Project B had at the time of the interview not got to this. The deployment in project C was done partly by the customer that had responsibility for most of the activities. Project D did utilize most elements from RUP.

The configuration and change management discipline

Project A did not follow RUP at all; this was however done using a specialized system guiding the process of configuration and change management. Project B and C did use RUP pretty extensively for this discipline. For project D the customer handled this responsibility following other procedures than described by RUP.

The project management discipline

Project A did not follow RUP except for the use of the software architect role. Project B and C did use most of the process elements from RUP. In the case of project D the customer had the project management responsibility themselves.

The environment discipline

Project A had merely no use of this discipline. Project B used most process elements. Project C used only a few but project D used several.

By mapping the use of process elements for the four projects we made a visual map documenting the use of each process element in RUP, ordered by the eight disciplines that RUP describes (see fig. 1).

4.2 Interview Round 2: Experiences with Using RUP

Five project participants with experience from several projects were interviewed to document positive and negative experiences from their use of RUP as well as any improvement suggestions. Note that these interviews are not related to the interviews in round one.

Some of the respondents had experience with more than one role and project type. The five persons interviewed had the following background:

- Respondent 1: Roles: Developer, project manager and test manager. Project types: Web applications with backend logic.
- Respondent 2: Roles: Developer, project manager and test manager (often combined). Project types: Web applications.
- Respondent 3: Roles: Project manager, requirements manager. Project types: Publication systems, banking systems.
- Respondent 4: Roles: Developer. Project types: Mostly system maintenance.
- Respondent 5: Roles: Key account manager. Project types: Secure systems.

Of the five respondents three defined their RUP knowledge as 'good', one as 'medium' and one as 'little'. Following is a summary of common statements from the interview transcriptions showing which respondents having which statements.

Business Modeling			-	-
	Α	в	С	D
loiea	r	r	1	_
Business Process Analyst Business Designer		-		-
Business Model Reviewer				-
rtefacts	Ļ	Ļ	-	Ļ
Glossary	r	r	r	<u> </u>
Supplementary Business Specification			-	⊢
Business Use Case Model	-	-		-
Business Object Model				-
Business Object Model Business Entities	-	-	-	-
Business Entities Business Use Case Realization				-
Business Workers				
Organisational Unit				
ctivities	_		_	_
Capture a Common Vocabulary				
Find Business Actors and Use Cases				
Structure the Business Use Case Model				
Detail a Business Use Case				
Find Business Workers				
Detail a Business Worker				
Detail a Business Entity				
Review the Business Use Case Model				
Review the Business Object Model				
Requirements	Δ	B	C	D
Roles			-	-
System Analyst		1		
Use Case Specifier				
User Interface Designer				
Architect				
Architect				
			-	
Glossary				
Vision				
Use-Case Model				
Requirements Management Plan				
Requirements Attributes				
Stakeholder Request				
Supplementary Specifications				
ctivities				
Analyze the Problem			1	Г
Understand Stakeholder Needs				
Define the System				
Manage the Scope of the System				
Refine the System Definition				_
Manage Changing Requirements				
Manage Changing Requirements		B		
Manage Changing Requirements Analysis & Design	A	в	C	D
Manage Changing Requirements Analysis & Design Roles	A	B	C	D
Manage Changing Requirements Analysis & Design Roles Security Engineer	A	B	c	D
Manage Changing Requirements Inalysis & Design toles Security Engineer Software Architect Reviewer	A	B	c	D
Manage Changing Requirements Inalysis & Design Roles Security Engineer Software Architect Reviewer Integrator	A	B	C	D
Manage Changing Requirements Malysis & Design boles Security Engineer Software Architect Reviewer Integrator Database Designer	A	B	c	D
Manage Changing Requirements nalysis & Design olos Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect	A	B	C	D
Manage Changing Requirements Analysis & Design blobe Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Designer	A	B	C	D
Manage Changing Requirements inahysis & Design Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Designer integrate	A	B	C	
Manage Changing Requirements Analysis & Design Otole Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Architect Document	A	B	C	
Manage Changing Requirements nalysis & Design lotes Software Architect Reviewer Integrator Database Designer Software Architect Software Designer vrofacts Software Architect Document Design Model	A	B	C	
Manage Changing Requirements inaliysis & Design Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Design Model Use-Case Realization		B	C	
Manage Changing Requirements Analysis & Design Otole Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Designer Software Architect Document Design Model Use-Case Realization Risk List		B	C	
Manage Changing Requirements nahysis & Design Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Architect Document Design Model Use-Case Realization Risk List Vision		B	C	
Manage Changing Requirements nalysis & Design Ools Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package			C	
Manage Changing Requirements Analysis & Design Otoles Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Document Design Model Use-Case Realization Risk List Vision Design Package Test Class				
Manage Changing Requirements inaliysis & Design Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package Test Interface Specification				
Manage Changing Requirements Analysis & Design Otoles Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Document Design Model Use-Case Realization Risk List Vision Design Package Test Class				
Manage Changing Requirements nalysis & Design Otos Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Document Software Architect Document Design Model Use-Case Realization Risk List Vision Design Package Test Class Test Interface Specification Review Record				
Manage Changing Requirements inanysis & Design Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Architect Document Design Model Use-Case Realization Risk List Vision Design Package Test Class Test Interface Specification Review Record Change Reguest				
Manage Changing Requirements Analysis & Design Ools Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package Test Interface Specification Review Record Change Request Capsule				
Manage Changing Requirements analysis & Design Otes Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Document Design Model User-Case Realization Risk List Vision Design Package Test Class Test Interface Specification Review Record Change Request Capsule Protocol				
Manage Changing Requirements manyois & Design totes Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Document Design Model Use-Case Realization Risk List Vision Design Package Test Interface Specification Review Record Change Request Capsule Protocol Interface				
Manage Changing Requirements nalysis & Design Otos Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package Test Interface Specification Review Record Change Request Chapsule Protocol Interface Design Sub-system				
Manage Changing Requirements Inanysis & Design Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Designer Integrator Design Model Use-Case Realization Risk List Vision Design Package Test Class Test Interface Specification Review Record Change Request Capsule Protocol Interface Design Sub-system Design Sub-system				
Manage Changing Requirements inalysis & Design Security Engineer Software Architect Reviewer Infogrator Database Designer Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package Test Inderface Specification Review Record Change Request Capsule Protocol Interface Design Sub-system Design Guidelines Supplementary Specifications				
Manage Changing Requirements natysis & Design Otes Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Document Software Architect Document Design Model Use-Case Realization Risk List Vision Design Model Design Ackage Test Class Test Interface Specification Review Record Change Request Capsule Protocol Interface Design Guidelines Supplementary Specifications Design Guidelines Supplementary Specifications Design Cases				
Manage Changing Requirements manyois & Design totes Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package Test Class Test Interface Specification Review Record Change Request Capsule Protocol Interface Design Sub-system Design Sub-system Design Class Supplementary Specifications Design Class				
Manage Changing Requirements Analysis & Design Ools Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package Test Interface Specification Review Record Change Request Caspuste Protocol Interfaces Design Guidelines Dupplementary Specifications Design Guidelines Data Model Camponet Class Design Guidelines Data Model Component Class Design Case Data Model				
Manage Changing Requirements Integrate Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Designer Integrator Design Action Design Action Design Action Design Package Test Class Test Interface Specification Review Record Change Request Capsule Protocol Interface Design Sub-system Design Sub-system Design Sub-system Design Sub-system Design Sub-system Design Sub-system Design Sub-system Design Sub-system Design Calcielines Supplementary Specifications Design Calcielines Supplementary Specifications Design Calcielines Supplementary Specifications Design Calcielines Supplementary Specifications Design Calcielines Component Class Cutivities				
Manage Changing Requirements manysis & Design Security Engineer Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Document Design Model Use-Case Realization Risk List Yision Design Package Test Interface Specification Review Record Change Request Capsule Protocol Interface Design Sub-system Design Glass Design Class Design Class				
Manage Changing Requirements analysis & Design Otes Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Document Software Architect Document Design Model Use-Case Realization Risk List Vision Design Rockage Test Interface Specification Review Record Change Request Chapsule Protocol Interface Design Guidelines Supplementary Specifications Design Guidelines Supplementary Specifications Design Guidelines Supplementary Specifications Design Guidelines Supplementary Specifications Design Guidelines Camponent Class ctivities Define a Candidate Architecture Perform Architectual Synthesis				
Manage Changing Requirements manysis & Design Security Engineer Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Document Design Model Use-Case Realization Risk List Yision Design Package Test Interface Specification Review Record Change Request Capsule Protocol Interface Design Sub-system Design Glass Design Class Design Class				
Manage Changing Requirements Analysis & Design Ools Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Software Architect Software Architect Design Model Use-Case Realization Risk List Vision Design Package Test Interface Specification Review Record Change Request Capsule Protocol Interface Design Guidelines Duplementary Specifications Design Guidelines Data Model Component Class Component Class Component Class Data Model Component Class Detated Perform Architect architecture Perform Architect Architecture Perform Architect Architecture Penform Architecture Penform Architecture Software Architecture Penform Architecture Penform Architecture Software Architecture Penform A				
Manage Changing Requirements analysis & Design Otes Security Engineer Software Architect Reviewer Integrator Database Designer Software Architect Document Software Architect Document Design Model Use-Case Realization Risk List Vision Design Rockage Test Interface Specification Review Record Change Request Chapsule Protocol Interface Design Guidelines Supplementary Specifications Design Guidelines Supplementary Specifications Design Guidelines Supplementary Specifications Design Guidelines Supplementary Specifications Design Guidelines Camponent Class ctivities Define a Candidate Architecture Perform Architectual Synthesis				

plementation	A		U
les	_		
Implementer			
Software Architect			
Integrator			
Content Editor			
Code Reviewer			
Asset Production Manager		-	
Tester	_	_	
efacts			_
Implementation Subsystem			
Software Architecture Document			
Implementation Model			
Integration Build Plan			
Component			
Test Component			
Review Record	-		
Build			
tivities		_	_
Structure the Implementation Model			
Plan the Integration			
Implement Components			
Integrate each Subsystem		1	
Integrate the system	+		
et		P	-
SL	A	в	U
les			-
Stakeholder			
Requirements Analyst			
Test Designer	T		
Integrator			
Implementer	1		1
tefacts			
Test Plan			1
			_
Test Automation Architecture			
Test Guidelines			
Test Envirnoment Configuration			
Test Script			
Test Evaluation Summary			
Test Results			
Test Data			
Test Suite			-
	-		
	_		
Test-ideas List			
Test Case			
Workload Model			
Issues List			
tivities			
Define Evaluation Mission			
Verify Test Approach			
Validate Build Stability	-	_	-
Test and Evaluate	_		_
Achieve Acceptable Mission			
Improve Test Assets			
eployment	Α	в	С
les			
Deployment manager	1	Г	
Implementor	1	1	
Technical Writer	1	t	
		1	
Graphic artist	+	-	
Course developer			
tefacts			
Deployment Plan		Ľ	
Bill of Materials			
Training Material		1	
End-user support Material		1	
		1	
Test Results	+	-	
Change Requests			
Deployment Infrastructure			
Product			
Product Artwork	1		
Deployment Unit	1	t	
tivities			
	- r	r –	_
Plan Deployment	_		
Develop Support Material		L	
Bereiep euppert material	T		
Manage Acceptance Test (Dev site)		1	
Manage Acceptance Test (Dev site) Produce Deployment Unit			-
Manage Acceptance Test (Dev site) Produce Deployment Unit			
Manage Acceptance Test (Dev site) Produce Deployment Unit Beta Test Product			
Manage Acceptance Test (Dev site) Produce Deployment Unit Beta Test Product Manage Acceptance Test (Inst site)			
Manage Acceptance Test (Dev site) Produce Deployment Unit Beta Test Product			

Configuration Manager Change Control Manager Project Manager	_	_	_	_
Change Control Manager				Γ
	-	-		Г
		-		
Integrator	_			_
Software Architect				
Tester				
tefacts				
CM Plan				
Project Respository				
	_	_	-	-
Workspace (Integration)	_			
Work Order				
Workspace (Development)				
Deployment Unit				
Configuration Audit Findings				
Project Measurements				-
	-			-
Change Requests				
tivities				
Plan Project Configuration and Change				
Create Project CM Environments				
Change and Deliver Config Items				
Manage Baselines and Releases	-	-		
	_	_		-
Monitor & Report Config Status				
Manage Change Requests				
oject Management	Α	в	С	E
les				
Business Strategist				Г
Project Reviewer	-	-		H
	⊢	-		⊢
Project Manager	1			L
Software Architect				L
Process Engineer				
Test Designer				
Tools Specialist				-
tefacts	_	_	_	_
Review Record				
Risk List				
Business Case				
Iteration Plan				-
	-	_		-
Software Development Plan				_
Develop QA Plan				
Measurement Plan				
Project Measurements				
Product Acceptance Plan				
Problem Resolution Plan	-	-		-
	_	_		⊢
Risk Management Plan				
Change Request				
Work Order				
Status Assessment				
Issues List				
	-	-		-
Iteration assessment	_	_		L
tivities				
Conceive New Project				
Plan for next Iteration				
Manage Iteration	-	-		F
		-		-
Evaluate Project Scope and Risk	⊢			L
	_			L
Develop Software Development Plan		1	L	L
				C
Develop Software Development Plan	A	В	С	
Develop Software Development Plan Close-out Project nvironment	A	В	С	
Develop Software Development Plan Close-out Project nvironment les	A	в	C	
Develop Software Development Plan Close-out Project Invironment Jes Process Engineer	A	В	C	
Develop Software Development Plan Close-out Project vironment les Process Engineer Tools Specialist	A	B	C	
Develop Software Development Plan Close-out Project virionment les Process Engineer Tools Specialist Business Process Analyst	A	B	C	
Develop Software Development Plan Close-out Project les Process Engineer Tools Specialist Business Process Analyst Test Designer	A	B	C	
Develop Software Development Plan Close-out Project virionment les Process Engineer Tools Specialist Business Process Analyst	A	B	C	
Develop Software Development Plan Close-out Project Virionment Des Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer	A	B	C	
Develop Software Development Plan Close-out Project virionment les Process Engineer Tools Specialist Business Process Analyst Test Designer Tecnical Writer Software Architect	A	B	C	
Develop Software Development Plan Close-out Project Virlonment Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Test Designer Requirement Analyst	A	B		
Develop Software Development Plan Close-out Project virionment les Process Engineer Tools Specialist Business Process Analyst Test Designer Tencinal Writer Software Architect Requirement Analyst System Administrator	A	B		
Develop Software Development Plan Close-out Project Vitromment Vitromment Dess Engineer Tools Specialist Business Process Analyst Test Designer Teschied Writer Software Architect Requirement Analyst System Administrator Process Engineer	A	B		
Develop Software Development Plan Close-out Project Vitromment Vitromment Dess Engineer Tools Specialist Business Process Analyst Test Designer Teschied Writer Software Architect Requirement Analyst System Administrator Process Engineer	A	B		
Develop Software Development Plan Close-out Project Virionment Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Test Designer Requirement Analyst System Administrator Process Engineer tefacts		B		
Develop Software Development Plan Ciose-out Project Verforment Ies Process Engineer Tools Specialist Business Process Analyst Test Designer Teorical Writer Software Architect Requirement Analyst System Administrator Process Engineer Tefacts Development-organisation Assessment		B		
Develop Software Development Plan Close-out Project Virlomment Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Testineal Writer Software Architect Requirement Analyst System Administrator Process Engineer Terocess Engineer Terocess Engineer Terjoect-specific Templates		B		
Develop Software Development Plan Close-out Project Virionment Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Test Designer Requirement Analyst System Architect Requirement Analyst System Administrator Process Engineer tefacts Development-organisation Assessment Project-specific Templates Development Case				
Develop Software Development Plan Close-out Project Virlomment Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Testineal Writer Software Architect Requirement Analyst System Administrator Process Engineer Terocess Engineer Terocess Engineer Terjoect-specific Templates				
Develop Software Development Plan Close-out Project Virformtent Process Engineer Tools Specialist Business Process Analyst Test Designer Testi Designer Testi Designer Software Architect Requirement Analyst System Administrator Process Engineer Effacts Development-organisation Assessment Project-specific Templates Development Case Tools		B		
Develop Software Development Plan Close-out Project Vivironment Ies Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Requirement Analyst System Administrator Process Engineer detat Development-organisation Assessment Project-specific Templates Development Case Tools Business Modelling Guidelines				
Develop Software Development Plan Close-out Project Vivironment Ies Procass Engineer Tools Specialist Business Process Analyst Test Designer Tecnical Writer Software Architect Requirement Analyst System Administrator Process Engineer efacts Development-organisation Assessment Project-specific Templates Development Case Tools Business Modelling Guidelines				
Develop Software Development Plan Ciose-out Project Vitronment Des Expectalist Business Process Analyst Tools Specialist Business Process Analyst Testi Designer Testi Designer Testi Designer Software Architect Requirement Analyst System Administrator Process Engineer Terocess Engineer Terocess Engineer Terocess Engineer Development Case Tools Business Modelling Guidelines Design Guidelines				
Develop Software Development Plan Close-out Project Virionment Jes Process Engineer Tools Specialist Business Process Analyst Test Designer Tecnical Writer Software Architect Requirement Analyst System Administrator Process Engineer tefacts Development-organisation Assessment Project-specific Templates Development Case Tools Business Modelling Guidelines				
Develop Software Development Plan Ciose-out Project Vitronment Des Expectalist Business Process Analyst Tools Specialist Business Process Analyst Testi Designer Testi Designer Testi Designer Software Architect Requirement Analyst System Administrator Process Engineer Terocess Engineer Terocess Engineer Terocess Engineer Development Case Tools Business Modelling Guidelines Design Guidelines				
Develop Software Development Plan Close-out Project Virformtent Vironment Des Process Engineer Tools Specialist Business Process Analyst Test Designer Testi Designer Testi Designer Software Architect Requirement Analyst System Administrator Process Engineer Effacts Development-organisation Assessment Project-specific Templates Development Case Tools Business Modellings Business Modellines Manual Styleguide Programming Gudellines				
Develop Software Development Plan Ciose-out Project Virionment Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Test Designer Requirement Analyst System Administrator Process Engineer Project-specific Templates Development-organisation Assessment Project-specific Templates Development Case Tools Business Modelling Guidelines Design Guidelines Manual Skyeuide Programming Guidelines Test Guidelines				
Develop Software Development Plan Ciose-out Project Virfornment Jes Process Engineer Tools Specialist Businese Process Analyst Test Designer Teorical Writer Software Architect Requirement Analyst System Administrator Process Engineer Verfacts Development-organisation Assessment Project-specific Templates Development Case Tools Business Modelling Guidelines Design Guidelines Manual Styleguide Programming Guidelines Test Guidelines Test Guidelines				
Develop Software Development Plan Ciose-out Project Virionment Process Engineer Tools Specialist Business Process Analyst Test Designer Test Designer Test Designer Requirement Analyst System Administrator Process Engineer Process Engineer Project-specific Templates Development-organisation Assessment Project-specific Templates Development Case Tools Business Modelling Guidelines Design Guidelines Manual Skyeuide Programming Guidelines Test Guidelines				

Table 1. Interviews summary

		Interview respondents						
+/-	Nodes from interview coding	1	2	3	4	5		
Positive experience	The RUP training was good				Х			
	Roles defined by RUP	х	х					
	Important to have a supporting process				Х			
	Used inception and elaboration [with success]				х			
	SW maintenance projects uses RUPs guidelines for transition between phases/milestones	х						
	Templates and role definitions are good checklists	х						
	Want to be better at using RUP	Х	х	х	х	Х		
	Reasonable division in phases and iterations	х						
Negative experience	To extensive for small projects	х	х	х	Х			
	Too document-driven				х			
	Too much focus on just the development					х		
	Missing roles for customer contact prior to and past the development project					х		
	Requires good knowledge [of RUP]					х		
	Missing a common standard of use		х	х		л		
	Does not fit a software maintenance processes	х	A	A				
	Miss adaptation to extreme programming	x						
	We do not evaluate our use of RUP	х						
	Continues with old practice				Х			
	We have not changed our practice after RUP was		Х					
	introduced							
	I have no progress [as a software professional]	х		х				
	Missing follow-up during projects		Х	х				
	Hard to understand RUP		х					

Note that this list is a collection of all statements found relevant to the use of RUP. Some are clear and can be generalized; others are specific to a single project. The definition of the nodes is based on an interpretation of the interviews (due to the constant comparison method).

Besides this overview of experience using RUP, the respondents also had improvement suggestions:

- RUP should be used in a regular manner through the whole project (avoiding deep focus in only parts of the project)
- Projects must be guided in the use of RUP
- Web-projects need more specialized support than RUP can offer
- Establish a project manager forum (for learning and experience exchange)
- Avoid the use of RUP in the case of software maintenance
- Offer support in using and adapting RUP

5 Discussion and Conclusion

Offering RUP out-of-the-box leaves all the responsibility of tuning RUP to each individual project. This may cost both time and resources. Good knowledge on RUP is also needed. As the results from interview round one show, the use of RUP is scattered and deviates partly from the RUP guidelines. Project participants seems to end up using some RUP elements mixed with old practice, not as a consequence of deliberate decisions but as a consequence of low knowledge of RUP and how to adapt it.

The phases (and disciplines) of RUP covers the complete lifecycle of a software development project. However, in a real context, as the interviews show, the customer often has done some part of the job initially following an internal process. This may affect the use of RUP later on in the project.

Looking at the results from interview round two we see that most respondents support the idea (in general) of having a guiding process that includes role descriptions and regulates the work in disciplines, phases and iterations. However, <u>all</u> of the respondents feel that they need to and want to be better at using RUP. The reason for this may be that RUP is extremely comprehensive and that the task of fitting this framework to a project may be overwhelming. We also see that four of the respondents find RUP too comprehensive for small projects. This indicates a definitively need for tailoring of RUP in advance of use in projects. Two respondents also miss a common practice for the use of RUP, also indicating the need of a general tailoring of RUP.

In general, the interview results show that providing RUP just in the form of the full documentation (in this case RUP Online – right out of the box) have negative effects, at least not as good effects as one would believe in advance. It is perceived as too comprehensive and the users have problems finding the parts that would benefit their project. The consequence may be avoidance of use or even worse, wrong use. Two respondents claim that they have not changed their practice of developing software after RUP became available. This resembles with known acceptance models[5]; the methodology must be perceived as <u>useful</u> (will using RUP enhance the job performance?) and it must be perceived as <u>easy to use</u> (will using RUP require low effort?).

Besides doing a thorough adaptation in advance to increase usefulness and ease of use, projects also need practical guidance throughout the project; two respondents miss this type of support, this is also on the list of improvement suggestions. Introducing guidance and mentoring would both improve the degree of use and the effect of use of RUP as well as it would serve as a experience transfer mechanism.

Conclusion: The basic learning from this case study is that a methodology or framework (such as RUP) can not be provided "as is" without experiencing low/wrong use. The users of the methodology need to keep their focus on doing their job (developing software), not struggling to understand the theory. This is actually what the RUP documentation says, but that many unfortunately forget. Introducing a methodology such as RUP is an investment beyond the license fee. In this case the outcome could have been better if the introduction of RUP was carefully managed and not left as an autonomous effort in each project.

A comment: The learning from this study made the company decide to initiate a RUP adaptation process to provide their employees with better process support. This work is described in [6].

6 Further Research

The research reported in this paper, and also in other papers has put emphasis on the challenges in implementing and tailoring RUP for use in an organization [6-8]. Implementing a process framework like RUP can be looked upon as implementation of a new technology in an organization. It would therefore be of interest to study such implementations in spite of technology acceptance models [5] and investigate the success factors of tailoring and introduction of methodologies.

Acknowledgements

The authors would like to thank the participants from the case company. We would also like to thank the SPIKE-project (funded by the Research Council of Norway) for funding and support.

References

- 1. Krutchen, P., *The Rational Unified Process: An Introduction*. 2nd ed. 2000: Addison-Wesley. 298.
- Yin, R.K., Case Study Research Design and Methods. Applied Social Research Methods Series, ed. D.S. Foster. 1994: SAGE Publications.
- 3. Seaman, C., *Qualitative methods in empirical studies in software engineering*. IEEE Transactions on Software Engineering, 1999. **25**(4): p. 557-572.
- 4. Avison, D., et al., Action Research. Communications of the ACM, 1999. 42(1): p. 94-97.
- Riemenschneider, C.K.H., et al., *Explaining software developer acceptance of methodologies: a comparison of five theoretical models*. Software Engineering, IEEE Transactions on, 2002. 28(12): p. 1135-1145.
- 6. Westerheim, H. and Hanssen, G.K., *The Introduction and Use of a Tailored Unified A Case Study*. in *31st EUROMICRO CONFERENCE on Software Engineering and Advanced Applications (SEAA)*. 2005. Porto, Portugal: IEEE.
- Bergström, S., Råberg, L., Adopting the Rational Unified Process. 2004, Addison-Wesley. p. 165-182.
- 8. Hanssen, G.K., et al., *Tailoring RUP to a defined project type: A case study.* in 6th International Conference on Product Focused Software Process Improvement, PROFES. 2005. Oulo, Finland: Springer.