Systematic Analysis of the Incremental Process as a Base for Comparison with the Agile Process

Seda Gunes Yilmaz
Siemens Enterprise Communications
METU-Tech
06531 Ankara, Turkey
seda.gunes@siemens-enterprise.com

Ayca Tarhan
Hacettepe University
Computer Engineering Department
06532 Ankara, Turkey
atarhan@cs.hacettepe.edu.tr

Background: The number of studies that quantitatively reveal the effect of using Agile models on development performance is scarce.
Aim: In this paper, we explain the analysis of a plan-driven, Incremental Process within Siemens EC, carried out to understand its performance as a base for the comparison with the Agile Process.
Method: We introduce a method to systematically analyze a software development process to understand its performance quantitatively, and explain the application of the method on the System Test phase of the Incremental Process.
Results: The utilization of the analysis method practically contributed to the implementation which was completed in 8 person-days. An analysis of derived measure values indicated that there was no outlier value in the interval of mean ± 3 standard deviation for the measures.
Conclusions: We expect that the method will be more beneficial in capturing the context of the Agile Process and its measures.

Keywords-*quantitative analysis, software development models, software measurement, process performance, product quality, GQM*

I. INTRODUCTION

Software development activities such as requirements analysis, design, implementation, testing, and maintenance are carried out within a software development life-cycle in organizations (Agarwal, et.al., 2010). Every software project has a life-cycle and software development life-cycles should be adapted from software development models that have been implemented for years and proved as successful. However, many software projects adapting different development models for their life-cycles fail to achieve their targets. In accordance to an analysis performed by The Standish Group in 2009, only 32% of software projects were reported as successful in comparison to reported 44% as challenged (late, over budget and/or with less than the required features and functions) and 24% as failed (cancelled prior to completion or delivered and never used).

Low success rates of software projects motivated inquiry of traditional software development models in 1990s and in the following years, Agile software development models were proposed as alternatives to the traditional models. The Agile models were defined to value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan (Abrahamson, et.al., 2002). Although Agile software development models have been widely used as a basis for the software project life-cycle since 1990s, the number of studies that quantitatively reveal the effect of using these models on development performance is scarce (Layman, et.al., 2004; Huo, et.al., 2004; Khramov, 2006; Li, et.al., 2010). One important reason for this is the difficulties in the measurement of the Agile models. Another reason is that the performance of the development models previously used in organizations have not been measured to constitute a base for comparison. Software process measurement and analysis are not easy to accomplish. It requires knowledge on the concepts of measurement, process management, and statistics as well as on their practical applications. More specifically, it requires a series of tasks to be carried out such as identifying the purpose of the analysis, capturing process context, identifying process components, ensuring consistency of process executions, gathering process data, selecting process measures, determining analysis methods, conducting the analysis, and interpreting analysis results. When comparing the performances of two or more development processes, reconciling the scopes of the processes as a basis for the comparison is another task to accomplish. Aside from these challenges, there are only a few systematic and practical methods (Van Solingen, et.al., 2002; ISO/IEC, 2002) that can be utilized as a guide while measuring the performance of a software development process. Nevertheless, quantitatively revealing the effect of using development models on development performance and product quality, is among the major needs of software organizations and software community.

In this paper, with a motivation to address the need stated above, we introduce a method to systematically analyze a software development process in order to understand its performance quantitatively. The method includes practical guidelines and assets for modeling of a development process to understand its attributes, investigating consistency of process executions, mapping of process scope to a process reference model, and evaluating the usability of process measures for statistical analyses. The Goal-Question-Metric Framework ("GQM") (Van Solingen, et.al., 2002) is taken as a guideline to identify analysis goals and required measures, and complemented by a bottom up approach in order to reconcile required measures and available process measures. The assets of a predefined assessment approach (Tarhan and Demirors, 2011) are utilized in order to verify consistency of process executions and evaluate measure characteristics prior to

quantitative analysis. In the paper, we also explain the implementation of this method on the System Test phase of a plan-driven, Incremental Process applied in many projects within Siemens EC in order to create a base for the comparison with the Agile Process. The Incremental Process is an adaption of the Waterfall Model (Agarwal, et.al., 2010) by the organization. Neither the Incremental Process nor a measurement process has been defined and applied organization-wide. The projects have been following the Incremental Process on a plan-driven basis. Although the main purpose of the analysis was to compare the two development processes, the analysis of the Incremental Process provided many valuable results in its own.

The remaining of the paper is organized as follows: Section two depicts the flow of the analysis method, section three explains the steps of the analysis method and their implementation for the Incremental Process, and finally section four provides the conclusions.

II. THE ANALYSIS METHOD

The analysis method has been developed with the purpose of providing a systematic, repeatable roadmap to perform quantitative analysis on a software development process. Such an analysis can be carried out to understand the performance of a single development process or a group of development processes for the purpose of comparison. The method has nine steps as shown in Figure 1.

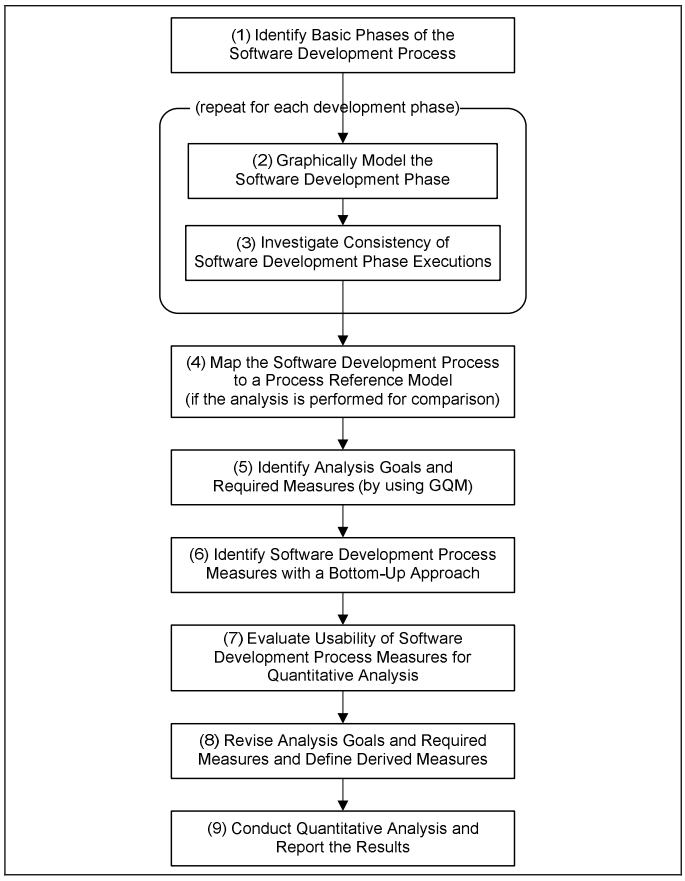

Figure 1. The Analysis Method

Due to space limitations, the details regarding the steps of the analysis method are explained in the following section together with the implementation for the System Test phase of the Incremental Process.

III. IMPLEMENTATION

The utilization of the analysis method has been aimed at comparing the performance and the product quality of the Incremental Process applied in many projects within Siemens EC with those of the Agile Process. The product that is subject to this study had been developed with the Incremental Process as a part of a product line family, and had a great interest and customer satisfaction in the inner and outer markets at the beginning. However, as the requirements of revision and extension were emerged in time, the complexity of the software was increased. Major defects that were introduced by and could not be caught at each increment were later reported by the customers. This led to the loss of the customers while increasing the development costs. Due to these problems and the marketing strategy, the organization decided the development of the product with an extended scope and by applying an Agile Process. The Agile Process is a combination of the Unified Software Development Process (Jacobson, et.al., 1999), Extreme Programming and SCRUM (Abrahamson, et.al., 2002), adapted to the specific needs of the organization.

A retrospective analysis of the Incremental Process was completed for the development of completely new 20 features of the product. The analysis results will be used as a base to compare the performances of the Incremental Process and the Agile Process. The project team that applied the Incremental Process included 9 people (a project manager, a technical project lead, 4 software developers, a system tester, a customer support staff, and a technical services staff). The common number of people involved in both processes is 7. The development environment and the language used to incorporate the features into the products in both processes are similar.

The first step is identifying basic phases of the development process. These may include requirements analysis, architectural design, detailed design, implementation, system test, and etc. This is a preparation step to understand the context of the software development. For the Incremental Process; the phases of Development (design and coding), System Test, and Customer Use were identified for the quantitative analysis.

The steps 2 and 3 are performed for each software development phase identified in step 1. In the step 2, the flow of a software development phase is modeled graphically to understand its basic attributes such as inputs, activities, outputs, roles, and tools. If there is an organization-wide definition of the software development process, it can be used as an input to this step. Otherwise, project plans and interviews with project managers and process performers can be used to capture phase details. We recommend to use Event-Driven Process Chain (EPC) notation to graphically model a phase, but any other notation to capture process flow and phase attributes specified above may serve the same purpose. In the model, each activity of the phase is numbered for a later reference in the step 4. The EPC model of the System Test phase of the Incremental Process is shown in Figure 2.

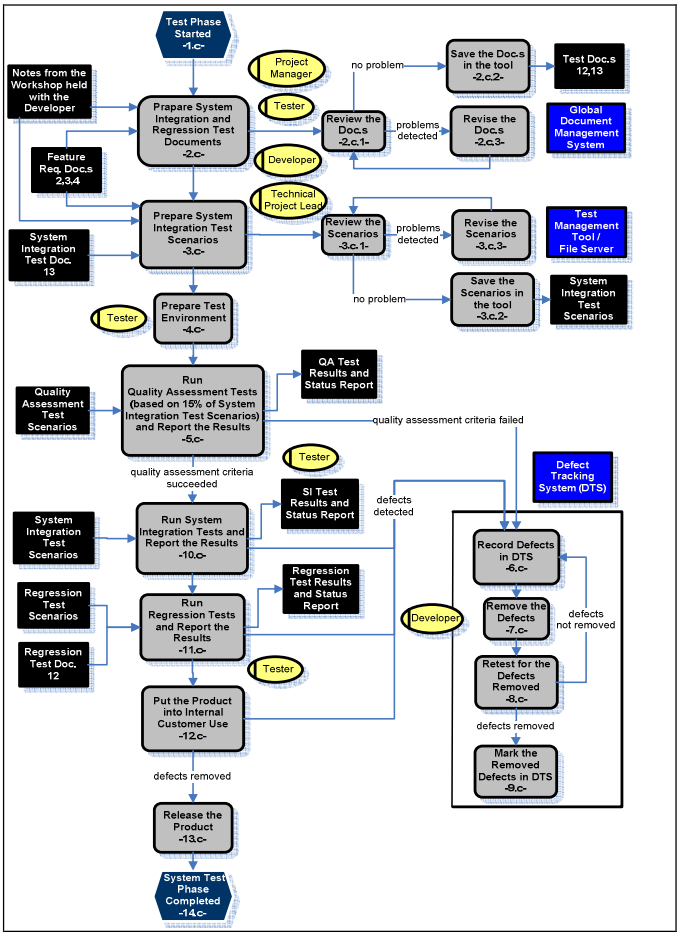

Figure 2. The EPC Model of the System Test Phase

		Features																			
	SYSTEM TEST PHASE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Inputs																				
	1.1 Feature Requirement Documents 2,3,4	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	1.2 Notes from the Workshop held with the Developer	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
2	Outputs																				
	2.1 System Integration (13) and Regression (12) Test Documents																				
	2.2 Test Results and Status Report	o	o		o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
3	Activities																				
	3.1 Prapare System Integration and Regression Test Documents																				
	3.2 Review the System Integration and Regression Test Documents																				
	3.3 Revise the System Integration and Regression Test Documents																				
	3.4 Prapare System Integration Test Scenarios	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.5 Review the System Integration Test Scenarios	o	o	o	o	o	o	o				o	o	o	o			o	o	o	o
	3.6 Revise the System Integration Test Scenarios	o	o	o	o	o	o	o				o	o	o	o			o	o	o	o
	3.7 Prepare Test Environment	o	o	o	o	o	o	o		o	o	o	o	o	o	o	o	o	o	o	o
	3.8 Run Quality Assessment Tests	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.9 Release the Quality Assessment Test Results and Status Report	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.10 Record the Quality Assessment Test Defects	o	o	o	o	o	o	o	o	o	o	o	o	o	o		o	o	o	o	o
	3.11 Remove the Quality Assessment Test Defects	o	o	o	o	o	o	o	o	o	o	o	o	o	o		o	o	o	o	o
	3.12 Retest the Quality Assessment Test Defects	o	o	o	o	o	o	o	o	o	o	o	o	o	o		o	o	o	o	o
	3.13 Run the System Integration Test Scenarios	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.14 Release the System Integration Test Results and Status Report	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.15 Record the System Integration Test Defects	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.16 Remove the System Integration Test Defects	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.17 Retest the System Integration Test Defects	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.18 Run the Regression Test Scenarios	o	o	o	o			o								o		o		o	
	3.19 Release the Regression Test Results and Status Report	o	o	o	o			o								o		o		o	
	3.20 Record the Regression Test Defects																				
	3.21 Remove the Regression Test Defects																				
	3.22 Retest the Regression Test Defects																				
	3.23 Put the Product into Internal Customer Use	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	3.24 Record the Defects found in Internal Customer Use	o	o	o	o	o	o	o	o		o	o	o	o	o	o	o	o	o	o	o
	3.25 Remove the Defects found in Internal Customer Use	o	o	o	o	o	o	o	o		o	o	o	o	o	o	o	o	o	o	o
	3.26 Retest the Defects found in Internal Customer Use	o	o	o	o	o	o	o	o		o	o	o	o	o	o	o	o	o	o	o
4	Roles																				
	4.1 Project Manager	o	o	o	o	o	o	o	o	o	o	o	o	o	o			o	o	o	o
	4.2 Technical Project Lead	o	o	o	o	o	o	o	o	o	o	o	o	o	o			o	o	o	o
	4.3 Developer (Feature Owner)	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	4.4 Tester	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
5	Tools																				
	5.1 Defect Tracking System	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	5.2 Global Document Management System	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
	5.3 Test Management Tool / File Server	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o

Figure 3. The PSM of the System Test Phase

In the step 3, the consistency of software development phase executions are verified against phase attributes captured in the previous step. If the attributes are similar between phase executions, it means the phase is consistently performed. This provides the confidence that phase measurement data come from a homogeneous source and can be utilized for quantitative analysis. A Process Similarity Matrix (PSM) (Tarhan and Demirors, 2011) is utilized to perform the verification. In the matrix, the values of phase attributes are shown in the rows and phase execution numbers are shown in the columns. By going over the executions, the values of attributes are questioned and marked if applicable. Process quality records and interviews with process performers are used as information sources. The PSM completed for the System Test phase of the Incremental Process is given in Figure 3. It was observed from the matrix that although a number activities were not performed as expected by the process model depicted in Figure 2, the System Test phase was carried out consistently between its executions in general and therefore could go into the quantitative analysis.

The step 4 is performed to understand the scope of a software development process with respect to a process reference model like CMMI for Development (CMU/SEI, 2010) or ISO/IEC 12207 (ISO/IEC, 2008), and is recommended if performances of two or more software development processes will be compared. This is a quick mapping of the activities of a software development process to the practices of a process reference model rather than being a formal assessment of the development process. The mapping requires primary knowledge on the practices of the process reference models and on the activities of the development process which are obtained in the steps 2 and 3. First, the processes of the reference model are selected as a base for the mapping by considering the purpose of the analysis and the context of the software development process. Then, the activities of the software development process are mapped to the practices of the selected processes by using the activity numbers given in the step 2. The degree of conformance is determined for each process practice in the following four values in the ordinal scale: Full, Large, Partial, None. The degree of the conformance to a process is determined by taking the median of the values for the process practices. At the end of this step, the degree of conformance of a software development process to the selected processes of a selected process reference model is identified. Understanding the scopes of software development processes with respect to the same process reference model prior to comparison is useful in interpreting analysis results and comparing process performances. Regarding the analysis of the Incremental Process, CMMI for Development was selected as the process reference model since the organization has a target to achieve CMMI maturity level 3. The selected process areas of the model as a base for the mapping included Requirements Management, Product and Process Quality Assurance, Requirements Development, Technical Solution, Product Integration, Verification, and Validation. As a results of the mapping, the conformances of the Incremental Process activities to the practices of Requirements Management and Technical Solution process areas were detected as “Partial”. The conformances of the Incremental Process activities to the practices of the remaining selected process areas were detected as “Large”.

TABLE I. THE GQAL-QUESTION-METRIC (GQM) TABLE FOR THE INCREMENTAL PROCESS

Goal	G1	Obtain an Understanding of Product Quality	Derived Measure Formulas
Question	Q1	What is the defect density in Implementation Phase?	
Measure	D1	Defect density per test scenario in Implementation Phase	B1 / B4
Measure	D2	Defect density in Implementation Phase	B1 / B19
Question	Q2	What is the defect density in System Test Phase?	
Measure	D3	Defect density per test scenario in System Test Phase	B2 / B5
Measure	D4	Defect density in System Test Phase	B2 / B19
Question	Q3	What is the defect density in Customer Use Phase?	
Measure	D5	Defect density in Customer Use Phase	B3 / B19
Question	Q4	What is the overall defect density?	
Measure	D6	Overall defect density	(B1+B2+B3) / B19
Goal	**G2**	**Obtain an Understanding of System Test Phase Performance**	
Question	Q5	What is the V&V effectiveness in System Test phase?	
Measure	D7	System test execution V&V effectiveness	B2 / (B1+B2+B3)
Question	Q6	What is the test effectiveness in System Test phase?	
Measure	D8	System test effectiveness	B2 / (B10+B11)
Question	Q7	What is the test speed in System Test phase?	
Measure	D9	System test speed	B5 / (B10+B11)
Question	Q8	What is effort estimation capability in System Test phase?	
Measure	D10	System test effort estimation capability	(B10+B11+B12+B13) / B18
Goal	**G3**	**Obtain an Understanding of Development Phase Performance**	
Question	Q9	What is the productivity in Development phase?	
Measure	D11	Software productivity	B19 / (B6+B7+B8+B9+B14+B15+B16)
Question	Q10	What is the effort estimation capability in Development phase?	
Measure	D12	Development effort estimation capability	(B6+B7+B8+B9+B14+B15+B16) / B17
Question	Q11	What is the test execution V&V effectiveness in Development phase?	
Measure	D13	Development test execution V&V effectiveness	B1 / (B1+B2+B3)
Question	Q12	What is the test effectiveness in Development phase?	
Measure	D14	Development test effectiveness	B1 / (B8+B9)
Question	Q13	What is the test speed in Development phase?	
Measure	D15	Development test speed	B4 / (B8+B9)
Goal	**G4**	**Obtain an Understanding of Incremental Process Performance**	
Question	Q14	What is the test execution V&V effectiveness of the Incremental Process?	
Measure	D16	Incremental Process test execution V&V effectiveness	(B1+B2) / (B1+B2+B3)
Question	Q15	What is the defect removal effort ratio in the Incremental Process?	
Measure	D17	Incremental Process defect removal effort ratio	(B14+B15+B16) / ((B6+B7+B8+B9)+(B10+B11+B12+B13)+(B14+B15+B16))
Question	Q16	What is the productivity of the Incremental Process?	
Measure	D18	Incremental Process productivity	B19 / ((B6+B7+B8+B9)+(B10+B11+B12+B13)+(B14+B15+B16))
Question	Q17	What is the effort estimation capability in the Incremental Process?	
Measure	D19	Incremental Process effort estimation capability	((B6+B7+B8+B9)+(B10+B11+B12+B13)+ (B14+B15+B16)) / (B17+B18)

TABLE II. BASE MEASURES USABLE FOR THE INCREMENTAL PROCESS

Category	No	Base Measure	Data Coll. / Ver. Mechanisms
Number of Defects	B1	Number of defects detected in Implementation	Tool / Form
	B2	Number of defects detected in System Test	Tool / Form
	B3	Number of defects detected in Customer Use	Tool
Number of Test Scenarios	B4	Number of test scenarios run by the developer	Form / Interview
	B5	Number of test scenarios run by the tester	Tool / Form
Design and Implemen-tation Effort	B6	Design effort	Form / Interview
	B7	Coding effort	Form / Interview
	B8	Test design effort of the developer	Form / Interview
	B9	Test execution effort of the developer	Form / Interview
System Test Effort	B10	Test design effort of the tester	Form / Interview
	B11	Test execution effort of the tester	Form / Interview
	B12	Defect reporting effort	Form / Interview
	B13	Retest effort for detected defects	Form / Interview
Defect Removal Effort	B14	Removal effort of defects detected in Implementation	Tool / Interview
	B15	Removal effort of defects detected in System Test	Tool / Interview
	B16	Removal effort of defects detected in Customer Use	Tool / Interview
Planned Effort	B17	Planned effort for Design and Implementation	Tool / Interview
	B18	Planned effort for System Test	Tool / Interview
Software Length	B19	Source Lines of Code	Tool / Interview

In the step 5, the analysis goals and required measures are identified by using the GQM Framework (Van Solingen, et.al., 2002). This recommends to identify measurement goals (at the conceptual level) and to follow a top-down approach for deriving questions (at the operational level) to evaluate the achievement of the goals and defining measures (at the quantitative level) to answer the questions. The output of this step is a draft GQM table including analysis goals, related questions, and required measures. Regarding the analysis of the Incremental Process, the draft GQM table is not provided here but its finalized version (as an output of the step 8) is shown in Table 1. The base measures in the formulas of the derived measures can be followed from Table 2 with their categories, names, and data collection and verification mechanisms.

The steps 6, 7, and 8 are recommended to complement the top-down approach of the GQM Framework. In the step 6, base measures of the software development process are identified by following a bottom-up approach. Process data available on the forms and tools are investigated for each process phase identified in the step 1, and the base measures that might have a relation to the goals identified in the step 5 are determined as candidates for quantitative analysis. Regarding the analysis of the Incremental Process, process data were collected from tools and forms and via interviews held with the process performers.

In the step 7, process base measures identified in the previous step are evaluated for their usability in quantitative analysis. A Measure Usability Questionnaire (Tarhan and Demirors, 2011) is utilized for the evaluation. This includes questions (and expected answers) to investigate measure and data characteristics in the following four usability attributes: Measure Identity, Data Existence, Data Verifiability, and Data Dependability. According to the answers of the questions, the satisfaction of each usability attribute is rated in the following four values in the ordinal scale: Fully, Largely, Partially, and None. Overall usability of a measure is determined based on the rates of these attributes. The output of this step is a list of process measures that are usable in the quantitative analysis. The list of usable measures for the Incremental Process is given in Table 2. The base measures of planned development start date, planned system test start date, planned development finish date, actual development start date, actual system test start

date, actual development finish date, and actual system test finish date were evaluated as not usable for the quantitative analysis of the Incremental Process.

Step 8 is performed to revise the draft GQM table created in step 5 and to define derived measures that will be used in answering the questions in the table. The goals, questions and measures identified in the draft GQM table are the planned ones rather than being real, and might require update considering the knowledge obtained in the steps 6 and 7. The purpose is to reconcile the required measures identified by using the GQM Framework and the measures identified as usable for quantitative analysis at the end of step 7. The draft GQM table is revised and finalized. In addition, derived measures, which will be used to answer the questions in the GQM table, are defined with their names and formulas. It is not definite in the GQM framework how a question will be answered by using the base measures (Van Solingen, et.al., 2002), and the formulas of the derived measures indicate how they are obtained from the base measures (Table 1).

Finally, in step 9, quantitative analysis of the software development process is carried out in accordance to the GQM table and the definitions of the derived measures. The values of the derived measures are calculated from the values of the base measures according to the formulas, the questions are answered by using the values of the derived measures, and the achievement of the goals are evaluated based on the answers to the questions. The results of the quantitative analysis are interpreted by using the knowledge obtained in the steps 3 and 4 about the scope and context of the software development process. The results and findings of the quantitative analysis are documented and reported to the managers. Regarding the analysis of the Incremental process, the values of the derived measures (multiplied by a constant to preserve the privacy of the company data) are given in Table 3.

TABLE III. DERIVED MEASURE VALUES OF THE INCREMENTAL PROCESS

Derived Measure and Unit	Incremental Process Phase			
	Design and Implementation	System Test	Customer Use	Overall
Test defect density (defect# / TS#)	0.184	0.255	-	-
Test defect density (defect# / LOC)	0.011	0.056	0.016	0.066
Test execution V&V effectiveness	0.501	0.255	-	1.050
Test effectiveness (defect# / hour)	0.111	0.126	-	-
Test speed (TS# / hour)	0.852	0.825	-	-
Effort estimation capability	1.552	1.102	-	1.468
Productivity (LOC / hour)	1.686	-	-	1.011
Defect removal effort ratio	-	-	-	0.251

Since the goals in the GQM table were defined for the purpose of the comparison, the correlation was utilized as a supportive statistical method to evaluate the performance of the Incremental Process by itself. Three important relations are reported here. First, there is a negative strong relationship (with a factor of -0.85) between the Incremental Process test execution V&V effectiveness (D16) and the Defect density in Customer Use phase (D5). This means the more effective the V&V activities in the process, the lower the number of defects detected by the customer. Second, there is a negative strong relationship (with a factor of -0.85) between the Development test execution V&V effectiveness (D13) and the Defect density in System Test phase (D4). This means the more effective the V&V activities in the development, the lower the number of defects detected by the system test. And third, there is a positive relationship (with a factor of +0.60) between the Defect density in Customer Use phase (D5) and the Incremental Process defect removal effort ratio (D17). This means the later the defects are being caught in the process, the higher the cost rate for removing the defects.

IV. CONCLUSIONS

The analysis method provided a systematic guideline for retrospective quantitative analysis of the Incremental Process. Evaluating the usability of the process and its measures for the quantitative analysis and defining the goals and measures in accordance to the GQM framework practically contributed to the implementation which was completed in 8 person-days. An analysis of derived measure values indicated that there was no outlier value in the interval of mean ± 3 standard deviation for neither of the derived measures, which supported the findings of process consistency and measure usability evaluations. A constraint related to this study might be the inclusion of a single project in the analysis, but more features can be added into the analysis in time by using the analysis method. The following steps of our study are the quantitative analysis of the Agile Process in the same way and the comparison of the performance and the product quality of the two processes. We expect that the method will be more beneficial in capturing the context of the Agile Process and its measures.

REFERENCES

[1] Agarwal, B.B., Tayal, S.P., and Gupta, M., 2010. *Software Engineering and Testing*. Infinity Science Press.

[2] Abrahamson, P., Salo, O., Ronkainen, J., and Warsta, J., 2002. Agile software development methods: review and analysis. VTT.

[3] CMU/SEI-CMMI Product Team, 2010. CMMI for Development V1.3. CMU/SEI-2010-TR-033.

[4] Huo, M., Verner, J., Zhu, L., and Babar, M.A., 2004. Software quality and agile methods. In proceedings of the 28th Annual International Computer Software and Applications Conference, vol.01, pp. 520-525.

[5] ISO/IEC, 2002. ISO/IEC 15939: Software measurement process.

[6] ISO/IEC, 2008. ISO/IEC 12207: Systems and software engineering-Software life cycle processes.

[7] Jacobson, I., Booch, G., and Rumbaugh, J., 1999. *The Unified Software Development Process*. Addison-Wesley Object Tech. Series.

[8] Khramov, Y., 2006. The cost of code quality. In proceedings of the AGILE Conference, pp. 119-125.

[9] Layman, L., Williams, L., and Cunningham, L., 2004. Exploring extreme programming in context: an industrial case study. In proceedings of the Agile Development Conference, pp. 32-41.

[10] Li, J., Moe, N.M., and Dyba, T., 2010. Transition from a plan-driven process to scrum-a longitudinal case study on software quality. In proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement.

[11] Tarhan, A., and Demirors, O., 2011. Investigating the effect of variations in test development process: a case from a safety-critical system. *Software Quality Journal*, Doi: 10.1007/s11219-011-9129-8.

[12] Van Solingen, R., Basili, V., Caldiera, G., and Rombach, D.H., 2002. Goal Question Metric (GQM) approach. *Encyclopedia of Software Engineering*, online vers.@Wiley Interscience.