How to build a good practice software project portfolio?

Huijgens, Hennie; Van Solingen, Rini; Van Deursen, Arie

DOI
10.1145/2591062.2591187

Publication date
2014

Document Version
Accepted author manuscript

Published in
36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings

Delft University of Technology
Software Engineering Research Group
Technical Report Series

How To Build a Good Practice Software Project Portfolio?

Hennie Huijgens, Rini van Solingen, and Arie van Deursen

Report TUD-SERG-2013-019

TU Delft

SERG

TUD-SERG-2013-019

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
http://www.se.ewi.tudelft.nl/techreports/

For more information about the Software Engineering Research Group:
http://www.se.ewi.tudelft.nl/

How To Build a Good Practice Software Project Portfolio?

Hennie Huijgens
Delft University of Technology and Goverdson, Delft, The Netherlands
h.k.m.huijgens@tudelft.nl

Rini van Solingen
Delft University of Technology and Prowareness, Delft, The Netherlands
d.m.vansolingen@tudelft.nl

Arie van Deursen
Delft University of Technology
Delft, The Netherlands
arie.vandeursen@tudelft.nl

ABSTRACT

Context: What can we learn from historic data that is collected in three software companies that on a daily basis had to cope with highly complex project portfolios?

Objective: In this paper we analyze a large dataset, containing 352 finalized software engineering projects, with the goal to discover what factors affect software project performance, and what actions can be taken to increase project performance when building a software project portfolio.

Method: The software projects were classified in four quadrants of a Cost/Duration matrix: analysis was performed on factors that were strongly related to two of those quadrants, Good Practices and Bad Practices. A ranking was performed on the factors based on statistical significance.

Results: The paper results in an inventory of 'what factors should be embraced when building a project portfolio?' (Success Factors), and 'what factors should be avoided when doing so?' (Failure Factors).

Conclusion: The major contribution of this paper is that it analyzes characteristics of best performers and worst performers in the dataset of software projects, resulting in 7 Success Factors (a.o. steady heartbeat, a fixed, experienced team, agile (Scrum), and release-based), and 9 Failure Factors (a.o. once-only project, dependencies with other systems, technology driven, and rules- and regulations driven).

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *performance measures, process metrics, product metrics.*

General Terms

Measurement, Economics.

Keywords

Success Factor, Failure Factor, Good Practice, Bad Practice, Productivity, Time-to-market, Quality, Agile, Learning Cycle.

1. MOTIVATION

1.1 Problem Statement

Growing complexity faces many software engineering companies nowadays with a portfolio- and project control capability that is lagging behind with their high IT-expenditure. A trend towards rapid application development, acceleration of the pace of change in information technology, in organizations, in competitive countermeasures, and in the environment has caused increasing frustration with heavyweight plans [1]. An answer to this challenge can be found in a need for instruments and techniques that support transparency and orchestration of software engineering activities, showing organizations what they can learn from their best performing projects (good practices), and from their worst performing projects (bad practices). Especially in complex environments successful software engineering requires companies to pay special attention to a learning capability that supports flexible portfolio- and project management to prevent from decreasing productivity, increasing time-to-market, and low software quality [2]. However this problem seems hard to solve. Still many software development organizations have enormous difficulties developing reliable effort estimates that result in on-time and on-budget delivery of their software products [3], [4]. Many companies have no or limited historic reference data on their software engineering activities available [5], [6]. This limits existing solutions on software estimation to immature and unreliable estimation techniques and hinders learning. An often-heard goal of continuous improvement seems searching for the pot of gold at the end of the rainbow.

What can we learn here from historic data that is collected in three software engineering companies that on a daily basis had to cope with such highly complex project portfolios?

1.2 Research Objectives

Within the scope of the implementation of several measurement programs as part of process improvements, a large dataset containing 352 software projects was collected in practice in three different organizations during a time-span of six years (2008 to 2013). We define the following research question:

What factors affect software project performance, and what actions can be taken to increase project performance when building a software project portfolio?

1.3 Context

Data was collected on finalized software engineering projects within three different companies. The companies – two large banks (in this paper referred to as 'B1' and 'B2'), and one telecom provider (referred to as 'T') – were, with regard to their software engineering activities, comparable to each other. The size of the software project portfolio of the banks was considerably larger than that of the Telco: The measured yearly throughput of software projects for B1 was approximately 10,000 function points (FPs) [7] [8], for B2 this was approx. 8,000 FPs per year, and T measured approx. 2,000 FPs per year. However, on a business domain scale the software engineering activities were equal in many ways. Software engineering was organized in projects or releases, different delivery models were adopted, and software engineering was characterized by a variety of programming languages and business domains (varying from business intelligence systems to mobile apps).

In all three companies a comparable and similar approach for measurement and analysis of software projects was implemented during the data collection period. A so-called *learning-cycle* was implemented: (1) finalized software projects were measured, col-

Table 1. Overview of the measurement repository

Category	Type	Occurrence	N	Definition of Project Factors
Organization ID (ORG)	Nominal	3	352	Identification code of the organization where a project was performed; three organizations were applicable (number of occurrence between brackets): B1 (206), B2 (125), T (23).
Project ID	Nominal	352	352	Identification code of a project.
Year of Go Life	Ordinal	6	352	Year when a project was finalized; the following years Go Live were applicable: 2008 (32), 2009 (59), 2010 (81), 2011 (131), 2012 (41), 2013 (10).
Business Domain (BD)	Nominal	10	352	Customers business sector; the following BD were applicable: Finance & Risk (54), Internet & Mobile (54), Payments (50), Client & Account Management (incl. CRM systems) (46), Savings & Loans (40), Organization (incl. HRM) (31), Call Centre Solutions (21), Mortgages (21), Data warehouse & BI (18), Front Office Solutions (17).
Primary Programming Language (PPL)	Nominal	21	352	Primary used programming language; the following PPL were applicable: JAVA (154), .NET (59), COBOL (55), ORACLE (29), SQL (9), 3GL (8, unknown was what specific languages were applicable here), Visual Basic (6), RPG (6), FOCUS (5), PowerBuilder (5), PRISMA (4), MAESTRO (3). In the analysis 4th Generation (1), PL1 (1), JSP (1), C++ (1), Clipper (1), Document (1), PL/SQL (1), Siebel (1) and Package (1, unknown what specific language was applicable) were referred at as Other.
Delivery Model (DM)	Nominal	2	352	Classification of the used delivery model; two DM were applicable: Structured (e.g. Waterfall) (307), and Agile (Scrum) (45). One project reported as DM RUP is included in the analysis of Structured.
Development Class (DC)	Nominal	4	352	Classification of the development; the following DC were applicable: New development (173), Major enhancement (25-75% new) (124), Minor enhancement (5-25% new) (27), Conversion (28).
Project Keyword (KW)	Nominal	20	351	Characteristics on a specific project (multiple keywords could be mapped on one project, on one project no keyword was mapped); the following keywords were applicable: Single-application (270), Business driven (150), Release-based (one application) (144), Once-only project (122), Phased project (part of program) (65), Fixed, experienced team (62),), Technology driven (58), Steady heartbeat (49), Dependencies with other systems (41), Migration (35), Rules & Regulations driven (33), Multi-application release (21), Many team changes, inexperienced team (17), Package with customization (16), Legacy (15), Security (14), Pilot; Proof of Concept (10), Bad relation with external supplier (9), New technology, framework solution (3), Package off-the-shelf (1).
Measure	**Type**	**Occurrence***	**N**	**For every project in the repository the measurements indicated in the table below are inventoried.**
Size (FP)	Ratio	-	352	Size of a project in Function Points (FPs).
Duration	Ratio	-	352	Duration of a project in Months; measured from the start of Project Initiation to (technical) Go Live.
Cost	Ratio	-	352	Cost of a project in Euros; measured from the start of Project Initiation to (technical) Go Live.
Effort	Ratio	-	352	Effort spent in a project in Person Hours (PHRs); measured from the start of Project Initiation to (technical) Go Live.
Defects	Ratio	-	172	The number of errors or faults found in a project from System Integration Test to (technical) Go Live. Not for all projects defects were administrated; for 172 projects defects info was recorded in the repository.

*No occurrences are indicated for the 5 Measures, due to the fact that these are different for every measured project.

lected in a measurement repository, and analyzed, (2) data of groups of finalized projects was analyzed on specific trends and benchmarked with internal and external peer-groups, and (3) finally trends were incorporated in an estimation process for newly to be started projects. With regard to step 1 of the learning cycle the measurement repository included both quantitative project data (e.g. size, cost, effort, duration, and defects) and qualitative project data (e.g. reasons named by the project manager that could explain the projects' performance).

In this paper we subsequently discuss research design, execution, analysis, interpretation, related work, and conclusions and future work.

2. RESEARCH DESIGN

2.1 Approach

There may be a large gap between how different stakeholders, and researchers, define success and failure of software projects. Similarly to Lindberg [9], we relate success and failure within this research to better or worse than average cost, effort, and duration performance. Although, where Lindberg compares to industry, we posed to focus the study on identifying success and failure factors of the sample software projects themselves. Every single project from the sample is compared with the average performance of the whole repository. The idea behind this is that a focus at identifying projects that performed better than average, and projects that

performed worse than average, might help companies to realize their ultimate goal of continuous improvement. The refined research objectives of this study are to identify factors that are instrumental in software project success or failure. Following from that, we analyze actions that help to improve the performance of such projects. In the scope of this study a performance-rating 'better than others' must be read as better than the average performance of the whole sample. 'Worse than others' must be read as worse than the average performance of the whole sample.

2.2 Design

The strategy for this research can be defined as a data analysis project. We did not study individual projects in our sample, but the primary author collected and maintained the majority of the projects in the measurement repository. A minority of the projects was collected by third parties. Our goal was to explore the performance of finalized software projects and identify key success and failure factors in order to build a Good Practice Project Portfolio. Our study proposition was twofold. First, we hypothesized that within the repository good performers and bad performers can be distinguished in a quantitative way. Second, we expected that the qualitative information in the repository gives valuable, additional information on success and failure of the projects and on software engineering projects in general.

2.3 The Measurement Repository

Table 1 gives an overview of the measurement repository, including the different aspects that are analyzed in our research. All data in the repository was collected before we started our research, and only with practical analysis purposes in mind. All data is about finalized projects: the dataset does not hold any data on prematurely stopped or failed projects. All projects are related to solution delivery, i.e. an information system was modified or completely new designed. The dataset contains software engineering projects, which in some cases contain an infrastructure component or an implementation of middleware: no full infrastructure or middleware projects are included in the repository. With regard to our research, the dataset is where possible and applicable, discussed with the measurement team that was responsible for the collection. For the research paper all available data in the repository has been used; no deviations or outliers have been removed from the dataset.

The repository contains both quantitative data (e.g. ratios on size, duration, and cost), and qualitative data (e.g. applicable keywords). The repository holds data of 352 software engineering projects carried out in three companies during a period from 2008 to 2013. The projects represent a total amount spent of 266M Euro. Project cost range from 12K Euro to 6.8M Euro. The sum of function points across projects in the repository is 91,105 FPs; ranging from projects with a size of 5 FPs to 4,600 FPs. The project duration ranges from 0.9 Months to 26.8 Months.

Qualitative data is recorded for the following research categories:

- Business Domain (BD);
- Primary Programming Language (PPL);
- Organization (ORG);
- Delivery Model (DM);
- Development Class (DC);
- Size Category (SC), and;
- Project Keyword (PK).

Within these 7 research categories are in total 56 project factors inventoried in the repository (see Table 1 for an overview).

For all inventoried projects size is measured in Function Points (FPs). Function Point Analysis (FPA) has been performed either by an expert member of a measurement team or an external certified FPA-specialist, according to ISO-standardized functional size measurement (FSM) methods [8] [7]. FPA was performed based on sets of final project documentation that usually were delivered by the project manager. All projects were collected and analyzed according to the so-called SEI Core Metrics, a standard set of process-based software engineering metrics [10] [11]. Project duration was measured in a number of months from the start of the project initiation to technical Go Live. Effort was measured in hours, although, measurement specialists that were responsible for the data collection reported that data quality with regard to effort was low, especially where (globally distributed) external suppliers were involved in a project. Because of that we decided to use both effort and cost data for analyzing purposes. Project cost – in this case all project related cost excluding investments (e.g. software license costs) - were recorded in Euros in the measurement repository. Besides that for a limited set of projects (N = 172) the number of defects was recorded.

2.4 Analysis procedure

All software projects in the measurement repository are analyzed from a portfolio point of view, meaning that we were interested in particular in the mutual coherence between the measured projects. We perform the analysis of the portfolio (the measurement repository) in four steps:

1. First, we analyze the overall average performance of all projects in the repository with regard to project size, project cost, and project duration, measured in function points, euros, and months, respectively.

2. Subsequently, we analyze what projects performed as a good practice, and what projects did perform as a bad practice. In the context of our research a project is assessed as a good practice when the performance on both cost and duration is better than the average cost and duration corrected for the applicable project size. A project is assessed as a bad practice when the performance on both cost and duration is worse than the average cost and duration again correct for the project size. To do so we classify the projects in a Cost/Duration Matrix (see Figure 2). We challenge the project performance exclusively against the performance of the whole sample (internal benchmarking). We compare the outcome of our analysis with other studies in Section 6 on Related Work.

3. Once all projects are classified in the Cost/Duration Matrix, we analyze how the 56 project factors (as defined Table 1) are related to the four quadrants of the matrix. This analysis results in a percentage based on number of projects per quadrant for every project factor and a percentage based on cost per quadrant for every factor. We assess the outcome of the analysis by calculating the significance of both outcomes (a range of percentages per quadrant based on number of projects, versus a range of percentages per quadrant based on project cost), by performing a chi-square test. An example: for the Primary programming Language (PPL) ORACLE 29 projects are measured, of which 26 small releases (measured in cost and size) score as a Good Practice. Although, when assessed based on project cost it shows that 3 large projects (in size and cost) score as a Bad Practice. This leads to an indistinct result that from a number of project point of view PPL ORACLE scores high in Good Practice, and from a cost point of view it scores

high in Bad Practice. In the further analysis factors with these kinds of indistinct outcomes will be excluded.

4. We identify factors that are strongly related for the composition of software engineering project portfolios, by analyzing specific subsets per research aspect. In other words: 'What factors should be embraced when composing a project portfolio?' and 'What factors should be avoided when doing so?' For this analysis we define a research aspect to be 'strongly related' when the percentage Good Practice or Bad Practice was 50% or more.

Once an inventory of strongly related factors is finalized we test for each of them whether it indeed affects the probability that a project ends up as a Good or Bad Practice. To that end, for each strongly related factor F, we generate a null and alternative hypothesis:

- H(F, 0): Factor F does not influence good (or bad) practice;
- H(F, 1): Factor F influences good (or bad) practice.

To test these hypotheses, we use the binomial distribution to estimate the chance that our actual observation takes place under the null hypothesis, as follows.

Let n be the total number of projects in the repository. Let *prob(F, n)* be the proportion of projects for which *F* holds in *n*, and let *k* be the number of good (bad) practice projects in which *F* holds. Then *p* is given by the binomial distribution as follows:

*p = (n over k) * prob(F,n)^k (1 - prob(F,n))^(n-k)*

We reject the null hypothesis in favor of the alternative hypothesis if the significance *p* is below 0.05.

3. EXECUTION

3.1 Distribution of the Sample

The distribution of the dataset is described as a positively skewed distribution, or one whose elongated tail extends to the right end of the range. The mean duration in the sample is 8.7 Months, while the median duration is 8.0 Months (min 0.9 Months, max 26.8 Months). The mean project cost is EUR 750,777, while the median project cost is EUR 459,150 (min EUR 12,233, max EUR 6,802,466). The mean project size is 259 FPs; the median project size is 147 FPs (min 5 FPs, max 4,600 FPs).

Although due to the positively skewed distribution analysis based on the median (instead of the mean) might be preferable (the expectation is that outliers will interfere the outcome less), we use the mean in our analysis of factors. After performing both analyses we found, except for some small differences in numbers and percentages, no differences in the outcome of the study (Success and Failure Factors). We assume the Central Limit Theorem to be applicable in this case.

4. ANALYSIS

4.1 Overall Performance Analysis

We analyze the overall weighted average performance, with project size (FPs) as weighting factor, of all projects in the repository with regard to project size, project cost, and project duration. For this purpose we calculate three Performance Indicators:

1. Productivity (PROD); expressed in cost per size unit (EUR/FP) and size unit per hour (FP/HR);
2. Time-to-Market (TTM); expressed in project calendar days per size unit (Days/FP);

Table 2. Average Performance per Size Category.

Performance Indicator	SP	SMP	LMP	LP	Overall
PROD (EUR/FP)	4,364	3,395	2,508	2,111	2,929
PROD (FP/HR)	0.024	0.034	0.045	0.047	0.037
TTM (Days/FP	2.74	1.03	0.75	0.38	1.08
PQ (Defects/FP)	0.20	0.13	0.13	0.21	0.18
Percentage in Sample	62%	19%	10%	9%	100%

Explanation of abbreviations: SP = Small Projects (<200 FP); SMP = Small Medium Projects (201-400 FP); LMP = Large Medium Projects (401-600 FP); LP = Large Projects (>601 FP); PROD = Productivity in resp. Euros per FP and FP per hour; TTM = Time-to-Market in Days per FP; PQ = Process Quality in Defects/ FP. The indicators are calculated as weighted average, with Size as weighting factor (e.g. Cost (Euros) divided by Size (FP), instead of number of projects as weighting factor.

3. Process Quality (PQ); expressed in quality of the process per size unit (Defects/FP).

In Table 2 the values of these indicators are inventoried. In order to get more insight in the effects of economy of scale, we divided the sample in four size categories. The used bins, size categories of 200 FP each, all projects larger than 600 FPs are considered large, are commonly used as a measure for project size in two of the applicable companies. Table 2 gives an overview of the weighted average scores per size category within the measurement repository with regard to the Performance Indicators. As the table shows economy of scale does play an important role here: the performance of the projects in the repository, measured in Productivity, Time-to-Market, and Process Quality, is related to the size of a project. The larger the project; the better the performance on average is in terms of time, money, and quality. The table further shows that most projects in the repository could be categorized as small projects (62% of the projects are smaller than 200 FP).

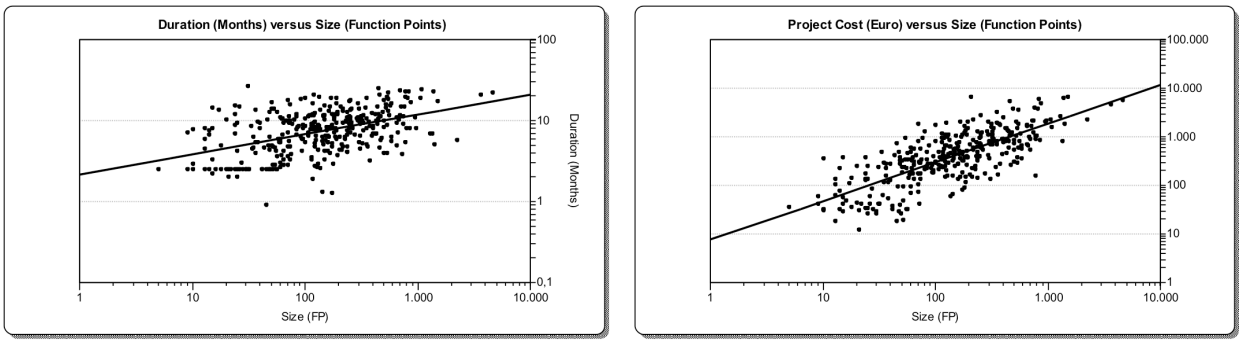
The analysis shows a remarkable fact: while on the one hand medium sized projects show the best performance in terms of time, money, and quality, on the other hand companies give preference to build their portfolio on small or small medium sized projects. An often witnessed adage that 'small projects do not fail while large projects often do' might play a role here. An interesting side-effect of the observation is that nowadays software companies, in an attempt to become more agile, tend to opt for small releases (however this is not always the case: our sample holds two large Scrum releases of resp. 1,067 FPs and 4,600 FPs). Yet maybe active steering on economy of scale can be an equally effective – or even more effective – improvement strategy for software companies. We did not study this side-effect; however future research on the background of economy of scale versus agile development methods might help software companies to find an optimum on project size when building a portfolio.

4.2 The Cost/Duration Matrix

As a second step in the analysis all projects from the repository are classified in the four quadrants of a Cost/Duration Matrix, by combining two Plotter Charts (see Figure 1):

1. A chart (left in Figure 1) where all projects from the repository (N = 352) are plotted in Size (FP) versus Duration (Months). This plotter chart indicates what projects score below the average trend line (M = 7.995, SD = 5.089, $r^2 = 0.21$) with regard to project duration, meaning the project duration

Figure 1. Two plotter charts representing Size (FP) versus Duration (Months) and Size (FP) versus Project Cost (Euros).



is shorter than average, and what projects score above the average trend line, meaning the project duration is longer than average.

2. A chart (right in Figure 1) where all projects from the repository (N = 352) are plotted in Size (FP) versus Cost (Euros). This plotter chart indicates what projects score below the average trend line (M = 750777, SD = 1019949, $r^2 = 0.56$) with regard to project cost, meaning the project cost are less than average, and what projects score above the average trend line, meaning the project cost are higher than average.

For each project the measure of deviation from the average trend line is calculated and expressed in a percentage; negative when below the average trend line, positive when above the trend line. Based on this percentage all projects from the repository are plotted in a matrix, resulting in four quadrants. Each quadrant is characterized by the measure of negative or positive deviation from the average trend (see Figure 2).

Table 3 gives an inventory of the most important performance indicators with regard to the four Cost/Duration Quadrants. The table clearly indicates that projects that score as Good Practice on average show the best Productivity, Time-to-Market, and Process Quality of all four quadrants. And for Bad Practices these performance indicators are the lowest of all four quadrants. The cost of a FP is for example for a Bad Practice as much as four times higher than the average cost for a FP of a Good Practice.

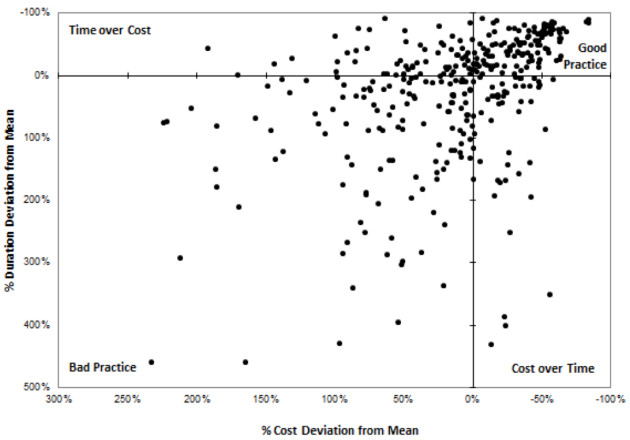
4.3 Analysis of factors

Once all projects have been classified in the Cost/Duration Matrix, we analyze how the 56 project factors are related to the four quadrants of this matrix (respective Good Practice, Bad Practice, Cost over Time, and Time over Cost). Two different perspectives are analyzed: the distribution over the four quadrants per number of projects and the distribution over the four quadrants based on project cost. In order to find significant differences between the two perspectives (i.e. number of projects and project cost) we use the chi-square test. After removal of non-significant results we base the interpretation at the percentage number of projects per Cost/Duration Quadrant; resulting in an average percentage per quadrant for all 56 research aspects. A summary of the total analysis, including the result of the chi-square test, is established in the appendix of the accompanying technical report [12].

Table 3. Average Performance per Cost/Duration Quadrant.

Performance Indicator	GP	CoT	ToC	BP
PROD (EUR/FP)	1,285	1,448	3,834	5,285
PROD (FP/HR)	0.082	0.064	0.028	0.021
TTM (Days/FP)	0.64	0.92	0.75	1.77
PQ (Defects/FP)	0.06	0.20	0.19	0.27
Number in Sample	114	55	51	131

Figure 2. Cost/Duration Matrix.



Explanation of the four quadrants in the Cost/Duration Matrix above:

Good Practice (GP) (upper right): This quadrant shows projects that scored better than the average of the total repository for both cost and duration.

Cost over Time (CoT) (bottom right): In this quadrant projects are reported that scored better than the average of the total repository for cost, yet worse than average for duration.

Time over Cost (ToC) (upper left): In this quadrant projects are plotted that scored better than the average of the total repository for duration, however worse than average for project cost.

Bad Practice (BP) (bottom left): This quadrant holds projects that scored worse than the average of the total repository for both cost and duration.

4.4 Success and Failure Factors

As a fourth and last step we identify specific success factors and failure factors for the composition of software engineering project portfolios, by analyzing how the scores of the different research aspects relate to the Cost/Duration Quadrants. We identify what aspects are strongly related (50% or more) to a high percentage of Good Practice and what aspects are strongly related (50% or more) to a high percentage of Bad Practice.

5. EVALUATION

In this section we evaluate results and implications of the study. Analysis results in an inventory of factors that are strongly related to a high percentage of Good Practice (also referred at as Success Factors) and factors that are strongly related to a high percentage of Bad Practice (Failure Factors). To create better insight in the measure of modification of the factors, both Success and Failure Factors are classified into categories of factors that are IT-organizational, business-organizational, and factors that are primarily technical. In total 10 research aspects were found to be strongly related to Good Practice (Success Factors), and 13 research aspects strongly related to Bad Practice (Failure Factors). Besides that we found 1 factor that is strongly related to a high percentage of Cost over Time, and 2 factors that could be related to a high percentage of Time over Cost.

5.1.1 Factors excluded from the inventory

In order to find significant differences between the two perspectives (i.e. number of projects and project cost) we use the chi-square test. We have found a number of significant differences between the two perspectives: 8 out of 56 factors are excluded from the interpretation. One factor, KW Package off-the-shelf (without customization), scores as related to a high percentage of Good Practice; however only one such project was in the sample. With regard to 6 factors we found a low significance between the percentages GP, CoT, ToC, and BP, measured on number of projects and the percentages measured on project cost (χ^2 of lower than 5). These 6 excluded factors are:

- PPL Oracle, $\chi^2(1, N = 29) = 0.38, p = .00$.
- BD Finance & Risk, $\chi^2(1, N = 54) = 3.27, p = .00$.
- PPL RPG, $\chi^2(1, N = 6) = 2.68, p = .18$.
- KW Phased project, $\chi^2(1, N = 65) = 4.26, p = .05$.
- DC Minor Enhancement, $\chi^2(1, N = 27) = 2.78, p = .04$.
- PPL 3GL, $\chi^2(1, N = 8) = 3.18, p = .19$.

Two of these factors, PPL ORACLE (86% GP based on number of projects) and BD Finance & Risk (69% GP based on number of projects) both score as factors that are strongly related to a high percentage of Good Practice, for which a remark is in place. PPL ORACLE scores with a relation to a high percentage of Good Practice when looked upon from number of projects (86%), however it also has a high percentage of Bad Practice when looked upon from project cost (73%). This effect is caused by the fact that a sample of 29 PPL ORACLE projects is analysed, including 26 very well performing small releases belonging to two applications [13]). The other three medium sized PPL ORACLE projects did not perform well; two score as a Bad Practice, and one scores in the Time over Cost quadrant, resulting in a low χ^2-score. This also affects the score for BD Finance & Risk, as all 26 good performing projects are performed within this domain, including one medium sized badly performing project that influences the score with regard to project cost, again leading to a low χ^2-value.

Finally a factor that is excluded from the inventory is the category PPL Other: representing a collection of 9 primary programming languages that were recorded for only one project each in the repository.

5.1.2 Factors strongly related to Good Practice

We identified 10 factors that were strongly related to a high percentage of Good Practice:

Table 4. Overview of factors strongly related to Good Practice.

Project Factor	% GP	*N*	*p*
PPL Visual Basic	83	6	.03
PPL FOCUS	80	5	.06
KW Steady heartbeat	71	49	.00
KW Fixed, experienced team	66	62	.00
BD Data Warehouse & BI	61	18	.02
PPL PowerBuilder	60	5	.14
DM Agile (Scrum)	56	45	.00
PPL SQL	56	9	.10
BD Organization	52	31	.02
KW Release-based (one application)	50	144	.00

5.1.3 Success Factors for Software Projects

After testing for statistical significance we found that of those 10 factors 7 satisfied the alternative hypothesis (H_1) (see Figure 3). 4 factors of these 7 are tested to be strongly significant ($p < 0.01$) to influencing project success (high probability to end up as a Good Practice), and due to that can be specified as strongly significant Success Factors for Software Projects (ranking based on probability of success):

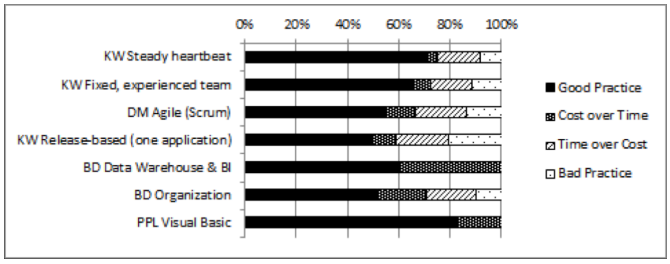
1. Steady Heartbeat;
2. Fixed, experienced team;
3. Agile (Scrum);
4. Release-based (one application).

It is likely that these four strongly significant Success Factors are related to each other, since release-based working, a steady heartbeat, and a fixed, experienced team, are in fact preconditions of an agile (Scrum) way of working. However keep in mind that not all projects where factor 1, 2, or 4 was applicable used an agile delivery model. In fact all types of projects can adopt these Success Factors, without opting for a specific delivery model. The promising idea behind these four Success Factors is that they all can be implemented relatively easy and fast in any software engineering company: no big organizational or technical changes are needed to start working according to these Success Factors. In a way an organization can decide to start working this way by tomorrow.

Besides these four strongly significant Success Factors, we found out of the group of 7 factors that were strongly related to Good Practice, 3 factors with a significant probability ($p < 0.05$) to perform as a Good Practice (ranking based on probability):

5. Business Domain Data Warehouse & BI;
6. Business Domain Organization;
7. Programming Language Visual Basic.

Figure 3. Overview of Success Factors.



One could argue that a difference is applicable with regard to the three Success Factors above with the Success Factors 1 to 4 on the fact that (where 1 to 4 seems relatively easy to implement) 5 to 7 are more difficult to change. A software company cannot simply change its business organization structure overnight, and opting for another programming language asks for a more long term approach. As an example we do see implementations in practice where a company actively builds its application portfolio on a limited number of programming languages, however in many cases the reason behind such an improvement strategy lies in steering on shortage versus availability of language-specific developers (avoiding of high labor cost) instead of implementing continuous performance improvement.

Although we found a strong significance for PPL Visual Basic a remark is in place: 6 projects were analyzed of which 5 scored as Good Practice and one as Cost over Time. The projects took place in two different companies and varied in size from 27 to 586 FPs.

5.1.4 Factors strongly related to Bad Practice

We identified 13 factors that were strongly related to a high percentage of Bad Practice:

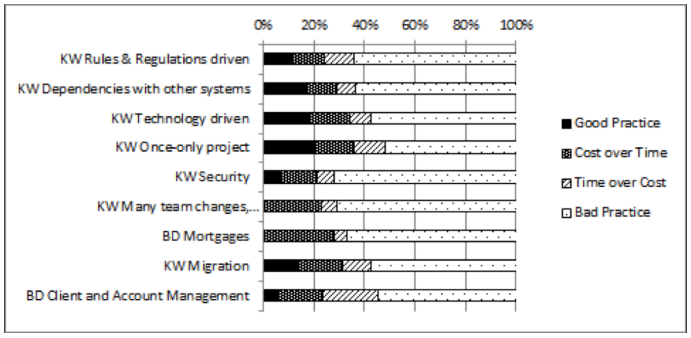
Table 5. Overview of factors strongly related to Bad Practice.

Project Factor	% BP	*N*	*p*
KW Security	71	14	.02
KW Many team changes, inexperienced team	71	17	.01
BD Mortgages	67	21	.01
KW Bad relation with external supplier	67	9	.07
KW New technology, Framework solution	67	3	.21
KW Rules & Regulations driven	64	33	.01
KW Dependencies with other systems	63	41	.00
KW Legacy	60	15	.05
KW Migration	57	35	.02
KW Technology driven	57	58	.00
BD Client & Account Management	54	46	.01
KW Once-only project	52	122	.00
KW Pilot, Proof of Concept	50	10	.15

5.1.5 Failure Factors for Software Projects

After testing for statistical significance we found that of those 13 factors 9 satisfied the alternative hypothesis (H_1) (see Figure 4). 4 factors of these 9 are tested to be strongly significant ($p < 0.01$) to influencing project failure (high probability to end up as a Bad Practice), and due to that can be specified as strongly significant

Figure 4. Overview of Failure Factors.



Failure Factors for Software Projects (ranking based on probability of failure):

1. Rules & Regulations driven;
2. Dependencies with other systems;
3. Technology driven;
4. Once-only project.

An interesting relation can be assumed here between the strongly significant Success Factor Release-based, and the strongly significant Failure Factor Once-only project. Although our research does not show, we assume that starting up a once-only project, including having to do things for the first time, and for one project only, leads to a high probability of ending in between Bad Practice. On the other hand the repeating character of release-based working, including the effect of learning on-the-job and creating an experienced team, creates a high probability to end up as a Good Practice.

Next to these four strongly significant Failure Factors, we found that 5 factors from the group of strongly related to a high percentage of Bad Practice showed a significant ($p < 0.05$) probability to perform as a Bad Practice (ranking based on probability of failure):

5. Security;
6. Many team changes, inexperienced team;
7. Business Domain Mortgages;
8. Migration;
9. Business Domain Client and Account Management.

5.1.6 Factors related to CoT and ToC

We did not find any factors that were strongly related to a high percentage of Cost over Time (cost lower than average, duration longer than average) or Time over Cost (cost higher than average, duration shorter than average) that where significant ($p < 0.05$) for respectively CoT or ToC.

5.2 Discussion

In this section we discuss the most important limitations with regard to the study. Owing to the fact that the data of software engineering projects that is used for the research was collected in a practical setting and primary for practical purposes, this paper must emphatically been seen as the result of analysis that was performed on an existing measurement repository. As a consequence of that we had to cope with the data that was available: no additional data was to be collected at a later stage. The dataset only contained data of finalized (successful) projects; no data of prematurely stopped or failed projects was available. In particular for the identification of Bad Practices, the study of failing projects will be relevant.

5.3 Interferences

Analysis of the performance of the separate organizations showed that Organization itself was not a distinguishing factor that could be strongly related to either a high percentage of Good Practice or Bad Practice. We assume that the results of our research generalize given this finding, in combination with the fact that our research included three different software companies.

6. RELATED WORK

In the following, we discuss the contribution of our study in the context of earlier research. Much has been written on identification of success and failure factors for software engineering, in many cases with reference to software process improvement (SPI), a popular process-based approach to delivering improvements in software products from the 80s [14]. Reel [15] argues that in software, more "advanced" technologies are far less critical to improving practice than five essential factors to managing a successful software project: start on the right foot, maintain momentum, track progress, make smart decisions, and institutionalize post-mortem analyses.

Later research seems to focus more at process specific factors. Dybå [16] did find strong support for SPI success to be positively associated with business orientation, employee participation, concern for measurement, and exploitation of existing knowledge, and partial support for SPI success to be positively associated with involved leadership and exploration of new knowledge. In another study Dybå [17] showed that small organizations reported that they implement SPI elements as effectively as large organizations, and in turn, achieve high organizational performance. Niazi et al. [18] identified seven factors that are generally considered critical for successfully implementing SPI: higher management support, training, awareness, allocation of resources, staff involvement, experienced staff and defined SPI implementation methodology. Rainer et al. [19] found four factors (reviews, standards and procedures, training and mentoring, and experienced staff) that practitioners generally considered had a major impact on successfully implementing SPI, and a further four factors (internal leadership, inspections, executive support and internal process ownership) that the more mature companies considered had a major impact on successfully implementing SPI.

In the 90s more advanced process improvement models such as CMMI, and ISO's SPICE were introduced [20], leading to supplementary research on success and failure factors. Stelzer and Mellis [21] describe ten factors that affect organizational change in software process improvement initiatives based on the CMM and ISO quality standards: management commitment and support, staff involvement, providing enhanced understanding, tailoring improvement initiatives, managing the improvement project, change agents and opinion leaders, stabilizing changed processes, encouraging communication and collaboration, setting relevant and realistic objectives, and unfreezing the organization. Mahmood et al. [20] inventoried 3 categories of critical success factors: awareness (senior management commitment, training and mentoring, staff involvement, awareness of SPI), organizational (creating process action teams, experienced staff, staff time and resources, formal methodology), and support (reviews). The following critical barriers were identified: lack of awareness, lack of support, lack of resources, time pressure, inexperienced staff, organizational politics, and lack of formal methodology. Procaccino et al. [22] found the most important factors for project success to be: (1) the presence of a committed sponsor and (2) the level of confidence that the customers and users have in the project manager and development team. Charette [23] names 12 factors why software fails so often, however there is no clear link with the results from our research with regard to these factors.

The common idea in these papers is that success and failure were interconnected with process-based activities: in other words, follow the process and success will come. Looking at the results of our research, a close link with process related research, often based on software projects performed during the last two decennia of the former century, seems absent due to the fact that we did not study this: no clearly process related (in terms of SPI, CMMI, or ISO) factors were present in our list of project factors.

Recent research focusing on the factors that affect success and failure of agile software engineering is more similar. Chow et al. [24] state that, 'as long as an agile project picks a high-caliber team, practices rigorous agile software engineering techniques and executes a correct agile-style delivery strategy; the project could be likely to be successful'. Three other factors that could be critical to certain success dimensions are: a strict agile project management process, an agile-friendly team environment, and a strong customer involvement. Although not stated in the same words a similarity with the results from our research is noticeable here. On the other hand we did not find evidence that some assumed prerequisites for success of agile projects such as strong executive support, strong sponsor commitment, ready availability of physical agile facility, or agile-appropriate project types, are actually critical factors for success. Misra et al. [25] found nine factors that have statistically significant relationship with success in adopting agile software development practices: customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture, control, personal characteristics, societal culture, and training and learning. No clear link with results from our research is found here. Sutherland et al. [26] studied best practices in a globally distributed Scrum environment, with fixed, experienced agile teams, and comes up with, as the paper states 'the most productive Java projects ever documented'. Maybe stated differently, however, a match with our findings is obvious here. Otherwise, the idea that agile is always linked to project success is not shared by Estler et al. [27] stating that 'choosing an agile rather than a structured process does not appear to be a crucial decision for globally distributed projects.'

Our research indicates a relation between agile (Scrum) and economy of scale (project size). Boehm [1] states that 'the most widely adopted agile method has been XP, whose major technical premise was that its combination of customer collocation, short development increments, simple design, pair programming, refactoring, and continuous integration would flatten the cost-of change-vs.-time curve. However, data reported so far indicate that this flattening does not take place for larger projects.' Apparently the success of agile does not hold for larger sized projects, something that matches an assumption in our research, since no medium large or large projects were performed based at an agile delivery method. The connection between agile (Scrum) and project size is not clear yet and seems a challenge for future research.

A fascinating gap in related research turns up with relation to the finding from our research that a programming language (Visual Basic) and specific business domains (Data Warehouse & BI, Organization) are significant for project success. It might be surprising that a view at improvement, focusing at benchmarking the outcome of the software process, instead on the software process itself, seems less represented in related work. Jones [28] inventories successful and unsuccessful project technologies, stating that unsuccessful projects are related to a lack of measurement activi-

ties, and conversely, successful projects are related to accurate measurement and analysis activities. Jones [6] inventoried a large amount of business domain and programming language specific effects on project performance. In the 90s he performed extensive research on performance aspects related to domain-specific and language-specific software engineering, however since that not many additional research on this area, and especially on the effects of agile delivery models, seem to be performed. Premrai et al. [29] investigated how software project productivity had changed over time, finding that an improving trend was measured, however less marked since 1990. The trend varied over companies and business sectors, a finding that matches the result of our research with regard to differentiation over business domains.

7. CONCLUSIONS AND FUTURE WORK

7.1 Impact

We found 7 Success Factors for software projects: (1) Steady Heartbeat, (2) Fixed, experienced team, (3) Agile (Scrum), (4) Release-based (one application), (5) Business Domain Data Warehouse & BI, (6) Business Domain Organization, and (7) Programming Language Visual Basic.

We found 9 Failure Factors for software projects: (1) Rules & Regulations driven, (2) Dependencies with other systems, (3) Technology driven, (4) Once-only project, (5) Security, (6) Many team changes, inexperienced team, (7) Business Domain Mortgages, (8) Business Domain Client and Account Management, and (9) Migration.

Based on the findings we identify the following guidelines for practice when aiming for a Good Practice Project Portfolio:

1. Avoid Bad Practice by steering at limitation of interdependencies between projects and systems, stay away from unnecessary team changes, and build teams where possible with experienced team-members.
2. Create Good Practice by actively steering on organizing software development in a release-based way, set up fixed teams with experienced team-members, implement a steady heartbeat, and go for an agile (Scrum) delivery approach where applicable.
3. When setting up a once-only project within a software project portfolio, pay special attention to standardization and limitation of procedures to smoothen the projects progress, re-use of knowledge of other once-only projects (Lessons Learned), and implementation of a learning cycle.
4. Implement a long- and medium term portfolio strategy, including a learning capability, to avoid Bad Practice by limitation (where possible) of projects that are characterized as technology driven, rules & regulations driven, migration of data, and security issues.

7.2 Future work

An interesting side-effect of the observation is that nowadays software companies, in an attempt to become more agile, tend to opt for small releases. Yet maybe active steering on economy of scale can be an equally effective – or even more effective – improvement strategy for software companies. We did not study this side-effect; however future research on the background of economy of scale versus agile development methods might help software companies to find an optimum on project size when building a portfolio. Another aspect that was not covered in our research, yet might help companies to really improve, is whether the software companies did really improve their performance over time. During the six year measurement period many improvement actions were undertaken, yet how successful were these actions in the end? Future research (on our dataset) might reveal this.

8. ACKNOWLEDGMENTS

We thank Hans Jägers, professor emeritus from the University of Amsterdam, Rob de Munnik (QSM Europe), Bart Griffioen, Georgios Gousios, Steven Raemaekers, and all other reviewers for their valuable contributions.

9. REFERENCES

[1] B. Boehm, “A View of 20th and 21st Century Software Engineering,” in *IEEE International Conference on Software Engineering (ICSE)*, Shanghai, Ghina, 2006.

[2] M. Tihinen, P. Parviainen, T. Suomalainen and K. Karhu, "ABB Experiences of Boosting Controlling and Monitoring Activities in Collaborative Production," in *6th IEEE International Conference on Global Software Engineering (ICGSE)*, Helsinki, 2011.

[3] C. Jones, "Software Cost Estimating Methods for Large Projects," *Crosstalk - The Journal of Defense Software Engineering,* pp. 8-12, 2005.

[4] K. Moløkken and M. Jørgensen, "A Review of Surveys on Software Effort Estimation," *Proceedings of ISESE - International Symposium on Empirical Software Engineering,* pp. 223-230, 2003.

[5] A. Dagnino, "Estimating Software-Intensive Projects in the Absence of Historical Data," in *International Conference on Software Engineering (ICSE)*, San Francisco, CA, USA, 2013.

[6] C. Jones, Software, Assessments, Benchmarks, and Best Practices, New York: Addison Wesley Longman, 2000.

[7] IFPUG, IFPUG FSM Method: ISO/IEC 20926 - Software and systems engineering – Software measurement – IFPUG functional size measurement method, New York: International Function Point User Group (IFPUG), 2009.

[8] NESMA, NESMA functional size measurement method conform ISO/IEC 24570, version 2.2, Netherlands Software Measurement User Association (NESMA), 2004.

[9] K. R. Lindberg, "Software developer perceptions about software project failure: a case study," *The Journal of Systems and Software,* vol. 49, pp. 177-192, 1999.

[10] CMMI Product Team, CMMI for Development, Version 1.3, Hanscom: Carnegie Mellon, 2010.

[11] S. Kan, "Metrics and Models in Software Quality Engineering," Addison Wesley Longman, Reading, Massachusetts, 1995.

[12] H. Huijgens, R. v. Solingen and A. v. Deursen, "Technical Report TUD-SERG-2013-019," Delft University of Technology, Delft, 2013.

[13] H. Huijgens and R. v. Solingen, "Measurement of Best-in-Class Software Releases," in *Joint Conference of the 23nd International Workshop on Software Measurement and the Eighth International Conference on Software Process and*

Product Measurement (IWSM-MENSURA), Ankara, Turkey, 2013.

[14] T. Hall, A. Rainer and N. Baddoo, "Implementing Software Process Improvement: An Empirical Study," *Software Process Improvement and Practice,* vol. 7, pp. 3-15, 2002.

[15] J. Reel, "Critical Success Factors in Software Projects," *IEEE Software,* Vols. May-June, pp. 18-23, 1999.

[16] T. Dyba, "An Empirical Investigation of the Key Factors for Success in Software Process Improvement," *IEEE Transactions on Software Engineering,* vol. 31, no. 5, pp. 410-424, 2005.

[17] T. Dybå, "Factors of Software Process Improvement Success in Small and Large Organizations: an Emperical Study in the Scandinavian Context," in *ESEC/FSE*, Helsinki, Finland, 2003.

[18] M. Niazi, D. Wilson and D. Zowgh, "Critical Success Factors for Software Process Improvement Implementation: An Empirical Study," *Software Process Improvement and Practice,* vol. 11, pp. 193-211, 2006.

[19] A. Rainer and T. Hall, "Key success factors for implementing software process improvement: a maturity-based analysis," *Journal of Systems and Software,* vol. 62, no. 2, pp. 71-84, 2002.

[20] M. Niazi, D. Wilson and D. Zowghi, "A maturity model for the implementation of software process improvement: an empirical study," *The Journal of Systems and Software,* 2003.

[21] D. Stelzer and W. Mellis, "Success Factors of Organizational Change in Software Process Improvement," *Software Process Improvement and Practice,* vol. 4, pp. 227-250, 1998.

[22] J. Procaccino, J. Verner and S. Overmyer, "Case Study: Factors for Early Prediction of Software Success & Failure," *Elsevier - Information and Software Technology,* vol. 44, no. 1, pp. 53-62, 2002.

[23] R. N. Charette, "Why Software Fails," *IEEE Computing / Software,* vol. September, pp. 1-9, 2005.

[24] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *The Journal of Systems and Software,* vol. 81, pp. 961-971, 2008.

[25] S. C. Misra, V. Kumar and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *The Journal of Systems and Software,* vol. 82, pp. 1869-1890, 2009.

[26] J. Sutherland, A. Viktorov, J. Blount and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," in *40th International Conference on System Sciences*, Hawaii, 2007.

[27] H. C. Estler, M. Nordio, C. A. Furia, B. Meyer and J. Scheider, "Agile vs. Structured Distributed Software Development: A Case Study," in *IEEE Seventh International Conference on Global Software Engineering (ICGSE)*, Porto Allegre, Brasil, 2012.

[28] C. Jones, "Patterns of large software systems: failure and success," *Computer,* pp. 86-87, 1995.

[29] R. Premrai, M. Shepperd, B. Kitchenham and P. Forselius, "An Empirical Analysis of Software Productivity Over Time," in *IEEE International Symposium Software Metrics*, Como, Italy, 2005.

10. Addendum - Technical Report

Xxx

10.1 Addendum A – Average performance per Size Category

Performance Indicator	Small projects	Small medium projects	large medium projects	Large projects	Total
Productivity (EUR/FP)	4.364	3.395	2.508	2.111	2.929
Productivity (FP/Hr)	0,024	0,034	0,045	0,047	0,037
Time-to-market (Days/FP)	2,74	1,03	0,75	0,38	1,08
Process Quality (Defects/FP)	0,20	0,13	0,13	0,21	0,18
Productivity Index (PI)	10,46	13,34	14,98	16,84	12,02
Number in sample	219	68	33	32	352
Percentage in sample	62%	19%	9%	9%	

10.2 Addendum B – Summary of findings in Cost/Duration plot

Cost/Duration Quadrant	Number of projects	% of Projects	Project cost (EUR)	Avg. Project cost (EUR)	Total Throughput (FP)	% Project Size	Avg. Project Size (FP)	Avg. Productivity Index (PI)
Good Practice	115	33%	34.239.442	297.734	26.655	29%	232	14,67
Cost over Time	55	16%	33.868.793	615.796	23.390	26%	425	12,88
Bad Practice	131	37%	150.429.066	1.148.313	28.466	31%	217	9,11
Time over Cost	51	14%	48.280.768	946.682	12.594	14%	247	12,55
Totals	352		266.818.069		91.105			

10.3 Addendum C – Average Performance Indicator scores in Cost/Duration plot

Performance Indicator	Good Practice	Cost over Time	Bad Practice	Time over Cost
Productivity (EUR/FP)	1.285	1.448	5.285	3.834
Productivity (FP/Hr)	0,082	0,064	0,021	0,028
Time-to-market (Days/FP)	0,64	0,92	1,77	0,75
Process Quality (Defects/FP)	0,06	0,20	0,27	0,19
Productivity Index (PI)	14,67	12,88	9,11	12,55
Number in sample	115	55	131	51
Percentage in sample	33%	16%	37%	14%
Project Cost - Maximum	2.249.310	5.637.000	6.802.466	2.582.689
Project Cost - Third Quartile	365.350	707.450	1.302.180	1.369.781
Project Cost - Median	176.888	374.140	653.760	724.470
Project Cost - First Quartile	46.645	190.706	344.355	441.602
Project Cost - Minimum	12.233	29.700	60.078	35.540

10.4 Addendum D – Average Performance Indicators per Project Factor

Business Domain	**Productivity (EUR/FP)**	**Time-to-market (Days/FP)**	**Process Quality (Defects/FP)**	**Number in sample (*N*)**	**Percentage in sample**
BD Call Center Solutions	2.181	0,57	0,22	21	6%
BD Client and Account Management	5.343	1,87	0,14	46	13%
BD Data Warehouse & BI	960	0,85	0,07	18	5%
BD Finance & Risk	2.636	0,65	0,07	54	15%
BD Front Office Solutions	2.805	1,00	0,21	17	5%
BD Internet & Mobile	3.263	1,53	0,39	54	15%
BD Mortgages	2.303	0,96	0,24	21	6%
BD Organization	1.794	0,70	0,06	31	9%
BD Payments	4.372	1,52	0,34	50	14%
BD Savings & Loans	3.642	1,55	0,08	40	11%
DC Conversion (<5% new)	3.258	1,52	0,18	28	8%
DC Major Enhancement (25-75% new)	2.521	0,84	0,17	124	35%
DC Minor Enhancement (5-25% new)	4.207	1,88	0,28	27	8%

Business Domain	Productivity (EUR/FP)	Time-to-market (Days/FP)	Process Quality (Defects/FP)	Number in sample (*N*)	Percentage in sample
DC New Development	3.143	1,17	0,18	173	49%
DM Agile (Scrum)	1.796	0,65	0,24	45	13%
DM Structured	3.116	1,15	0,17	307	87%
KW Bad relation with external supplier	3.859	1,15	0,74	9	3%
KW Business driven	2.249	0,90	0,24	150	43%
KW Dependencies with other systems	3.650	1,56	0,36	41	12%
KW Fixed, experienced team	1.792	0,57	0,27	62	18%
KW Legacy	4.727	1,43	0,73	15	4%
KW Many team changes, inexperienced team	3.994	1,65	0,60	17	5%
KW Migration	3.029	0,97	0,34	35	10%
KW Multi--application release	1.909	1,87	0,08	21	6%
KW New technology, Framework solution	6.398	1,45	0,64	3	1%
KW Once-only project	3.288	1,24	0,14	122	35%
KW Package off-the-shelf	588	0,53	-	1	0%
KW Package with customization	1.552	0,76	0,24	16	5%
KW Phased project (part of program)	3.283	1,12	0,26	65	18%
KW Pilot; Proof of Concept	2.014	0,59	0,34	10	3%
KW Release-based (one application)	2.565	0,82	0,18	144	41%
KW Rules & Regulations driven	4.610	1,45	0,23	33	9%
KW Security	5.051	3,08	0,24	14	4%
KW Single-application	2.535	1,00	0,17	270	77%
KW Steady heartbeat	1.835	0,62	0,19	49	14%
KW Technology driven	3.890	2,27	0,27	58	16%
ORG Organization A	2.678	1,13	0,28	206	59%
ORG Organization B	3.464	0,89	0,03	125	36%
ORG Organization C	2.929	1,87	0,08	21	6%
PPL .NET	2.390	1,08	0,17	59	17%
PPL 3GL	4.227	1,53	0,00	8	2%
PPL COBOL	4.766	1,63	0,33	55	16%
PPL FOCUS	1.055	0,52	0,02	5	1%
PPL JAVA	2.782	1,05	0,21	154	44%
PPL MAESTRO	2.625	0,34	0,06	3	1%
PPL ORACLE	6.956	2,02	0,01	29	8%
PPL Other	2.007	0,88	0,06	9	3%
PPL PowerBuilder	1.924	0,53	0,02	5	1%
PPL PRISMA	3.673	0,66	0,00	4	1%
PPL RPG	3.534	0,87	0,00	6	2%
PPL SQL	1.479	0,81	0,01	9	3%
PPL Visual Basic	1.513	0,69	0,02	6	2%
SC Large Medium Project (401-600 FP)	2.508	0,75	0,13	33	9%
SC Large Project (>601 FP)	2.111	0,38	0,21	32	9%
SC Small Medium Project (201-400 FP)	3.395	1,03	0,13	68	19%
SC Small Project (<200 FP)	4.364	2,74	0,20	219	62%

10.5 Addendum E – Percentage Number of Projects per Quadrant per Project Factor

Business Domain	% Good Practice (nr. projects)	% Cost over Time (nr. projects)	% Bad Practice (nr. projects)	% Time over Cost (nr. projects)
BD Call Center Solutions	33%	19%	24%	24%
BD Client and Account Management	7%	17%	54%	22%
BD Data Warehouse & BI	61%	39%		
BD Finance & Risk	69%	4%	22%	6%
BD Front Office Solutions	47%	12%	29%	12%
BD Internet & Mobile	22%	17%	48%	13%
BD Mortgages		29%	67%	5%

Business Domain	% Good Practice (nr. projects)	% Cost over Time (nr. projects)	% Bad Practice (nr. projects)	% Time over Cost (nr. projects)
BD Organization	52%	19%	10%	19%
BD Payments	22%	14%	44%	20%
BD Savings & Loans	25%	10%	48%	18%
DC Conversion (<5% new)	36%	11%	43%	11%
DC Major Enhancement (25-75% new)	34%	14%	32%	20%
DC Minor Enhancement (5-25% new)	15%	15%	59%	11%
DC New Development	34%	18%	36%	12%
DM Agile (Scrum)	56%	11%	13%	20%
DM Structured	29%	16%	41%	14%
KW Bad relation with external supplier		22%	67%	11%
KW Business driven	38%	19%	32%	11%
KW Dependencies with other systems	17%	12%	63%	7%
KW Fixed, experienced team	66%	6%	11%	16%
KW Legacy	27%	13%	60%	
KW Many team changes, inexperienced team		24%	71%	6%
KW Migration	14%	17%	57%	11%
KW Multi--application release	10%	43%	43%	5%
KW New technology, Framework solution			67%	33%
KW Once-only project	20%	16%	52%	12%
KW Package off-the-shelf	100%			
KW Package with customization	19%	50%	31%	
KW Phased project (part of programme)	25%	22%	46%	8%
KW Pilot; Proof of Concept	10%	40%	50%	
KW Release-based (one application)	50%	9%	20%	21%
KW Rules & Regulations driven	12%	12%	64%	12%
KW Security	7%	14%	71%	7%
KW Single-application	37%	17%	33%	13%
KW Steady heartbeat	71%	4%	8%	16%
KW Technology driven	19%	16%	57%	9%
ORG Organization A	33%	17%	41%	9%
ORG Organization B	36%	8%	30%	26%
ORG Organization C	10%	43%	43%	5%
PPL .NET	42%	15%	31%	12%
PPL 3GL	13%	25%	38%	25%
PPL COBOL	15%	15%	47%	24%
PPL FOCUS	80%			20%
PPL JAVA	19%	19%	49%	12%
PPL MAESTRO			33%	67%
PPL ORACLE	86%		7%	7%
PPL Other	56%	11%	22%	11%
PPL PowerBuilder	60%		20%	20%
PPL PRISMA	25%		25%	50%
PPL RPG	50%	17%		33%
PPL SQL	56%	33%	11%	
PPL Visual Basic	83%	17%		
SC Large Medium Project (401-600 FP)	21%	36%	27%	15%
SC Large Project (>601 FP)	34%	19%	28%	19%
SC Small Medium Project (201-400 FP)	34%	15%	38%	13%
SC Small Project (<200 FP)	34%	12%	40%	14%

10.6 Addendum F – Percentage Project Cost per Quadrant per Project Factor

Business Domain	% Good Practice (project cost)	% Cost over Time (project cost)	% Bad Practice (project cost)	% Time over Cost (project cost)
BD Call Center Solutions	30%	8%	26%	36%
BD Client and Account Management	3%	9%	64%	25%
BD Data Warehouse & BI	41%	59%		
BD Finance & Risk	19%	11%	64%	6%
BD Front Office Solutions	29%	2%	41%	28%
BD Internet & Mobile	7%	11%	69%	13%
BD Mortgages		40%	55%	5%
BD Organization	33%	16%	15%	36%
BD Payments	6%	7%	74%	13%
BD Savings & Loans	7%	5%	59%	28%
DC Conversion (<5% new)	19%	8%	65%	8%
DC Major Enhancement (25-75% new)	13%	18%	41%	28%
DC Minor Enhancement (5-25% new)	1%	7%	64%	29%
DC New Development	13%	10%	66%	11%
DM Agile (Scrum)	22%	36%	20%	22%
DM Structured	12%	10%	60%	18%
KW Bad relation with external supplier		15%	76%	9%
KW Business driven	14%	23%	44%	19%
KW Dependencies with other systems	4%	15%	70%	11%
KW Fixed, experienced team	30%	22%	17%	31%
KW Legacy	9%	5%	86%	
KW Many team changes, inexperienced team		10%	83%	7%
KW Migration	2%	23%	62%	13%
KW Multi--application release	2%	33%	49%	16%
KW New technology, Framework solution			68%	32%
KW Once-only project	10%	11%	67%	12%
KW Package off-the-shelf	100%			
KW Package with customization	4%	71%	25%	
KW Phased project (part of program)	4%	21%	66%	9%
KW Pilot; Proof of Concept	1%	46%	53%	
KW Release-based (one application)	24%	7%	39%	31%
KW Rules & Regulations driven	2%	6%	81%	11%
KW Security	6%	8%	83%	3%
KW Single-application	16%	15%	51%	19%
KW Steady heartbeat	42%	4%	14%	39%
KW Technology driven	9%	9%	75%	8%
ORG Organization A	12%	20%	55%	14%
ORG Organization B	15%	3%	59%	24%
ORG Organization C	2%	33%	49%	16%
PPL .NET	27%	11%	51%	12%
PPL 3GL	1%	6%	51%	41%
PPL COBOL	4%	6%	72%	17%
PPL FOCUS	56%			44%
PPL JAVA	10%	17%	59%	14%
PPL MAESTRO			30%	70%
PPL ORACLE	9%		73%	19%
PPL Other	29%	18%	26%	27%
PPL PowerBuilder	55%		25%	20%
PPL PRISMA	10%		23%	66%
PPL RPG	13%	3%		84%
PPL SQL	32%	32%	36%	
PPL Visual Basic	62%	38%		

Business Domain	% Good Practice (project cost)	% Cost over Time (project cost)	% Bad Practice (project cost)	% Time over Cost (project cost)
SC Large Medium Project (401-600 FP)	8%	22%	54%	16%
SC Large Project (>601 FP)	16%	18%	50%	16%
SC Small Medium Project (201-400 FP)	14%	9%	62%	16%
SC Small Project (<200 FP)	11%	7%	60%	22%

10.7 Addendum G – Statistics per Project Factor

Business Domain	$\chi 2$	Base-rate	Number positive Good Practice	Probability Good Practice (p)	Number positive Bad Practice	Probability Bad Practice (p)
BD Call Center Solutions	9,57	0,06	7	0,15	5	0,10
BD Client and Account Management	10,44	0,13	3	0,00	25	0,01
BD Data Warehouse & BI	10,07	0,05	11	0,02	0	
BD Finance & Risk	3,27	0,15	37	0,00	12	0,01
BD Front Office Solutions	5,91	0,05	8	0,09	5	0,15
BD Internet & Mobile	7,21	0,15	12	0,04	26	0,03
BD Mortgages	13,36	0,06	0		14	0,01
BD Organization	9,48	0,09	16	0,02	3	0,00
BD Payments	6,04	0,14	11	0,04	22	0,07
BD Savings & Loans	6,66	0,11	10	0,09	19	0,05
DC Conversion (<5% new)	8,94	0,08	10	0,12	12	0,11
DC Major Enhancement (25-75% new)	7,68	0,35	42	0,07	40	0,04
DC Minor Enhancement (5-25% new)	2,78	0,08	4	0,04	16	0,02
DC New Development	6,56	0,49	59	0,06	63	0,07
DM Agile (Scrum)	5,59	0,13	25	0,00	6	0,00
DM Structured	7,81	0,87	90	0,00	125	0,00
KW Bad relation with external supplier	13,91	0,03	0		6	0,07
KW Business driven	6,93	0,43	57	0,02	48	0,03
KW Dependencies with other systems	7,44	0,12	7	0,02	26	0,00
KW Fixed, experienced team	5,92	0,18	41	0,00	7	0,00
KW Legacy	6,25	0,04	4	0,18	9	0,05
KW Many team changes, inexperienced team	9,33	0,05	0		12	0,01
KW Migration	6,12	0,10	5	0,02	20	0,02
KW Multi--application release	7,57	0,06	2	0,02	9	0,12
KW New technology, Framework solution	25,68	0,01	0		2	0,21
KW Once-only project	10,32	0,35	25	0,00	63	0,00
KW Package off-the-shelf						
KW Package with customization	5,94	0,05	3	0,13	5	0,16
KW Phased project (part of program)	4,26	0,18	16	0,05	30	0,04
KW Pilot; Proof of Concept	5,58	0,03	1	0,12	5	0,15
KW Release-based (one application)	7,32	0,41	72	0,00	29	0,00
KW Rules & Regulations driven	5,86	0,09	4	0,01	21	0,01
KW Security	10,94	0,04	1	0,05	10	0,02
KW Single-application	7,60	0,77	100	0,00	89	0,01
KW Steady heartbeat	7,77	0,14	35	0,00	4	0,00
KW Technology driven	9,41	0,16	11	0,01	33	0,00
ORG Organization A	7,20	0,59	68	0,07	84	0,03
ORG Organization B	6,41	0,36	45	0,05	38	0,02
ORG Organization C	7,57	0,06	2	0,02	9	0,12
PPL .NET	9,84	0,17	25	0,03	18	0,06
PPL 3GL	3,18	0,02	1	0,19	3	0,23
PPL COBOL	6,96	0,16	8	0,00	26	0,04
PPL FOCUS	8,98	0,01	4	0,06	0	
PPL JAVA	11,39	0,44	30	0,00	76	0,00
PPL MAESTRO	21,42	0,01	0		1	0,37

Business Domain	$\chi 2$	Base-rate	Number positive Good Practice	Probability Good Practice (p)	Number positive Bad Practice	Probability Bad Practice (p)
PPL ORACLE	0,38	0,08	25	0,00	2	0,00
PPL Other	7,57	0,03	5	0,10	2	0,20
PPL PowerBuilder	17,68	0,01	3	0,14	1	0,29
PPL PRISMA	8,82	0,01	1	0,36	1	0,34
PPL RPG	2,68	0,02	3	0,18	0	
PPL SQL	7,89	0,03	5	0,10	1	0,12
PPL Visual Basic	9,57	0,02	5	0,03	0	
SC Large Medium Project (401-600 FP)	7,14	0,09	7	0,07	9	0,08
SC Large Project (>601 FP)	8,26	0,09	11	0,12	9	0,09
SC Small Medium Project (201-400 FP)	7,20	0,19	23	0,09	26	0,09
SC Small Project (<200 FP)	6,26	0,62	74	0,07	87	0,04

TUD-SERG-2013-019
ISSN 1872-5392