

5-15-2012

MEASURING THE INFLUNENCE OF PROJECT CHARACTERISTICS ON OPTIMAL SOFTWARE PROJECT GRANULARITY

Moritz C. Weber
Goethe University Frankfurt

Carola Wondrak
Goethe University Frankfurt

Follow this and additional works at: <http://aisel.aisnet.org/ecis2012>

Recommended Citation

Weber, Moritz C. and Wondrak, Carola, "MEASURING THE INFLUNENCE OF PROJECT CHARACTERISTICS ON OPTIMAL SOFTWARE PROJECT GRANULARITY" (2012). *ECIS 2012 Proceedings*. Paper 66.
<http://aisel.aisnet.org/ecis2012/66>

This material is brought to you by the European Conference on Information Systems (ECIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ECIS 2012 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

MEASURING THE INFLUENCE OF PROJECT CHARACTERISTICS ON OPTIMAL SOFTWARE PROJECT GRANULARITY

Weber, Moritz Christian, Goethe University Frankfurt, Campus Westend, Grüneburgplatz 1, 60329 Frankfurt, Germany, moweber@wiwi.uni-frankfurt.de

Wondrak, Carola, Goethe University Frankfurt, Campus Westend, Grüneburgplatz 1, 60329 Frankfurt, Germany, cwondrak@wiwi.uni-frankfurt.de

Abstract

With a well-structured design of reusable software, it is possible to save costs in future software development cycles and generate added value. Nowadays it is difficult to quantify which grade of project structuring enhances its reusability. This is the question this paper aims to examine, based on the financial Portfolio Selection theory and Value at Risk methods that can be adapted to the field of software project structuring. Using these methods, the value of the reusable source code can be evaluated and the risks of reimplementation be compared with alternative code structures. So we can quantify and measure the effects of the software project structure on the risks of reimplementation. This model is applied to 27 Github open source software projects and enables us to investigate the influences of the project characteristics on the best possible software project granularity.

Keywords: Project Management, IT Project Portfolio, Software Granularity, Value at Risk.

1 Motivation

Static and monolithic information systems mark the beginning of enterprise IT infrastructures (Krafzig et al., 2004). With the emergence of spreading network structures, these inflexible infrastructures are split into subsystems and managed as interrelated parts of an IT portfolio (Erl, 2005). ‘The foundation of an IT portfolio is the firm’s information technology’ (Weill and Vitale, 2002). This is — among IT components and human infrastructures — defined by shared IT services and standard applications (Weill and Vitale, 2002). Not only software development, but also general strategy research reflects synergies and diversifications of portfolios. These allow the generation of additional positive economic values gained by the merging and restructuring of financial portfolios (Amihud and Lev, 1981; Christensen and Montgomery, 1981; Farjoun, 1998). As the realization of synergistic economies is crucial, it can be seen as a third pillar next to financial and vertical economics (Hill and Hoskisson, 1987). Markowitz’s Portfolio Selection theory (1952) defines a theoretical framework for portfolio diversification and synergies which can be extended (Myers 1982) towards strategic planning and has the potential to be applied in other research domains (Rockafellar and Uryasev, 1999). As IT projects are also aggregated in portfolios, this theory supports evidence as to how software portfolios should be structured.

Synergies typically depend on the size and structure of the portfolio (Eckbo, 1983; Bradley et al., 1983). Two kinds of synergies can be distinguished for IT portfolios (Cho and Shaw, 2009; Tanriverdi, 2005; 2006): On the one hand bundling two components together allows sub-additive cost savings within similar processes (Teece, 1980). On the other hand, two interrelated components can actually enable an additional, super-additive benefit at the same time as increasing the effort for both (Milgrom and Roberts, 1995). Economies of scope enhance the potential of those synergies (Teece, 1980; 1982) and the ability to diversify a portfolio (Willig, 1978). This gives reason to expect lower risk costs and higher generated added value (Panzar and Willig, 1981), also for IT project portfolios.

Particularly the risk stemming from size and the structure of a portfolio can be expressed in granularity. In the following sections granularity measurement is defined as ‘quantifying the contribution of [...] concentrations to portfolio risk’ (Lütkebohmert and Gordy, 2007). Unlike the monolithic enterprise IT infrastructures at the beginning, it has not yet been investigated how granular and diversified a software portfolio should be from an economical point of view (Friedl, 2011). It is said that two thirds of all software projects fail or are not finished within the fixed requirements or the given time. This is also due to organizational failures in software development (Standish, 2011). That is why we ask the following research questions:

- How granular should software project structures be?
- How can project-granularity levels be measured and compared?
- How can an optimum level of granularity be determined at present, so that good reusability and lower change risks can be expected?
- Which project characteristics have a critical influence on the optimum of granularity?

In the following section, we reflect the Portfolio Selection theory and the Value at Risk method (2.1) as theoretical framework to value portfolio risks. Related work on IT portfolio risk, project management (2.2) and granularity analysis (2.3) let us derive research hypotheses on risk costs, project characteristics and granularity from literature. To analyze these hypotheses, we adopted Portfolio Selection and Value at Risk to measure quantitatively risk costs in software project data (3.1 and 3.2). Alternative levels of project granularities are simulated by three methods (3.3) that are applied to software version control data for the 27 most observed Github (2011) open source projects. The Value at Risk is measured for each method and each simulated project granularity level. Accordingly, a regression setup is developed to explain the influences of project characteristics on optimal software project granularity (3.4). As a result alternative project granularities are selected to show and discuss

corresponding changes of risk costs (4.1, 4.2 and 4.3). Changes in the project characteristics are measured over time for all projects and each version. These changes are taken to explain their influence to the granularity level and the minimal risk costs of software projects (4.4). Finally we summarize our results, highlight limitations and provide an outlook on future research (5).

2 Related work

As it is often claimed that software projects are structured like portfolios with component assets, we align our methodology with the Portfolio Selection theory (Markowitz, 1952) and the risk measurement approach called Value at Risk (Wolke, 2008) in Section 2.1. This economic perspective is applied to IT project portfolio management and risk management to link the theory to the scope of information systems research (Section 2.2). It is often assumed that software development projects – as a subset of IT projects – have a high risk of failure (Standish, 2011). That is why we supplement our review of related work by recent studies on structuring of software development projects and analyzing the quality of software granularity (Section 2.3).

2.1 Portfolio Selection and Value at Risk

A software project contains several components that are linked with each other. This is similar to a financial portfolio where each portfolio component generates a value contribution. Since software can be a substantial long-term investment, it is important to select portfolio components in the most efficient way, so that a high level of reusability and lower maintenance costs can be anticipated.

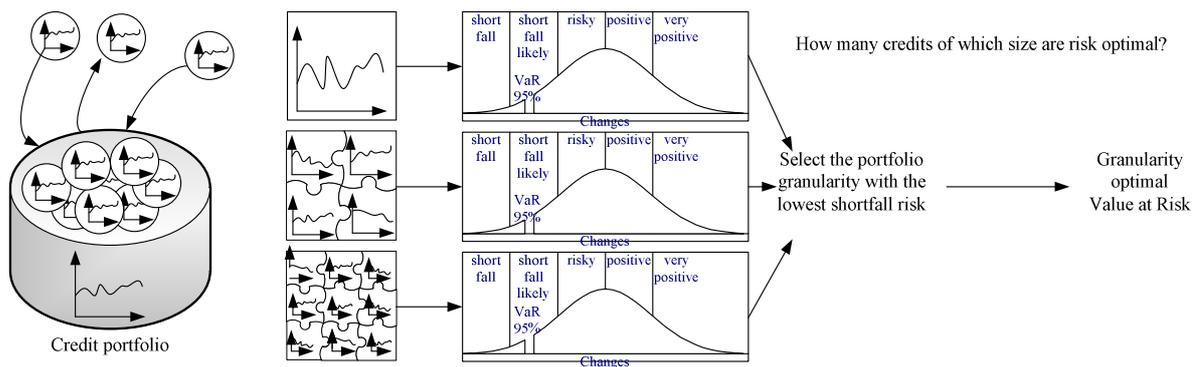


Figure 1. Portfolio diversification and risk in the context of correlation effects

A theoretic-based approach for the selection of components of a financial portfolio is the Portfolio Selection theory developed by Harry Markowitz (1952). It outlines how to structure an optimal portfolio with minimal risks by efficiently dividing it into partial investments (so called diversification). Thereby, interrelated influences between the components are taken into account and assembled in a single structure, so that the total risk of value depreciation is minimized (so called correlation). The value depreciation can be estimated using the Value at Risk (VaR) approach (Albrecht and Maurer, 2005) (see Figure 1). It ‘can be applied to any risk-factor model of portfolio [...] risk, and [is ...] a very general framework’ (Lütkebohmert and Gordy, 2007). Hereby, the maximum value depreciation is a value that will not be exceeded in a given time period with a given probability (Wilkens, 2001; Wolke, 2008). The VaR approach calculates not only the expected value of losses (Expected Loss), but also explicitly outlines the unexpected losses (Campbell et al., 2001). Like this, it enables us to include the extreme cases of value depreciation in our estimations.

This concept can be applied to any single evaluation, but also to multiple investment assets and entire investment portfolios. If there are multiple investments with considerable risks in one portfolio, additional synergy effects need to be taken into account (Wiedemann, 2002). Value fluctuations of these components could be both chances and risks for the whole portfolio. These risks can be

diversified by combining components which compensate the interrelated value depreciation risks. The goal is to find synergies that compensate the risks among themselves (Rockafellar and Uryasev, 1999).

Through systematic validation of different portfolio structures, it is possible to assess and value the risk of value depreciation using the VaR approach. In the case of inefficient combinations, the so called 'clustering risks' can emerge (Huschens and Stahl, 2004). This kind of insufficient granularisation leads to the danger of a higher, diversified VaR (Albrecht and Maurer, 2005). The Portfolio Selection theory allows evaluating the efficient selection of portfolio components, so that the risk costs of value depreciation will decline for the future period. Rockafellar and Uryasev (1999) explicitly suggest that the VaR concept should be applied as methodology in other areas. Outside of the financial research domain we adopt this for managing the portfolio risk of IT projects.

2.2 IT portfolio risk and project management

Cho and Shaw (2009) model and simulate the influences of synergies on the IT portfolio selection by applying the Markowitz theorem and the VaR approach to the research scope of IT portfolios. They find that not only portfolio returns, but also portfolio risks need to be taken into account. Moreover, they argue that only few studies investigated the influences of IT synergy on portfolio risks. Benaroch (2002; 2006) analyzes IT risks and their influence on large portfolios of IT investments. Strong evidence is found that IT risk management is largely driven by intuitions, which leads to suboptimal and counterproductive results. Tanriverdi and Ruefli (2004) state that synergies of IT projects also influence the risk of those. A failure of one IT portfolio component may cause failures in other parts of the portfolio, which may increase the risks at higher levels of diversification (Tanriverdi, 2005; 2006). This indicates that the following hypothesis on software project granularity should be rejectable:

Hypothesis H1₀: Risk costs have no crucial influence on optimal software project granularity.

The organization of software development project portfolios can be improved by a well coordinated project management. Lichter and Mandl-Striegnitz (1999) show that most software development projects fail because of inappropriate organizational settings and not because of the technical implementations. In addition, inefficient processes may cause design errors and an unexpected increase in project costs. A rule of thumb on how the cost of a single present design error influences future development and refactoring expenditures is given by so called 'Rule of Ten' (Pfeifer, 2001). It estimates that an undiscovered design error multiplies the required cost of rectification by ten for each following period. It is possible to recognize design errors earlier by using efficient project management and verifiable key figures (Lichter and Mandl-Striegnitz, 1999).

Especially the reuse of previously implemented software artefacts is an opportunity to reduce costs and work effort in multiperiodic software projects. Necessarily the design of software artefacts should not be too complex and problems with complex interdependence between the artefacts should be taken into account (Schirmer and Zimmermann, 2008). This involves both, dependencies between different parts of the software and a good kind of granularity. More specific, special attention must be paid to project characteristics, the structure of the portfolio and the selection of subprojects (Schirmer and Zimmermann, 2008). So we expect to reject the following null hypothesis:

Hypothesis H2₀: Project characteristics have no influence on optimal software project granularity.

2.3 Granularity analysis

Different approaches analyze optimal software project structures (Krafzig et al., 2004, p.18). Thereby, efforts are being made to quantify the quality of structures by measuring the utility and monetary value to software structuring decisions: Erradi et al. (2006) propose a quantitative ranking with numerical steps up to 10, which considers the level of dependencies between the project components. Furthermore, they criticize the lack of theory that would support the findings of correct granularity for maximal component reuse and lower project risks (Erradi et al., 2006). Alternative research strategies evaluate the outsourcing and awareness potential of distributed projects and identify project

structuring candidates by an economical analysis (Klose et al., 2007). In their empirical analysis Braunwarth and Friedl (2010) took the internal cost allocation of a company as a benchmark and simulate potential structures of granularity. In addition of this study, Friedl classified granularity approaches based on a literature review and gave advice for designs of good granularity on typical factors of influence in software development projects (Friedl, 2011). Katzmarik supports the granularity decision with a micro-economical approach, in which the project structures are modeled as ‘Software as a Service’-products (SaaS) and evaluated (Katzmarzik, 2011). Therefore, we assume that the following null hypothesis cannot be supported:

Hypothesis H3₀: Optimal software IT project granularity cannot be estimated empirically.

3 Methodology

In the following section the VaR concept is applied to software development projects by using version change data from software control systems. The software project structures as well as the historical variance and covariance of source code lengths are measured. Through three different methods alternative granularity structures are simulated and the level of minimal change expenditure risk is determined by VaR valuation. Finally, this minimal level is regressed against the given project characteristics to measure the influences of the project parameters on the optimal level of granularity.

3.1 Assumptions

In accordance with the Portfolio Selection theory (Markowitz, 1952) it is assumed that each portfolio can be diversified by a requirement-optimal or risk-minimal allocation, if a value can be determined. Modern software engineering approaches separate classes, services and other entities into single files (Erl, 2005). Therefore, we assume that each change of a file is an implicit change of a portfolio component. Agile software development practitioners (like in open source projects) claim that each change of a software project implements a single functional requirement or bug fix, which is documented in our case by the version control system. For simplification, it is assumed that the change expenditure of a file is highly correlated with the number of changed and unchanged lines of code (LOC). It is further assumed that the kind of development and refactoring is anchored in the project characteristics and that it stays similar in the history of a project. That is why historical change data is a good estimate of future change risks. The current size of a source code file is taken as present value. Using historical recorded project development costs, the LOC calculation can easily be transformed into a utility measure by applying COCOMO methodology (Katzmarzik, 2011, p. 23).

3.2 Data aggregation and measuring method

The data of software version control systems is used to measure direct development structures and characteristics of software projects. Thereby, the historical sizes of each file as well as realized software changes are measured. So the extent of each requirement implementation or bug fix can be precisely determined. Each requirement implementation can affect multiple file changes. Like Grigore and Rosenkranz (2011), Turek et al. (2010) and Wierzbicki et al. (2010) we measure the residual value of reusable source code: This is the number of document lines which are created by one author and remain in the document after the document revision (adds/removes). This residual value of reusable code is identified for each file of every requirement implementation (patch). This value is saved with the residual values of the corresponding files of each change (see Figure 2 ‘Data aggregation’).

The VaR concept is applied (Wiedemann, 2002) to identify at which structural level the residual value of reusable source code remains the highest possible. This determines the loss that is not exceeded in the covered period of time with a given probability. Thus the calculation also takes historical fluctuations and correlative dependencies between the components of a portfolio into account. Here, the fluctuations of the residual value of reusable code (determined in the data aggregation) are used. For all pairwise changed files, a covariance value is calculated and aggregated in a covariance matrix

(Wiedemann, 2002). The risk value is calculated by matrix multiplications of the file lengths at the current point in time and the variance-covariance matrix of relative residual values of reusable source code. Finally, the risk quantile for the given probability is determined (Wilkens, 2001; Wolke, 2008) (see Figure 2 'Measuring method').

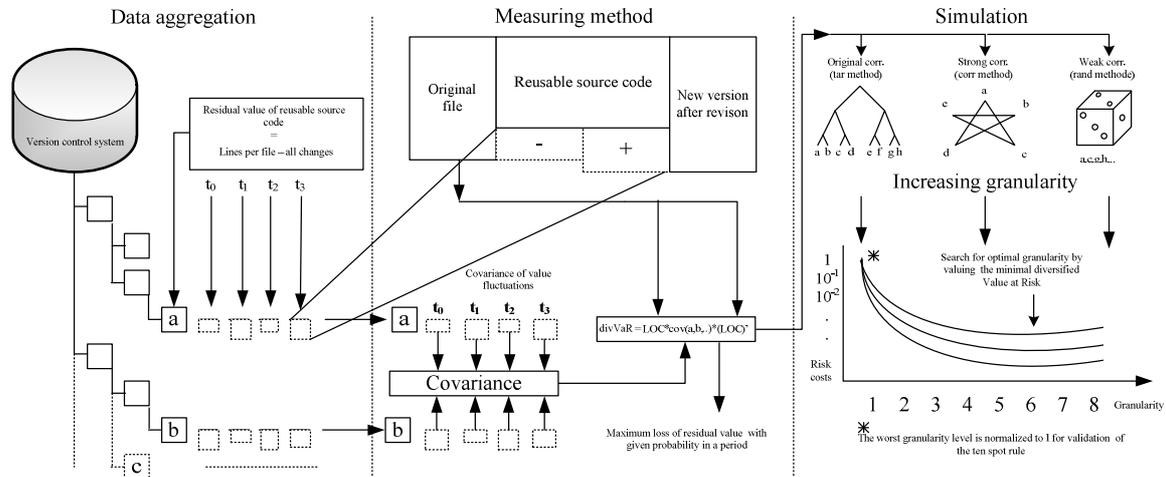


Figure 2. Data aggregation, measuring method and simulation of the analysis

3.3 Simulation

The granularity of every historic grown project is given implicitly by the project structure. Therefore, a diversified VaR can be calculated directly. It quantifies risk of reimplementation for the given granularity structure. To evaluate the risk of alternative granularity levels, variants with comparable structures must be derived. A sufficient effective method (hereinafter 'tar method') starts with the lowest folder level and aggregates all files per folder into a single file. Afterwards this file is moved to the superior folder (Lee et al., 2007). This way, granularity variants can be generated which keep the same basic structure. In a simplification of Lee's approach, the common tar algorithm is recursively applied (FSF, 2006). With source code packages and functional structuring of folders the dependency structures and conditional changes remain unaltered even for extensive aggregations. The most coarse and presumable worst granularity generated by this method, is to copy all source code into a single file. For each aggregation level, the size of files and changes as well as the covariance are remeasured and a risk value is calculated using the VaR approach.

The results of tar methodology are validated by methodological triangulation using two additional control methods. The first validation method ('corr method') selects an initial file from the project for each simulation run (Blobel and Lohrmann, 1998). Starting with this file, dependently changed files are recursively added. This is repeated until the number of files fits the amount of files of the corresponding granularity level of the tar method. The second method ('rand method') selects random files, but does not consider correlation structures. Thus, the first control method simulates subprojects of the entire project with a strong correlation. The second method simulates the omission of these correlations. Huschens and Stahl (2004) show that changes in the correlation structures influence the relative evaluation of granularities only slightly. The structure of the entire project is approximated by averaging the partial results (Blobel and Lohrmann, 1998).

The validation methods yield good comparative values. But it should be kept in mind that these methods are too aggressive and too coagulative in their simulation/estimation, due to their random selection and partial simulation. As a result, badly diversified and extreme project structures can be generated. This is why these two validation methods are used for triangulation reasons only and not for measuring the absolute risk levels of the relative residual value of the reusable source code. To facilitate comparison — also with the 'Rule of Ten' (Pfeifer, 2001) — we start at the granularity with

the highest VaR for each method. Subsequent all VaRs are normalized to this initial value of 1. The normalized value of 1 is usually the case, if just one file contains all source code (Katzmarzik, 2011, p.29) (see Figure 2 ‘Simulation’).

3.4 Regression setup

To investigate our research hypotheses and the influences of given project characteristics on the optimal project structure, we regress typical project key figures on the optimal project granularity of each release version (see Equation 1). As the time and the amount of version releases is not synchronized between the 27 projects, we build a cross-sectional dataset containing the interversion differences in optimal granularity (OptGranu) and the risk costs at optimal granularity (RiskCosts). Additionally typical project characteristics like size units (Katzmarzik, 2011, p.8) and time measures (Pfeifer, 2001) are added. These are number of patches (Patches), average file changes per patch (AvgPFiles), number of files (Files), total lines of code (LOC) and duration of the version release (Time) are added. To harmonize sizes of increasing and decreasing changes, all differences are logarithmized. Moreover, we added control variables (Controls) for each project to identify project specific influences.

$$OptGranu = \alpha + \beta RiskCosts + \gamma Patches + \delta AvgPFiles + \epsilon Files + \zeta LOC + \eta Time + \sum_{j=1}^n \theta_j Controls + \epsilon \quad (1)$$

The sample is tested for a typical structural bias. Considering the assumably close dependency of the exogenous variables, no multicollinearity is observed. Since heteroscedasticity of the residuals occurred in the original model, robust standard errors are applied.

4 Results

4.1 Descriptive results

Due to easy accessibility and publishability our data sample consists of top ranked Github open source projects. However, this setup can also be easily applied for commercial projects like SOA, SaaS or proprietary software. Github ranks the interest in projects by counting the number of observers following the project and the number of projects that are build upon this project. We selected 27 projects and ensured that the sample consists of equally distributed programming paradigm styles (scripting, imperative/functional and object-oriented), execution styles (native, byte code, interpreted) and software category (webserver/ -framework, database/ libraries, system/ languages). The occurrences of each language, the type and paradigms of projects within this sample can be seen in Table 1:

Exec	native	bytecode	interpet	Lang	c	c++	java	objc	erlang	python	ruby	js
27	9	9	9	27	6	3	3	3	3	2	2	5
Categ	web	db/lib	sys/lan	Type	sys	lang	db	lib	webserver	webframe		
27	9	9	9	27	6	3	7	2	2	7		
Style	oo	script	im/fun	Parad	oo	script	imp	func				
27	9	9	9	27	9	9	6	3				

Table 1. Descriptive statistics of the selected sample of projects

Overall, we extracted 1625 observations (version tags). Some projects like ‘postgres’ have a long version history, others like ‘nu’ displayed a history of continuous integration into a single master version. As we want to observe the influence of project characteristics on the optimal project granularity, we accept also these extreme project settings. Project-specific influences are tested with the aid of control variables in the final regression. Descriptive figures are listed in Table 2 and 3.

Name	Cassandra	Clojure	CouchDB	Django	Ejabberd	Git	Gitx	Httpd	jQuery	Memcached	MongoDB	NodeJS	Nu	OTP
Language	java	java	erl	python	erl	c	objc	c	js	c	c++	c++	c	erl
Type	db	lang	db	webfr	sys	sys	sys	webse	webfr	db	db	sys	lang	lib
Paradigm	oo	oo	func	script	func	imp	oo	imp	script	imp	oo	oo	imp	func
Versions	54	3	5	23	28	296	11	174	63	30	104	105	1	6
Fold. Avg.	204	39	70	3152	36	178	87	187	32	23	83	359	103	2109
Files Avg.	870	250	512	4087	336	2242	330	2905	155	135	1703	4968	379	9754

Table 2. Overview of project characteristics (part1)

Due to page limitations and a better understanding of the calculation results, we highlight the findings for the popular web framework (Ruby on) Rails as an example. It shows average figures in its project statistics and the calculation yields similar patterns and results for the rest of 26 projects. In the final regression all 1625 observations are taken as single events. However, for clarity we aggregate all the change data of the 116 versions in Rails and calculated the all-time VaR for each granularity level.

Name	Phone-gap	Postgres	Prototype	Rails	Redis	Ruby	Scriptulous	Solr	Sparke	Three20	Tornado	V8	YUI3
Language	js	c	js	ruby	c	ruby	js	java	objc	objc	python	c++	js
Type	webfr	db	webfr	webfr	db	lang	webfr	db	sys	lib	webse	sys	webfr
Paradigm	script	imp	script	script	imp	script	script	oo	oo	oo	script	oo	script
Versions	5	260	11	116	61	73	3	6	4	6	9	157	22
Folders Avg.	76	380	47	777	45	459	25	1392	49	205	57	56	1587
Files Avg.	113	3826	130	2055	275	3460	111	4647	297	1159	185	1646	4358

Table 3. Overview of project characteristics (part2)

Figure 3 illustrates that starting with a normalized VaR of 1, the VaR of all three methods declines with an increase in granularity levels. As expected, the control methods (rand and corr) calculate structures with higher VaRs. The tar method shows a similar slope to the ‘Rule of Ten’, but at the granularity level of 5 the VaRs stabilize at a stationary level. All three methods indicate the optimal VaR at a granularity level of 7. Even though the ‘Rule of Ten’ does not depend on the project characteristics of Rails, it approximates the tar graph to the granularity of 5 in the stationary region. Consequently, the control methods cannot predict correct risk cost but they estimate the same optimal granularity level as the tar method. The ‘Rule of Ten’ cannot predict the level of granularity correctly, yet it indicates a good benchmark for the risk costs predicted by the VaR.

4.2 Rule of Ten as cost measure for incorrect granularity

In general, the slope of the graphs shows that the theoretical ‘Rule of Ten’ underestimates the risk costs, with increasing levels of granularity. Additionally, it can be easily observed that the ‘Rule of Ten’ approximates the risk cost value of the tar method best towards the optimum. Simultaneously, the project shows that this rule of thumb noticeably underestimates the risk costs between the levels of the worst and optimal granularity. This is especially the case when multiple levels of granularity have measurable similar risk costs. Moreover, it can be argued that the validation methods simulate more aggressively, coarsely and result in worse diversified portfolios than the ‘Rule of Ten’ and the tar method. It is in line with the results of other empirical studies that even the randomized validation methods confirm the minimum risk costs found by the tar method (Huschens and Stahl, 2004). Comparing with the ‘Rule of Ten’, it can be argued that the tar method estimated the risk costs best,

but also the validation methods determine the same level of granularity as optimal, when the absolute risk cost values are negligible.

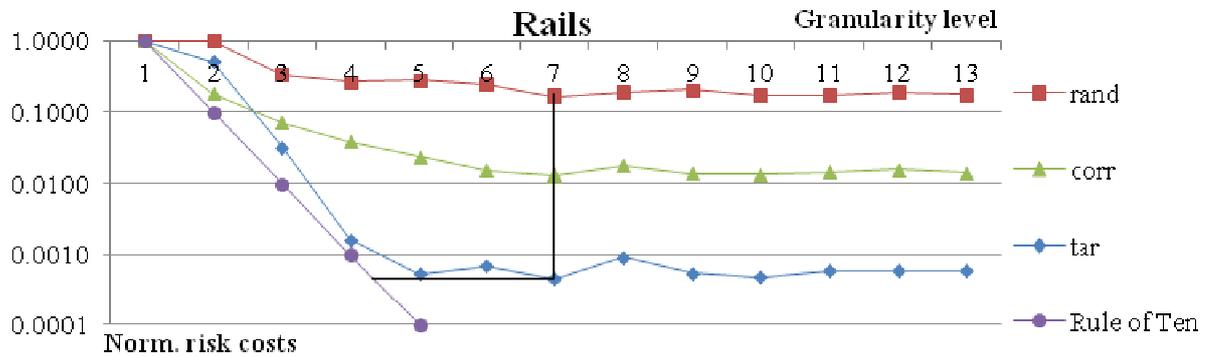


Figure 3. Analysis of Rails project results

4.3 Marginal utility of granularity

The data gives insights into the risk cost decline with an increase in granularity and subsequent stabilization at a stationary level. It can be intuitively understood that projects are structured with best practice methodologies and iterative approaches (Erl, 2005) until a manageable software foundation is formed. Coevally, it can be seen at the stationary levels of the project that more or less adequate level of granularity can already be discovered at lower levels of granularity. The marginal utility of additional granularity towards the optimum is declining due to complexity of coordination between the components. This can be explained by the effect of overspecification which is mentioned in literature (Millard et al., 2009). If software structures are initially designed for good reusability, but then never reused, then they might result in an inefficient overspecification and overhead. The measurement of all projects shows, that a slight over- and underspecification can be tolerable. On the one hand the risk cost increase slightly around the area of optimal granularity, while this increase is much smaller than at lower or even monolithical granularity.

4.4 Influence of project characteristics on optimal granularity

Figure 3 illustrates that the implicit ‘Rule of Ten’ assumption of a log-log dependency of risk costs and granularity is too unprecise. That is why we regress all 1625 version observations of the 27 projects as a cross-sectional dataset of first differences of log and investigate influences of additional project characteristics. Results of the regression analysis explain changes of risk costs at optimal granularity. The number of patches, files per patch and total number of files have a highly significant influence on the change of optimal granularity. Even the influence of LOC changes is significant at least at a 5% level and time at a 10% level of significance. While risk costs, patches, files and LOC changes have a positive impact on the changes in level of optimal granularity, changes of files per patch and time have a negative influence. As all project-specific control variables do not show any significance, they are removed from the regression equation and the final results (see Table 4).

All measured project characteristics are significant and allow us to reject null hypothesis H_{2_0} that there are no influences by the project characteristics. As the impact of changes in risk costs shows the highest t-value null hypothesis H_{1_0} is rejected as well. The whole regression model has an adjusted R-squared value of 54.05% for the analyzed cross-sectional dataset. Applying an F-test for the significant influence of all coefficients of the model yields an F-Value of 273.62, which is highly significant for 1625 observations and enables us to reject also null hypothesis H_{3_0} . From a project management point of view, this gives evidence that increases in project parameters like project risk costs, number of patches, files and total lines of code cause a need of more granular structuring. Risk cost influence has the highest significance of the measured project characteristics. If the number of files per patch increases in a current version, it indicates too granular project structures and results in a need for a less

granular structuring. A shortage of release cycles and version snapshots shows similar results: if the time between two releases decreases, the need for more granular and more diversified project structures increases.

	Const	Risk Costs	Patches	Files/Patch	Files	LOC	Time
Coefficient	-0.0118	0.0229	0.0885	-0.3272	1.0451	0.1976	-6.5603
Std. Error	0.0135	0.0011	0.0083	0.0397	0.3231	0.0775	3.9047
t-Value	-0.8728	20.8817	10.6532	-8.2349	3.2348	2.5486	-1.6801
P-Value	0.3829	<0.00001	<0.00001	<0.00001	0.0012	0.0109	0.0931
		***	***	***	***	**	*

//** indicate significance at the 10%/5%/1% level. n = 1625; adjusted R² = 0.5405;*

F(6, 1618) = 273.6164; P-Value(F) < 0.00001

Table 4. Regression results of optimal granularity level and the project specifically influences

5 Conclusion

Four research questions regarding the structuring of software projects have been defined. They deal with: the structure of software projects, the measurability and comparability of granularity levels, the optimal granularity, so that good reusability and low change risks can be expected, and finally the influences of project characteristics. In this paper, the Portfolio Selection theory and the VaR concept were applied to software project portfolios and extended with the aid of project risk costs to measure directly the granularity level of software repositories. Thereby, the risk of reimplementation is measured by evaluating the relative residual value of reusable source code. The selection of the correct software project structure with efficient granularity becomes measurable by valuing the risk costs.

The sizes of software components and historical software changes have been measured with the aid of software version control systems for empirical quantifiability. A sample of the 27 most active Github open source projects was simulated in various granularities generated with three different methodical approaches and accordingly rated concerning the risk of reimplementation. The hypotheses derived from theory and related work were tested to explain which project determinants influence the level of granularity. Furthermore, a model was developed to determine how to adjust granularity for given changes in project characteristics.

The analysis shows that despite differences in the simulation methods, all of them value the same level of granularity as optimal, although the control methods are not equally reliable about the absolute level of risk cost. Coevally, the 'Rule of Ten' can be confirmed empirically, even when the logarithmic structure of this rule reflects the specific characteristics only partially correct. Furthermore, it can be shown that there is a marginal utility of granularisation and that the utility around the optimal level of granularity only declines very slightly. Empirical evidence has been provided that risk costs have the highest influence on the granularity decision. It was further shown how additional project characteristics influence a systematical estimation of optimal software granularity. Researchers get a well-established theory, flexible methodology and applicable framework to finding of correct granularity for good component reuse and lower project risks as claimed in Erradi et al. (2006). This enables design science support as well as positivistic evidence for research streams like service granularity and project risk management. In practice project managers receive a toolset to predict the consequence of changes in project parameters and find advise how granularity adjustments can decline possible risk cost of this changes. Software architects and developer get decision support for selecting the best choice from multiple implementation opportunities or might apply this methodology in search filters of registries to find service and software that let expect less risk cost with future changes.

The presented method is an approach to quantify the quality of software project structures. This approach is limited to project risks and neglects possible returns that are more difficult to quantify

from project source code data. The deducted model has an explanatory power of 54%, which means 46% of the observed influences are caused by influence that cannot be identified with the facts from data set. The data set contains only open source projects. The measurement methods are just able to simulate the granularity between 1 and the given project granularity. Even with the deducted model an initial project version snapshot is needed to determine a good level of granularisation. In further research, we will address these limitations and deepen our analysis of the relation between risk costs and optimal granularity as these have the highest significance in our regression model. Depending on data accessibility and publishability we plan to shift our data scope from open source to SOA, SaaS and proprietary software projects to support structuring decisions in commercial software projects. We also want to investigate the dynamics of granularity over time to understand how previous versions influence the current optimal granularity.

Finally, concerning the research questions, it has been shown that the optimal granularity structure of software projects, can be measured at present. A method has been presented that enables us to measure and compare the quality of software project granularities and the influence on the optimal level of granularity. Consequently, it becomes possible to estimate quantitatively how granular software projects should be structured and how this granularity must be adjusted when specific project parameters change. We thankfully acknowledge the support of Martin Haferkorn and the E-Finance Lab, Frankfurt for this work.

Literature

- Albrecht, P. and Maurer, R. (2005). *Investment- und Risikomanagement: Modelle, Methoden, Anwendungen*. Schäffer-Poeschel, 2. Edition.
- Amihud, Y. and Lev, B. (1981). Risk reduction as managerial motive for conglomerate mergers. *Bell Journal of Economics* (12), pp. 605-617.
- Benaroch, M. (2002). Managing information technology investment risk: A real options perspective. *Journal of management information systems*, (19:2), pp. 43–84.
- Benaroch, M., Lichtenstein, Y. and Robinson, K. (2006). Real options in information technology risk management: an empirical validation of risk-option relationships. *MISQ*, (30:4), pp. 827 – 864.
- Blobel, V. and Lohrmann, E. (1998). *Statistische und num. Methoden der Datenanalyse*; Teubner.
- Bradley, M., Desai, A. and Kim, E.H. (1983). The rationale behind interfirm tender offers: information or synergy. *Journal of Financial Economics* (11), pp. 182-206.
- Braunwarth, K. and Friedl, B. (2010). Towards a financially optimal design of IT services. *ICIS 2010 Proceedings*, Saint Louis.
- Campbell, R., Huisman, R. and Koedijk, K. (2001). Optimal portfolio selection in a Value-at-Risk framework. *Journal of Banking & Finance* 25.
- Christensen, H.A. and Montgomery, C.A. (1981). Corporate economic performance: Diversification strategy versus market structure. *Strategic Management Journal* (2:4), pp. 327-343.
- Cho, W. and Shaw, M. J. (2009). Does IT Synergy Matter in IT Portfolio Selection? *ICIS 2009 Proceedings*. Paper 160.
- Eckbo, B. E. (1983). Horizontal mergers, collusion, and stockholder wealth. *Journal of Financial Economics* (11), pp. 241-274.
- Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology & Design*, Prentice Hall.
- Erradi, A., Anand, S. and Kulkarni, N. (2006). *SOAF: An Architectural Framework for Service Definition and Realization*.
- Farjoun, M. (1998). The independent and joint effects of the skill and physical bases of relatedness. *Strategic Management Journal* (19:7), pp. 611-630.
- FSF (2006). Free Software Foundation - TAR. Accessed August 6th 2011, <http://www.gnu.org/software/tar/tar.html>.
- Friedl, B. (2011). Zur optimalen Granularität von IT-Services - Eine Analyse relevanter ökonomischer Einflussfaktoren, 10. Int. Tagung Wirtschaftsinformatik, Zürich.
- GitHub (2011). Github Social Coding. Accessed November 9th 2011, <https://github.com/repositories>.

- Grigore, M. and Rosenkranz, C. (2011). Increasing the Willingness to Collaborate Online: Analysis of Sentiment-Driven Interactions in Peer Content Production. ICIS 2011 Proceedings.
- Hill, C.W.L. and Hoskisson, R.E. (1987). Strategy and structure in the multiproduct firm. *Academy of Management Review*, (12:2), pp. 331-341.
- Huschens, S. and Stahl, G. (2004). Granularität dominiert Korrelation, *Risknews* 06/04.
- Katzmarzik (2011). Product differentiation for SaaS-Providers, *BISE*, 01/2011, pp. 19-31.
- Klose, K., Knackstedt, R. and Beverungen, D. (2007). Identification of Services - A Stakeholder-based approach to SOA development and its application in the area of production planning, *ECIS*.
- Krafzig, D., Banke, K. and Slama, D. (2004). *Enterprise SOA: Service Oriented Architecture Best Practices*, Prentice Hall International.
- Lee, J., Seo, Y. and Lee, J. (2007). Method and apparatus for merging data objects, *European Patent EP181840A2*.
- Lichter, H. and Mandl-Striegnitz, P. (1999). Defizite im Software- Projektmanagement - Erfahrungen aus einer industriellen Studie, *Informatique* 5.
- Lütkebohmert, E. and Gordy, M. (2007). Granularity adjustment for Basel II. No 2007,01, Discussion Paper Series 2: Banking and Financial Studies, Deutsche Bundesbank, Research Centre.
- Markowitz, H. (1952). Portfolio Selection: *Journal of Finance*, p. 77-91.
- Millard, D. E., Howard, Y., Abbas, N., Davis, H. C., Gilbert, L., Wills, G. B. and Walters, R. J. (2009). Pragmatic web service design: An agile approach with the service responsibility and interaction design method. *Computer Science*.
- Milgrom P. and Roberts, J. (1995). Complementarities and fit Strategy, structure, and organizational change in manufacturing. *Journal of Accounting and Economics* (19), pp. 179 -208.
- Myers, S.C. (1984). Finance theory and financial strategy. *Interfaces* (4), pp. 26-37.
- Pfeifer, T. (2001). *Qualitätsmanagement: Strategien, Methoden, Techniken*, Hanser Fachb., 3. Aufl.
- Rockafellar, R. and Uryasev, S. (1999). Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, Volume 26, Issue 7, Pages 1443-1471
- Schirmer, I. and Zimmermann, K. (2008). *Visualisierung von Projektportfolios zur Unterstützung des Architekturmanagements*, Universität Hamburg.
- Standish (2011). *The Standish Group CHAOS Manifesto 2011*. Accessed June 9, 2011, http://standishgroup.com/newsroom/chaos_manifesto_2011.php.
- Panzar, J. C. and Willig, R. D. (1981). Economies of scope. *American Eco. Rev.*, (71:2), pp. 268-272.
- Tanriverdi, H. and Ruefli, T. W. (2004). The Role of Information Technology in Risk/Return Relations of Firms. *Journal of the Association for IS*, Vol. 5, No. 11-12, pp. 421-447.
- Tanriverdi, H. (2005). Information Technology Relatedness Knowledge Management Capability, and Performance of Multibusiness Firms. *MIS Quarterly* (29:2), pp. 311-334.
- Tanriverdi, H. (2006) Performance Effects of Information Technology Synergies in Multibusiness Firms. *MIS Quarterly* (30:1), pp. 57-77.
- Teece, D.J. (1980). Economies of scope & the scope of the enterprise. *Journal of Economic Behavior and Organization* (1:3), pp. 223-247.
- Teece, D. J. (1982). Towards an economic theory of the multiproduct firm. *Journal of Economic Behavior and Organization* (3), pp. 39-63.
- Turek, P., Wierzbicki, A. and Nielek, R. (2010). WikiTeams: Evaluating Teamwork in Wikipedia. In: *Proceedings of the WebSci10*.
- Weill, P. and Vitale, M. (2002). What IT infrastructure capabilities are needed to implement e-business models? *MIS Quarterly*, (1:1), pp.17-34.
- Wiedemann, A. (2002). Messung von Zinsrisiken mit dem Value-at-Risk-Konzept, *WISU* 12, pp. 1548-1553.
- Wilkens, M. (2001). Basel II- Berücksichtigung von Diversifikationseffekten im Kreditportfolio durch das Granularity Adjustment, *Zeitschrift für das gesamte Kreditwesen*, 12; pp. 670-676.
- Willig, R. (1978). Technology and market structure. *American Economic Review* (69), pp. 346-351.
- Wolke, T. (2008). *Risikomanagement*; Oldenbourg Wissenschaftsverlag.
- Wierzbicki, A., Turek, P. and Nielek, R. (2010). Learning about team collaboration from Wikipedia edit history. *International Symposium on Wikis and Open Collaboration*.