Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

BACHELOR THESIS
SEBASTIAN DUDA

# A METHOD TO DETERMINE THE RETURN ON INVESTEMENT OF INNER SOURCE

Submitted on 25 September 2017

Supervisors:
Prof. Dr. Dirk Riehle, M.B.A. and Maximilian Capraro
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg

# Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

---

Erlangen, 25 September 2017

# License



---

Erlangen, 25 September 2017

# Abstract

Inner source is the use of open source software developing practices in proprietary software development across organizational borders. A significant amount of companies are adopting inner source. Some companies already utilize inner source practices without a coordinated effort to adopt inner source.

It is unclear if the gains of inner source outweight the costs of running and adopting it because there is no quantitative method to determine the return on investment for inner source yet. In this paper, we develop a quantitative method to determine the return on investment of inner source.

We followed a four phase research approach: First, we conducted a methodological literature review to collect methods and best practices on how to create a return on investment model. Second, we performed a exploratory literature review to identify typical inner source costs and gains. Third, we hypothesized formulas to quantify the costs and gains. Fourth, we prepared but not fully conducted an industry case study to evaluate the method.

We provide a methode on how to determine the gains and costs induced by inner source and on how to aggregate them to the return on investment value. We evaluated our method at an organization already adopting inner source. As the results were inconclusive, we suggest further research on evaluating the method. This paper contains the first method to determine the return on investment for inner source.

# Contents


**1 Thesis** **1**

1.1 Introduction . . . 1
1.2 Related Work . . . 2
1.2.1 Inner Source . . . 2
1.2.2 Return on Investment . . . 3
1.3 Research Approach . . . 4
1.3.1 Analysis of Existing Return on Investment Models . . . 5
1.3.2 Identification of Gains and Costs of Inner Source . . . 5
1.3.3 Quantifying the Gains and Costs . . . 5
1.3.4 Evaluation of the Method . . . 6
1.4 Return on Investment . . . 6
1.5 The Qualitative Model . . . 7
1.5.1 Gains . . . 7
1.5.2 Costs . . . 10
1.6 The Quantitative Model . . . 11
1.6.1 Quantifying Quality Improvements . . . 12
1.6.2 Quantifying Cost Savings . . . 15
1.6.3 Quantifying Speed Improvements . . . 20
1.6.4 Quantifying Adoption Costs . . . 21
1.6.5 Summary of Required Values . . . 24
1.7 The Method to Determine the Return on Investment of Inner Source 25
1.8 Validation and Limitations . . . 28
1.8.1 Validation of the Qualitative Model . . . 28
1.8.2 Validation of the Quantitative Method . . . 29
1.8.3 Limitations . . . 29
1.9 Future Work . . . 30
1.10 Conclusion . . . 30

**2 Elaboration** **31**

2.1 Thesis Goals . . . 31
2.1.1 Original Thesis Goals . . . 31
2.1.2 Changes to Thesis Goals . . . 31

2.2 Model Developed in Previous Research (NYT) . . . . . 31
2.2.1 Column Names . . . . . 38
2.2.2 Completion of the Preliminary Model . . . . . 38
2.2.3 Deduplication . . . . . 38
2.2.4 Omit Entities . . . . . 38
2.3 Decision Making . . . . . 41
2.3.1 Gitlab Plan . . . . . 41
2.3.2 Software Complexity Measurements . . . . . 42
2.3.3 Better Global Perspective or Simpler Code . . . . . 42

**Appendices** **43**
Appendix A Abbreviations . . . . . 43
Appendix B List of Tables . . . . . 44

**References** **45**

# 1 Thesis

## 1.1 Introduction

The endless need to improve the software products and services forces companies to continuously advance their software development practices. Today, the practice of choice to do so, is to operate with product line engineering best practices.
Inner source (IS) can be the next step in software development approaches. The main advantages of IS are the high reuse of code and the improved detection of bugs. The problem many companies face when considering the adoption of IS is a lack of methods to determine the profitability of IS.
The Return on Investment (RoI) is an economical indicator, stating the ratio of the capital spent and the profit the investment induced. In our case, the profit is equivalent to the money saved due to the measures of IS and the investment ist equivalent to the costs of IS. There is neither a method, nor a model to estimate and determine the RoI of IS yet. Without a method describing how to determine the RoI of IS many companies back off from adopting IS. Thus, many of them miss the chance to improve their software development practices.

The overarching question is whether applying and maintaining IS is lucrative. With this paper we take a first step in the direction with our research question: How to determine the RoI of IS?

In detail, this paper contributs the first method to determine the RoI of IS:

- A qualitative model of the gains and costs of IS
- Formulars to quantify the gains and costs
- A method how to use the quantitative model
- An evaluation of the method

To gather the required data we conduced three independent literature reviews. We analyze the gains and costs and compare them to determine the RoI of IS. This method is as general as possible in order to make it adoptable for many corporations considering the adopting of IS.

The paper is structured as follows: In section two, we discuss the related work while section three states the research approach of this paper. Section four analyzes existing RoI models and in section five we conduct a literature review to collect the gains and costs of IS. In the section six, we present forumulars to calculate each gain and cost. Section seven describes the method on how to determine the RoI. In section eight we validate the method. Section nine contains the future work on the topic of RoI of IS. The conclusion of the paper is in section ten.

## 1.2 Related Work

At first, we analyze the state of the research about IS topics with a special focus on literature addressing gains/benefits and costs/risks. Furthermore, we sight papers about RoI. We consider papers approaching best practices, models and methods for open source (OS) and code reuse.

### 1.2.1 Inner Source

> "The majority of definitions in literature [...] are akin to this first definition published by Dinkelacker et al. [2002] and share two characteristics:
> (1) IS leverages practices from open source development.
> (2) Contrary to open source, only a limited group of developers (employees of a specific organization) can take part in the community."
>
> Capraro and Riehle, 2016

The two requirements of IS shared by the majority of all defintions are required by the gains and costs identified below.

Stol, Avgeriou, Babar, Lucas and Fitzgerald, 2014 discussed nine key factors for adopting IS that need to be considered when implementing IS in order to prevent the major obstacles. In a second paper, Stol and Fitzgerald, 2014 describe a "tutorial" on how to adopt IS. Riehle, 2016 pictures an exemplary governance structure.
These papers analyze a theoretical approach on how to adopt IS in a company. We use said roadmaps to analyze the challenges and investments which have to be faced. These challenges and investments are condensed in the costs of the adoption of IS. In contrary, the method in our paper addresses already running IS programs.

Opposed to this the following papers describe the adoption of IS in a company. The introduction of IS at Phillips Healthcare was monitored by Wesselius, 2008. He expounded the development of the IS culture in the company from an economic perspective. Later Riehle et al., 2009 introduced IS to SAP. He described how SAP organizes the sharing of code. Lindman, Rossi and Marttiin, 2010 analyzed the changes a company is facing while adopting IS. In further research, Lindman, Riepula, Rossi and Marttiin, 2013 show two approaches with a differing focus.
On the contrary, we do not analyze the adoption of IS in companies. We determine the successfulness of the IS program by measuring the RoI.

Höst, Stol and Orucevic-Alagic, 2014 analyzed the characteristics of IS the management is facing. They provide a small list, which motivates to adopt IS as well. In the paper "Inner Source Definition, Benefits, and Challenges" Capraro and Riehle, 2016, analyze the state of the research of IS. They provide a summary of the benefits and challenges of IS. These benefits and challenges have a causal relationship with the gains and costs of IS. Riehle, Capraro, Kips and Horn, 2016 further researched how IS affects platform-based product engineering. The case studies presented in the paper show additional gains and costs. Gaughan, Fitzgerald and Shaikh, 2009 published a paper comparing IS with OS. They state serveral benefits of IS reported by companies which already implemented IS. Vitharana, King and Chapman, 2010 analyzed one of the main gains of IS, namely the reuse.

The papers contain nearly all gains and costs of IS (legal costs are missing). Nevertheless the focus of said papers are not the gains and costs, thus they don't provide all of them. We completed and quantified the list of gains and costs.

### 1.2.2 Return on Investment

The RoI or equity return is one of the main economic key indicators (Baum, 2013). The RoI is frequently used to measure the effectiveness of education (Hansen, 1963; Elaug, 1965; Kirkpatrick, 1998; Plaza, 2006; Pine and Tart, 2007; Phillips and Schirmer, 2008; Phillips, 2012). Pine and Tart, 2007 use the term "benefits" and "challenges" which are related with gains and costs. Thus, we use the terms as well in the exploratory literature review in section five. In the IBM developerWorks O'Neill, 2009 brings the RoI into an IT context. He claims that the IT became a main part of many companies' infrastructures so that an improvement of the RoI in the IT leads to a competitive advantage.

### Best Practices

Emam, 2003 exemplary calculates the RoI for several computer science topics like risk assessment and automated defect detection. In the paper, the models are not derived from a standard RoI model but from scratch. Thus, the methodology is not adoptable to our paper.
Sonnenreich, Albanese and Stout, 2006 adopt the classical RoI model to the context of security. The model is called return on investment for a security investment (ROSI). The gain of the RoI model is substituted with the security gain.

$$ROSI = \frac{(risk\ exposure * \%\ risk\ mitigated) - solution\ cost}{solution\ cost}$$

The paper shows how the classical RoI model can be altered. In our model, we will alter it from a single investment to a continuous investment.
Lee et al., 2012 describes the RoI model used by the Washington State Institute for Public Policy. In this model, the gains are substituted with a non-monetary benefit like "public health". The paper describes the best practices defined by the institute to grant a consistent quality of the results.

### Models and Methods

**Open Source** J. M. Pearce, 2015 developed a method to determine the RoI of OS hardware. The model for hardware is quite similar despite the fact that software has no reproduction costs. Furthermore, the RoI model does not analyze only a single return but a return over a longer time. Nevertheless, the investment is only done once in contrast to our model.

**Code Reuse** Emam, 2003 exemplary calculated the RoI for code reuse. This model includes the costs needed to search for reusable code. In our model, we have no costs like this. There is no effort to find reusable code, this is an effect of IS.
Poulin and Caruso, 1993 developed a metric to estimate the financial benefit of an organizational reuse program. The authors took a look at the reuse of code.

## 1.3 Research Approach

We followed a four step approach to create a method to determine the RoI of IS. First, we conducted a literature review regarding existing RoI models and identified model presented by Schmalen and Pechtl, 2013 as a good fit for us. Second, we searched for gains and costs of IS in the existing literature. The

gathered gains and costs are applied to the RoI model from step one. Third, we provided formulas to quantify the gains and costs detected in step two. These formulars are taken from the available literature or developed by the authors. Furthermore, we described a method on how to use the model. The previously developed models are a component of the method. Fourth, we took a first step towards the validation of the method. We evaluated the method with one case. We conducted the research in this kind of modular approach to easily extend or substitute single modules. If the validation of the qualitative model would have failed, we could easily substitute this part of the research work.

### 1.3.1 Analysis of Existing Return on Investment Models

We analyzed literature about the RoI in general, best practices on how to create such models and available models handling similar topics like open source, open source software, code reuse and other software development techniques. We did this by conducting a methodological literature review (M. Pearce, 2017). In this review, we analyzed the existing literature addressing the RoI. We extracted the methodology depicting how to create a RoI model from the papers. We used the papers as best practices. We expected to find a method or best practices to create a new RoI model. We did not find the one method on how to create a new RoI model, but instead we came across methods on how to build a model from scratch and how to derive existing models. For our model, we will reuse the standard model proposed by Schmalen and Pechtl, 2013 and alter it to suit our needs.

### 1.3.2 Identification of Gains and Costs of Inner Source

To collect the gains and costs of IS we conducted an exploratory literature review (Liston, 2006) by searching for literature by common terms. We used the terms: "*inner source*" combined with one or more of these words: "*gains*", "*chances*", "*benefits*", "*costs*", "*risks*". We collected all gains and costs mentioned in the found 14 papers. We applied the 16 gains and six costs of IS found in literature to the RoI model from step one.

### 1.3.3 Quantifying the Gains and Costs

We continued with conducting a focused literature review (Liston, 2006) to find ways to measure the found gains and costs. To quantify the gains and costs, we looked into literature for forumulars to calculate each gain and each cost. We expected to find a method to quantify each gain and cost. Instead we found that

the current literature does not provide methods to quantify most of the gains and costs. The IS literature provided formulas to quantify three gains and zero costs. However, for ten gains and four costs, the literature did not provide any hints. Consequently, for the remaining gains and costs we propose theoretical formulas ourselves. There are some gains and costs (e.g. G2 or C4) which were not quantified at all.

### 1.3.4 Evaluation of the Method

To evaluate the model we performed case study research at an organization currently adopting IS. However, the data collection is not yet finished. Our preliminary results are inconclusive about the utility of the model. In a first step, we validated our qualitative model by comparing our model with their interests. The second step was to apply the data provided by the company to our method. Due to the bad quality of the data supplied by the company the validation was inconclusive.

## 1.4 Return on Investment

The RoI or equity return is one of the main economic key indicators (Baum, 2013). It states the ratio of an investment and the gain caused by the investment. A high RoI indicates a lucrative investment. Thus, there is plenty literature about the RoI. Schmalen and Pechtl, 2013 presented the following RoI model:

$$RoI = \frac{income}{costs} = \frac{gains - costs}{costs}$$

The RoI model only requires two values: the gains and costs.
The classic RoI model uses the cost of one single investment and the gain of one single RoI. When adopting IS, there is not only one investment and one single payday. The costs have to be covered on a monthly basis and the gains are on a continuous base as well. Furthermore, there is no correlation between the gains and the costs. Higher expenses do not lead to higher gains. Thus, we have to calculate the RoI not only for one investment, but over a time span.

$$RoI(time\ span) = \frac{gains(time\ span) - costs(time\ span)}{costs(time\ span)}$$

The gains and costs analyze the difference between the software development without IS and with IS which is said time span from above. The difference (marked as $\Delta$) always has to be calculated between the two dates.

## 1.5 The Qualitative Model

According to the RoI Model, we have to collect the gains and costs which come with the adoption of IS. We collected the gains and costs in an exploratory literature review.

### 1.5.1 Gains

IS has many advantages for the companies. We summarized the gains in the table below.

**Table 1.1:** Qualitative List of Gains of Inner Source

| ID | Gains | Description | Source |
|---|---|---|---|
| G1 | Better detection of bugs | Peer reviews and testing detect many bugs | |
| G2 | Higher static code quality | High code quality is enforced by the committers | |
| G3 | More innovative development | IS improves the research to product transfer | Capraro and Riehle, 2016; Riehle et al., 2016; Melian et al., 2002; Morgan et al., 2011 |
| G4 | Better global perspective | Developers are able to perceive the global perspective by reading the archived informations, thus more simple code can be developed | Capraro and Riehle, 2016; Riehle et al., 2016; Stol and Fitzgerald, 2014 |
| G5 | Employee happiness | The high liberty of developers increase the happiness | |
| G5.1 | Higher employee satisfaction and motivation | The higher level of happiness motivates employees | Capraro and Riehle, 2016; Riehle et al., 2016; Gurbani et al., 2006; K., 2006 |
| G5.2 | Higher employee loyalty | Due to the happiness, employees resign less | |

Continued on next page...

**Table 1.1 – continued from previous page**

| ID | Gains | Description | Source |
|---|---|---|---|
| G6 | Development cost | The overall development costs shrink while fostering IS | |
| G6.1 | Increased code reuse | With IS more code is reused | Capraro and Riehle, 2016; Riehle et al., 2016; Stol and Fitzgerald, 2014; Riehle and Kips, 2012; van der Linden et al., 2009 |
| G6.2 | Reduced process overhead | IS simplifies many processes | Riehle, 2016 |
| G6.3 | Less time spent in meetings | The communication mainly takes place via the mailing-list, thus there are less meetings | |
| G7 | Community-based learning | Through openness of information the employees can learn without prepared training | Capraro and Riehle, 2016; Martin and Hoffman, 2007; Smith and Garber-Brown, 2007 |
| G7.1 | Less project-specific training needed | All project-specific knowledge is written down and accessible | |
| G7.2 | Creation of self-reliant learning culture | Employees are not dependening on trainings but can read the required information purposefully | Capraro and Riehle, 2016; Riehle et al., 2016; Melian and Mähring, 2008 |
| G8 | Avoidance of provider bottlenecks | Employees don't have to wait for developers of a foreign project to fix a bug, instead they can submit their own solution to the foreign project | |

Continued on next page...

**Table 1.1 – continued from previous page**

| ID | Gains | Description | Source |
|---|---|---|---|
| G9 | Cost and risk sharing | Due to the collaboration across organizational units (OrgUnit), the costs and risks are shared | Capraro and Riehle, 2016; Wesselius, 2008 |
| G10 | Independence of reuser and provider | In case of an emergency, the user is not dependent on the provider, and can help himself autonomously | Capraro and Riehle, 2016; Vitharana et al., 2010 |
| G11 | Relief of component providers | The issuer can commit changes, so that the provider can solely adopt it | Capraro and Riehle, 2016; Vitharana et al., 2010 |

In a second step, we analyzed the gains found and assign each gain to a perspective and a group. The perspective states the affected entity:

G lobal; affects the company globally

L ocal; affects single employees or single OrgUnits

The group stated the improvement each gain causes. The improvements can be split in three main groups (Goldin, 1999):

B etter

C heaper

F aster

The found gains are assigned to the perspective and the group in the matrix below.

**Table 1.2:** Assiging Gains to Perspective and Group

| ID | Gains | Group | | | Perspective | |
|---|---|---|---|---|---|---|
| | | B | C | F | G | L |
| G1 | Better detection of bugs | x | | | x | x |
| G2 | Higher static code quality | x | | | x | x |
| G3 | More innovative development | x | | | x | x |
| G4 | Better global perspective | x | | | x | x |

Continued on next page...

**Table 1.2 – continued from previous page**

| ID | Gains | Group | | | Perspective | |
|---|---|---|---|---|---|---|
| | | B | C | F | G | L |
| G5 | Employee happiness | x | | | x | x |
| G5.1 | Higher employee satisfaction and motivation | x | | | x | x |
| G5.2 | Higher employee loyalty | | x | | x | |
| G6 | Development cost | | x | | x | |
| G6.1 | Increased code reuse | | x | x | x | x |
| G6.2 | Reduced process overhead | | x | x | x | x |
| G6.3 | Less time spent in meetings | | x | | x | x |
| G7 | Community-based learning | | x | | x | x |
| G7.1 | Less project-specific training needed | | x | | x | x |
| G7.2 | Creation of self-reliant learning culture | | x | | x | x |
| G8 | Avoidance of provider bottlenecks | | | x | x | |
| G9 | Cost and risk sharing | | | | x | x |
| G10 | Independence of reuser and provider | | | x | | x |
| G11 | Relief of component providers | x | | | | x |

The gains affecting only the local perspective are written in gray because they are ignored within the following model. These gains do not affect the company, but only single employees or OrgUnits. Thus, the gains have no effect on the RoI, but can be used for convincing employees of the advantages of IS. Gain 11 "Cost and risk sharing" is not quantifiable. Its effect is to share the risks and costs of projects over multiple OrgUnits. Thus, many OrgUnits can deal with risks of one project. In return one OrgUnit has to partly face the risks of multiple projects. Nevertheless, these gains are listed for the sake of completeness.

### 1.5.2 Costs

It costs money to adopt and maintain IS. The following five groups of costs are considerd by the literature we looked into. Unlike the gains, there are no categories nor perspectives. All costs always exist.

**Table 1.3:** Qualitative List of Costs of Inner Source

| ID | Costs | Description | Source |
|---|---|---|---|
| C1 | Infrastructure | A company has to maintain the infrastructure to use IS (Hardware and Software) | |

Continued on next page...

**Table 1.3 – continued from previous page**

| ID | Costs | Description | Source |
|---|---|---|---|
| C2 | Training for employees | The company has to provide training for its employees on how to use the tools and how to collaborate successfully | Capraro and Riehle, 2016; Melian et al., 2002 |
| C3 | Committees | The committees maintain and develop IS all times | Riehle, 2016 |
| C3.1 | Program board | The program board subserves all management questions of IS topics like wether to outsource the hosting of the tools required by IS | Riehle, 2016 |
| C3.2 | Program office | The program office is responsible for operational tasks like marketing | Riehle, 2016 |
| C4 | Legal issues | Depending on the company, legal costs like taxes cannot be avoided when adoping IS | |

## 1.6 The Quantitative Model

We now use the previously found gains and costs and provide formulas to quantify them. The gains will be split into the three categories. Thus, there are four sub-models.

**Better** Gains of IS improving the quality of the software product

**Cheaper** Gains of IS shrinking the costs of the product

**Faster** Gains of IS improving the time to market

**Costs** Costs of IS

This segmentation enables users of the method to focus on the RoI suiting their needs. The user can select up to three sub-models of gains and compare them with the costs sub-model. Consequently, he receives a well-tailored RoI of IS. Furthermore, we will ignore all gains and costs operating only as a collection of gains and costs. For example, G6 (Development cost) is an aggregation of the gains G6.1 to G6.3.

For describing how to quantify gains and costs, we use the following notations:

- # The pound sign states the cardinality of a set. For example, #Bugs refers to the total amount of bugs.
- $\overline{\text{foo}}$ The overline states the average of a set. For example, $\overline{\text{Bugs in project}}$ describes the average amount of bugs in all projects.

### 1.6.1 Quantifying Quality Improvements

Software quality is a well researched topic. According to the product quality model from ISO25010, software quality is based on the following dimensions:

- Functional Suitability
- Performance Efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability
- Portability

The following gains will each improve one or more of these dimensions. The improvements of multiple dimensions is not summarizeable to one value. Thereby, we will only state the improvement of single dimensions.

### G1 Better Detection of Bugs

**Explanation** Linus law says:

> "Given enough eyeballs, all bugs are shallow"
>
> Raymond, 1999

In IS, no developer can simply publish the developed code. Before code can be published has to pass the peer review. This peer review is conducted by a committer and detects the majority of bugs. Wiegers, 2002 says 60% of the bugs in software can be detected by peer review.
This gain improves the *reliability*.

**Formula**

$$Q_{R,1} = \Delta\#_{bugs,\ crashes\ reported\ by\ users}$$

### G2 Higher Static Code Quality

The commiter enforces code quality. As a result of the peer review the code is orderly, and easy to maintain. Developers can easily work with the code because they are used to the style of code and naming of the variables. The strict compliance of standards (e.g POSIX) or certain versions of coding languages (e.g. C++-14) additionally increases the static code quality. A merge request without appropriate static code quality will simply be rejected. This encourages developers to instantly foster the quality standards.
This gain improves the *maintainability*.

The effects of the high static code quality are hard to measure. Many effects overlap with other gains of IS. Consequently, the effects of this gain will not be measured to prevent imbrication.

### G3 More Innovative Development

**Explanation** With IS, there is more innovation due to a great diversity of developers. Research by Melian et al., 2002 shows an improvement of the research to product transfer.
This gain improves *functional suitability*, *performance efficiency* and *usability*.

**Formula** An innovation does not affect any special quality dimension, but can affect all the quality dimensions presented by ISO25010. As a result, we cannot measure the improvement of any quality dimension directly. However we will measure the changes in innovativeness.

$$Q_{G3} = \Delta\#_{research\ components\ first\ used\ in\ customer\ products}$$

### G4 Better Global Perspective

**Explanation** The improved global perspective is caused by the all-inclusive archiving of communication. All information, decisions, goals, requirements and code are archived in the mailing list, the discussion board or within a wiki.
With all this information, each developer knows what the code is for. Furthermore, developers know what is going on outside their project. This enables developers to design much simpler code than before. The development will develop from a sweeping blow facing many possibilities to a well structured, simple and aligned development. The developer can now write simple code serving the single necessary purpose.
Additionally, the developers know code which can be reused in their current task and the developers are encouraged to write understandable, well documented code. Contributions containing sections with some "magical code" (unintelligible code) will be rejected by the committer.
The improved global perspective is not quantifiable with a justified effort. However the complexity of code is easily measurable. Consequently, we will only measure the complexity of the code. This represents the major influence of the gain.
This gain improves *maintainability*.

**Formula**

$$Q_{M,G4} = \Delta complexity$$

### Estimation Aid

*complexity* To measure the complexity of the code we recommend a method compliant to ISO 14143. The method used can be substituted by the method of the company's choice. A further discussion of the measures can be found in section 2.3.2.

#### G5.1 Higher Employee Satisfaction and Motivation

**Explanation** The liberties, IS grants a developer leads to a higher employee satisfaction and motivation. There is a high correlation between employee satisfaction and customer satisfaction (Chi and Gursoy, 2009). As a result, not a single dimension of the quality model improves, but the total quality of services increases significantly.

**Formula**

$$Q_{G5.1} = 0.32 * \Delta employee\ happiness$$

**Estimation Aid**

0.32 Chi and Gursoy, 2009 state the correlation value of employee satisfaction and customer satisfaction with 0.32.

*employee happiness* To measure the employee's happiness, we suggest company's own survey or the Gallup Workplace Audit method (Harter, Schmidt and Keyes, 2003).

### 1.6.2 Quantifying Cost Savings

#### G5.2 Higher Employee Loyalty

**Explanation** Due to the higher employee motivation and satisfaction, less employees will resign. There are less people with know-how leaving the company, and no new employees are needed. Bliss, 2004 states the costs of an employee leaving and hiring a new with 150% to 250% of the annual salery depending on the employee's position. Bliss presents five groups of costs:

- Costs Due to a Person Leaving
- Recruitment Costs
- New Hire Costs
- Training Costs
- Lost Productivity Costs

**Formula**

$$S_{G5.2} = cost_{employee\ turnover}$$

**Estimation Aid**

$cost_{employee\ turnover}$ Bliss, 2004 presents a checklist of the costs in his paper "Cost of employee turnover".

### G6.1 Increased Code Reuse

**Explanation** When fostering IS, the developed code will be accessible within the company. Furthermore, developers will be used to the foreign code they may worked on. Due to good documentation and awareness of the code it can be reused. The reuse of code is favorable because reused code has no need to be developed and maintained additionally. The maintenance will take place on the original code. (Soora, 2014; Mohagheghi, Conradi, Killi and Schwarz, 2004; Devanbu, Karstu, Melo and Thomas, 1996; Emam, 2003)

**Formula**

$$S_{G6.1} = \#loc_{total} * cost_{LoC} * ratio_{reusable\ code}$$

**Estimation Aid**

$\#loc_{total}$ The total number of lines of code (LoC) has to be calculated. We recommend to count logical LoCs (LLoC). LLoCs ignore the lines containing only whitespace characters or braces. To get a better idea of LLoCs take a look at the paper "A SLOC Counting Standard" by Nguyen, Deeds-Rubin, Tan and Boehm, 2007.

$cost_{LoC}$ If a standard value for this cost already exists within the analyzed company this value has to be used. Otherwise, Poulin and Caruso, 1993 state the cost of developing a new line of code with $200\$/_{loc}$ (without maintenance).

$ratio_{reusable\ code}$ To measure the amount of code averted, the methodology suggested by Frakes and Terry, 1996 is used.

### G6.2 Reduced Process Overhead

#### Explanation

> "[...] inner source [...] enables collaboration across intra-organizational boundaries"
>
> Riehle et al., 2016

This collaboration enables developers to fix their issues directly without initiating a complicated process.
Without IS, a developer facing a bug has to file an issue. The issue has to be prioritized, stated (bug, feature or wont fix) and allocated to a responsible developer. This developer will work on the issue some days later. The developer will have to ask the issue's initiator to fully understand the issue. Finally, the bug is fixed.
With IS, a competent developer can branch the affected project, fix the bug himself and create a merge request. The responsible committer only has to accept the request and the bug is fixed.
There are many processes which can be simplified or omitted using IS.

#### Formula

$$S_{G6.2} = \sum_{simplified\ processes} \Delta C_{processes}$$

#### Estimation Aid

$C_{processes}$ Several processes will get easier, like the bug fixing process. In a first step, all simplified processes have to be collected. Second, the difference of costs for each process have to be calculated. The process can get cheaper because the process is less used or the process became simplified.

### G6.3 Less Time Spent in Meetings

**Explanation** While fostering IS, the communication mainly runs via open communication medium like mailing lists or discussion boards. This has multiple effects:

- Archiving of communication by default
- Discussions made on mailing list don't use up time in a meeting.
- The geographical distance between developers doesn't matter.

We will only take a look at the second point. Due to IS, the time of meetings can be scaled-down or meetings even become obsolete.

**Formula**

$$S_{G6.3} = \overline{hourly\ wage} * (\Delta t_{shorter\ meetings} + t_{dropped\ meetings})$$

$$\Delta t_{shorter\ meetings} = \sum \Delta t_{meeting_{shorter}}$$

$$t_{dropped\ meetings} = \sum t_{meetings_{dropped}}$$

**Estimation Aid** To measure the time saved due to more efficient meetings all types of meetings within a software project have to be gathered. The second step is to detect the difference between the time in meetings with and without IS.

$\Delta t_{meetings_{shorter}}$ This states the amount of time saved due to shorter meetings. For example, the sprint planning meeting, in which the product owner describes the project and developers discuss their workloads, might become shorter because the product owner shared the information earlier in the mailing list. The first part becomes obsolete and developers can better prepare themselves for the meeting.

$t_{meetings_{less}}$ This states the amount of time saved due to omitted meetings. For example, the daily scrum meeting, in which the developers discuss experienced problems, becomes obsolete because these problems will be discussed via mailing list.

### G7.1 Less Project-Specific Training Needed

**Explanation** IS communities and open information allows self-reliant learning. On account of the availability of information about the project there is no need to create training sessions about these topics. The time used by employees to attend the training can now be used to read the mailing list purposefully.

**Formula**

$$S_{G7.1} = \sum cost_{training\ design}$$

**Estimation Aid** With IS, information regarding problems, decisions, discussions and documentation are archived on the mailing list. At first, the saved training has to be identified. All training teaching project-specific information and skills have to be analyzed.

$cost_{training\ design}$ The costs of each identified training have to be gathered.

### G7.2 Creation of Self-Reliant Learning Culture

**Explanation** With time a self-reliant teaching culture will be established. The carefully created documentation of decisions, problems and experience will allow new programmers to solve issues by themselves without the help of an experienced programmer.

**Formula**

$$S_{G7.2} = \overline{hourly\ wage} * t_{Mentoring}$$

**Estimation Aid**

$t_{Mentoring}$ This states the amount of time saved due to self-reliant learning of new programmers. To measure the time used by experienced developers to help newcomers an interview with the experienced ones is recommended.

### 1.6.3 Quantifying Speed Improvements

The effect of the gains mentioned below does not affect the Time-to-Market (TtM) directly, however it affects the duration of some tasks in a project. Consequently, the following gains indirectly influences the TtM.

#### G6.1 Increased Code Reuse

**Explanation** One of the advantages of code reuse is that the reused code already exists. Furthermore, the code doesn't have to be developed seperately. Due to the time saved by omitting the reproduction, the product can be published earlier.
The omitted time for maintenance is mainly no gain for the TtM but saves costs.

**Formula**

$$TTM_{G6.1} = \sum_{reused\ code} t_{coding}$$

**Estimation Aid**

$t_{coding}$ Time used to develop the observed code.

#### G6.2 Reduced Process Overhead

**Explanation** In some scenarios (for example software product line engineering (Riehle et al., 2016)), IS can significantly reduce the process overhead. By simplifying processes, the time span between start and end will become smaller. This leads to lower latency until the effect of the process takes place.

**Formula**

$$TTM_{G6.2} = \sum \#process\ iterations * \Delta delay_{process}$$

**Estimation Aid** At first, the affected processes has to be analyzed.

$\#process\ iterations$ Number of times a process was used.

$\Delta delay_{process}$ The difference of delay caused by the process.

#### G8 Avoidance of Provider Bottlenecks

**Explanation** IS can avoid provider bottlenecks because individual parties are empowered to perform own contributions instead of depending on component providers' schedules.

**Formula**

$$TTM_{G8} = \#issues * \overline{time\ until\ fix}$$

**Estimation Aid**

$\#issues$ Amount of issues stating bug fixes and feature requests.

$\overline{time\ until\ fix}$ The average time needed to close a bug report. Additionally issues stating feature requests have to be included. While fostering IS, each developer can implement the necessary code himself. The latency between creating the issue and publishing the code converges towards zero. To pinpoint this value, the time needed to code the fix has to be subtracted. The coding itself still has to be done.

### 1.6.4 Quantifying Adoption Costs

#### C1 Infrastructure

**Explanation** Nowadays, companies often use a publicly managed cloud for the tools needed. Thereby, they outsource the risk of self-hosting the service. To adopt IS within a company, a mailinglist/discussion board, a central repository and an issue tracking system are required. We recommend using services like GitLab, because they combine all required tools.

**Formula** The service can be self-hosted or outsourced. Depending on the company's choice, the measurement of the costs differ.

**Self-Hosted** Self-hosting the service causes several different costs. These result in the costs of the hardware and software.

$$C_1 = C_{Hardware} + C_{Software}$$

$$C_{Hardware} = write\ down_{Hardware} + maintenance_{Hardware}$$

$$C_{Software} = write\ down_{Software} + maintenance_{Software} + development$$

**Managed Cloud** The costs of hosting the service is easier to calculate. There are only the costs stated by the contract with the hosting company.

$$C_1 = C_{Renting}$$

**Estimation Aid**

$write\ down_{Hardware}$ The cost of the hardware is split over a certain time span (differs between countries).

$maintenance_{Hardware}$ Hardware needs maintenance, power and cooling.

$write\ down_{Software}$ The cost of the software is split over a certain time span (differs between countries).

$maintenance_{Software}$ Similar to the hardware, software needs maintenance as well, for example the user management.

*development* To integrate the service in the IT infrastructure of the company, some development effort is necessary. Moreover, GitLab is open source, so missing functionality can be developed by the company itself.

$C_{Renting}$ There will be an individual contract for big companies. GitLab states its price with $\$199/_{year*user}$ for enterprises (Effective at 2017-09-02, Enterprise Edition Premium).

### C2 Training

**Explanation** Adopting IS will change many processes in the company. The developers have to use a new tool chain. To inform the employees, training is required.

**Formula** At first, the training has to be developed. Second, the training has to be attended by all affected employees.

$$C_2 = C_{development} + C_{attendance}$$

$$C_{attendance} = \sum_{\#employees} (\sum t_{training}) * \overline{hourly\ wage}$$

### Estimation Aid

$C_{development}$ The cost of the development of the training saved.

### C3.1 Program Board

**Explanation** The program board is responsible for management questions like *self hosting* vs *managed cloud*. The main costs are the labor costs. There might be costs for rooms or material, but those are marginal.

**Formula**

$$C_{3.1} = \overline{hourly\ wage} * time_{program\ board}$$

### C3.2 Program Office

**Explanation** The program office is responsible for operational tasks, like marketing or preparation of training. Analogously to the program board, the main costs are labor costs. In the beginning of the IS program marketing is necessary. The IS adoption comes with many changes for the developers. An other task of the program office are the modifications of the processes during the adoption of IS (see G6.2). This development of new processes is work of the program office.

**Formula**

$$C_{3.2} = \overline{hourly\ wage} * time_{program\ office}$$

### C4 Legal Issues

Despite the research by Stol, Babar, Avgeriou and Fitzgerald, 2011, we think there are legal costs and challenges. There can be issues with the following topics which have to be faced:

- Taxation
- Internal licensing [1]
- Intellectual property [1]
- Export control

This list is not reliable and might not cover all legal issues. We do not provide any formulas here. They will significantly differ between companies. The legal department in the company will know the costs.

## 1.6.5 Summary of Required Values

- $\overline{hourly\ wage}$
- $\#_{bugs,\ crashes\ reported\ by\ users}$
- $\#_{research\ components\ first\ used\ in\ customer\ products}$
- $complexity_{code}$
- $employee\ happiness$
- $cost_{employee\ turnover}$
- $\#loc_{total}$
- $cost_{LoC}$
- $ratio_{reusable\ code}$
- $C_{process_{simplified}}$
- $t_{meetings_{shorter}}$
- $t_{meetings_{dropped}}$
- $cost_{training_{design}}$
- $t_{mentoring}$
- $t_{coding}$
- $\#process\ iterations$
- $delay_{process}$
- $\#issues$
- $\overline{time\ unitl\ fix}$
- $write\ down_{hardware}$
- $maintenance_{hardware}$
- $write\ down_{software}$

[1] These challenges were identified by Stol et al., 2011 for OS. We think they are adoptable to IS.

- $maintenance_{software}$
- $development_{software}$
- $renting$
- $development_{training}$
- $attendance_{training}$
- $t_{program\ board}$
- $t_{program\ office}$

## 1.7 The Method to Determine the Return on Investment of Inner Source

To summarize the previous steps we will now apply the found gains and costs (1.5) with their measurements (1.6) to the RoI model (1.4).

$$RoI(time\ span) = \frac{gains(time\ span) - costs(time\ span)}{costs(time\ span)}$$

The analyzed gains and costs below state the difference between before IS and afterwards - this is the time span. The differences (marked as $\Delta$) always have to be calculated between the two dates.

At first, one can decide which sub-mod els ought to be used. Up to two sub-models can be skipped, depending on the focus of the measurement. Second, the selected models and the cost model have to be applied to the company. Finally, the calculated values have to be applied to the formula calculating the RoI. Gains not affecting the analyzed company can be skipped.

**Better** To determine the quality improvement triggered by IS the following gains have to be calculated:

**Table 1.4:** List of Gains Improving the Quality

| ID | Gains | Formula |
|---|---|---|
| G1 | Better detection of bugs | $Q_{R,G1} = \Delta\#_{bugs,\ crashes\ reported\ by\ users}$ |
| G3 | More innovative development | $Q_{G3} = \Delta\#_{research\ components\ first\ used\ in\ customer\ products}$ |
| G4 | Better global perspective | $Q_{M,G4} = \Delta complexity$ |
| G5.1 | Higher employee satisfaction and motivation | $Q_{G5.1} = 0.32 * \Delta employee\ happiness$ |

It's not easy to unify the effects of the quality gains. Consequently, the effects will be merged into a set of improvements.

$$Better = \{G1,\ G3,\ G4,\ G5.1\}$$

**Cheaper** To determine the savings triggered by IS, the following gains have to be calculated:

**Table 1.5:** List of Gains Omitting Costs

| ID | Gains | Formula |
|---|---|---|
| G5.2 | Higher employee loyalty | $S_{G5.2} = cost_{employee\ turnover}$ |
| G6.1 | Increased code reuse | $S_{G6.1} = \#loc_{total} * cost_{LoC} * ratio_{reusable\ code}$ |
| G6.2 | Reduced process overhead | $S_{G6.2} = \sum_{simplified\ processes} \Delta C_{processes}$ |
| G6.3 | Less time spent in meetings | $S_{G6.3} = \overline{hourly\ wage} * (\Delta t_{shorter\ meetings} + t_{dropped\ meetings})$<br>$\Delta t_{shorter\ meetings} = \sum \Delta t_{meeting_{shorter}}$<br>$t_{dropped\ meetings} = \sum t_{meetings_{dropped}}$ |
| G7.1 | Less project-specific training needed | $S_{G7.1} = \sum cost_{training\ design}$ |
| G7.2 | Creation of self-reliant learning culture | $S_{G7.2} = \overline{hourly\ wage} * t_{Mentoring}$ |

The saving due to IS can be unified by adding the result of each gain. Consequently, the total saving due to IS can be determined.

$$Cheaper\ =\ G5.2\ +\ G6.1\ +\ G6.2\ +\ G6.3\ +\ G7.1\ +\ G7.2$$

**Faster** To determine the improvement of the Time-to-Market triggered by IS, the following gains have to be calculated:

**Table 1.6:** List of Gains Improving the Time-to-Market

| ID | Gains | Formula |
|---|---|---|
| G6.1 | Increased code reuse | $TTM_{G6.1} = \sum_{reused\ code} t_{coding}$ |
| G6.2 | Reduced process overhead | $TTM_{G6.2} =$<br>$\sum \#process\ iterations * \Delta delay_{process}$ |
| G8 | Avoidance of provider bottlenecks | $TTM_{G8} = \#issues * \overline{time\ until\ fix}$ |

The Time-to-Market improvements due to IS can be unified by adding the result of each gain. Consequently, the total improvement due to IS can be determined.

$$Faster \ = \ G6.1 \ + \ G6.2 \ + \ G8$$

**Costs** To calculate the costs of IS, the following costs have to be calculated:

**Table 1.7:** List of the Costs of Inner Source

| ID | Costs | Formula |
|---|---|---|
| C1 | Infrastructure | $C_1 = C_{Hardware} + C_{Software}$<br>$C_{Hardware} =$<br>$write\ down_{Hardware} + maintenance_{Hardware}$<br>$C_{Software} = write\ down_{Software} +$<br>$maintenance_{Software} + development$<br>or<br>$C_1 = C_{Renting}$ |
| C2 | Training for employees | $C_2 = C_{development} + C_{attendance}$ |
| C3.1 | Program board | $C_{3.1} = \overline{hourly\ wage} * time_{program\ board}$ |
| C3.2 | Program office | $C_{3.2} = \overline{hourly\ wage} * time_{program\ office}$ |
| C4 | Legal issues | |

The costs of adopting and maintaining IS can be unified by adding the result of each cost. Consequently, the total cost of IS can be determined.

$$Costs \ = \ C1 \ + \ C2 \ + \ C3.1 \ + \ C3.1 \ + \ C4$$

**RoI** The RoI of IS can now easily be calculated with the formula from section five.

$$RoI = \frac{Gains - Costs}{Costs}$$
$$= \frac{(Better + Cheaper + Faster) - Costs}{Costs}$$

The values of the ignored sub-models will be set to 0.
Finally, the RoI of IS was determined and calculated.

## 1.8 Validation and Limitations

We conducted a case study research to validate the method at a company adopting IS. A two-step approach to validate the method was used. First, we validated our qualitative model. Second, we validated the method. Both steps were conduced with different data provided by said company.

### 1.8.1 Validation of the Qualitative Model

In a first step we validated our qualitative model, stating the gains and costs, by comparing our model with the company's interests. We discussed our model and checked the importance of each gain and cost for the company. The company expressed the importance of each gain and cost on a scale from 1 (unimportant) to 3 (very important). The results are shown below.

**Table 1.8:** Validation of Qualitative Gains

| ID | Gains | Importance |
|---|---|---|
| G1 | Better detection of bugs | 3 |
| G2 | Higher static code quality | 1 |
| G3 | More innovative development | 2 |
| G4 | Better global perspective | 1 |
| G5 | Employee happiness | 1 |
| G5.1 | Higher employee satisfaction and motivation | 1 |
| G5.2 | Higher employee loyalty | 1 |
| G6 | Development cost | 1 |
| G6.1 | Increased code reuse | 1 |
| G6.2 | Reduced process overhead | 1 |
| G6.3 | Less time spent in meetings | 2 |
| G7 | Community-based learning | 2 |
| G7.1 | Less project-specific training needed | 2 |
| G7.2 | Coreation of self-reliant learning culture | 2 |
| G8 | Avoidance of provider bottlenecks | 1 |

**Table 1.9:** Validation of Qualitative Costs

| ID | Costs | Importance |
|---|---|---|
| C1 | Infrastructure | 1 (software development 2) |
| C2 | Training for employees | 2 |
| C3 | Committees | 1 |
| C3.1 | Program board | 2 |
| C3.2 | Program office | 3 |
| C4 | Legal issues | 1 |

The table shows that the company has very precise expectations of the gains. In this case, the precision of the qualitative model is low. With the separation of the model into the three sub-models, the precision increases. The company didn't have any other gain or cost in mind. As a result, the model seems to be complete. The recall is zero. As we only validated the model with one case, the validation is not generalizable. We suggest further case studies to validate the model.

### 1.8.2 Validation of the Quantitative Method

To validate our method, the company provided company-internal measurements of the projects. Unfortunately this data was of no use for the validation of the quantiative method. Accordingly, we weren't able to validate the quantitative method.

### 1.8.3 Limitations

In this paper, we only validated the qualitative model with one case. The case resulted in our model being complete. One successful case doesn't imply that our model is complete, but indicates it. Further case studies are required to better evaluate the qualitative model.
The validation of the method using the quantitative model had no result due to insufficient data provided by the company. Further case studies are required as well. For the evaluation of both - the qualitative and the quantitative model - a case with a company maintaining IS for quite some time is helpful.
Future research might address the question whether the RoI method provided in this paper is applicable to all categories of IS presented by Capraro and Riehle, 2016 and Stol et al., 2014. To validate the method, one could apply historical data to the method and analyze the outcome. The data published by a company which had already adopted IS could be used as historical data.

Moreover, some of the formulas used to measure a certain gain or cost could not be taken out of common literature, but carefully were developed by the authors. These formulas have to be utilized with caution. They were not supplied by the literature, but were obviously to the authors (and approved by the case study).

## 1.9 Future Work

In future research, one can conduct further validation of the method as described in the limitation section. Especially formulas which were not found in literature have to be validated in future research work. In this paper, the legal costs are not pinpoint. To determine the legal costs, research focused on one country is necessary. In future research the gain better global perspective has to be refactored to precisely name the simplification of the code not only as effect but as gain. A collection of best practices assists companies considering an adoption in the question of how much of the projects costs have to be spent on marketing. Further work on how to quantify and unify the quality gains is desirable. Finally, the model could be extended by an analysis of the initial costs of the IS adoption, if existing.

## 1.10 Conclusion

In this paper, we described the first method to determine the RoI for IS. At first, we analyzed existing RoI models in literature. Second, we collected the gains and costs of IS. We applied the gains and costs to the model from step one. Third, we quantified all gains and costs by supplying forumlas to calculate each gain and cost. Fourth, we described an easy-to-use method on how to use the quantified model.
The validation of the qualitative model exemplary indicates that the model is complete. The validation of the method to determine the RoI of IS was inconclusive.
With the method described in the paper, companies considering the adoption of IS can now determine the RoI. As a result, companies might hesitate less to adopt IS.

# 2 Elaboration

## 2.1 Thesis Goals

### 2.1.1 Original Thesis Goals

The main goal of the thesis was to develop a generic, quantitative and scientifically well-grounded return on investment model for inner source on literature research. The validation in the end is based on a workshop in a cooperating company. During the research, the following results should be elaborated:

- Identified generic costs and gains of IS
- Identified formulas or metrics to quantify costs and gains
- Evaluation of the method

### 2.1.2 Changes to Thesis Goals

The goals have not been changed.

## 2.2 Model Developed in Previous Research (NYT)

We started with the preliminary RoI of IS model, developed in a previous research project. The model lists most of the gains and costs of IS. Unfortunately, the preliminary model is a qualitative model, only listing the gains and costs. Furthermore, the model is not fully consistent as for example the effects of some gains are overlapping (G2, G4: efficient development leads to faster Time-to-Market). The column names “Category” and “Perspective” were renamed to “Cat” and “Per” due to reasons of space.

**Table 2.1:** Preliminary Model of the Gains of IS

| ID | Gains | Description | Source | Cat | Per | Indicator |
|---|---|---|---|---|---|---|
| G1 | Reuse of code | Marketplace to share well documented code. Up to 85% can be reusedJones, 1984. | Capraro and Riehle, 2016; Riehle et al., 2016; Stol and Fitzgerald, 2014 | AQS | GUP | ♯man-days needed to code and test the reused code |
| G2 | Faster time-to-market | Due to the reuse of code and experience from other software projects a new product can be deployed faster. | Dinkelacker et al., 2002; Stol and Fitzgerald, 2014 | A | G | ♯man-days from project beginning to deployment |
| G3 | More innovative development | | Capraro and Riehle, 2016; Riehle et al., 2016; Melian et al., 2002; Morgan et al., 2011 | Q | G | |
| G4 | More efficient development | | Neus and Scherf, 2005 | S | GU | |
| G5 | More efficient use of resources | During his spare time a programmer can work on other projects. | Stol and Fitzgerald, 2014 | P | G | ♯man-days in spare time |

Continued on next page...

**Table 2.1 – continued from previous page**

| ID | Gains | Description | Source | Cat | Per | Indicator |
|---|---|---|---|---|---|---|
| G6 | Simplified developer deployment | Due to the good documentation of the code new developers can be easily deployed to new projects. | Capraro and Riehle, 2016; Stol and Fitzgerald, 2014; Melian et al., 2002; Dinkelacker et al., 2002 | S | G | Compare time to familiarization with and without IS |
| G7 | Collaboration of detached developers | Collaboration of developers working in different locations | Capraro and Riehle, 2016; Riehle et al., 2016; Lindman et al., 2013 | P | G | |
| G8 | Higher employee motivation | Voluntariness increases motivation. | Capraro and Riehle, 2016; Riehle et al., 2016; Gurbani et al., 2006; K., 2006 | PS | GPU | Compare quality and speed with and without IS |
| G9 | Cost and risk sharing | | Capraro and Riehle, 2016; Wesselius, 2008 | P | G | |
| G10 | Internal information exchange | | Capraro and Riehle, 2016; Riehle et al., 2016 | Q | (G)P | |
| G11 | Community-Based learning | | Capraro and Riehle, 2016; Martin and Hoffman, 2007; Smith and Garber-Brown, 2007 | QS | (G)P | |

Continued on next page...

**Table 2.1 – continued from previous page**

| ID | Gains | Description | Source | Cat | Per | Indicator |
|---|---|---|---|---|---|---|
| G12 | Openness of knowledge | Company wide sharing of experience in Wikis and Mail-inglists. | Capraro and Riehle, 2016; Melian and Mähring, 2008 | Q | (G)P | |
| G13 | Improved global perspective | To implement a component suiting multiple projects the global perspective has to be known. | Capraro and Riehle, 2016; Riehle et al., 2016; Stol and Fitzgerald, 2014 | Q | PU | |
| G14 | Indepen-dence between reuser and provider | | Capraro and Riehle, 2016; Vitharana et al., 2010 | S | U | |
| G15 | Relief of component providers | The issuer can commit the changes so that the provider can solely adopt it. | Capraro and Riehle, 2016; Vitharana et al., 2010 | Q | P | |

Continued on next page...

**Table 2.1 – continued from previous page**

| ID | Gains | Description | Source | Cat | Per | Indicator |
|---|---|---|---|---|---|---|
| G16 | Increase code quality | | Capraro and Riehle, 2016; Riehle et al., 2016; Stol and Fitzgerald, 2014; Riehle et al., 2009; Raymond, 1999; Dinkelacker et al., 2002; Smith and Garber-Brown, 2007 | Q | GPU | |
| G17 | Decrease of maintenance costs | Reused code is already tested. | | P | GU | |
| G18 | Less build breaks | | Raymond, 1999 | AQS | GU | ♯Less build breaks * costs per build |
| G19 | Faster implementation of bug fixes | Multiple users can fix the bug. Furthermore, fixing a bug in a shared component has a higher priority. | | S | GPU | |
| G20 | Less bugs shipped to customer | | Raymond, 1999 | Q | GU | ♯less bugs shipped * price per bug shipped |

**Table 2.2:** Preliminary Model of the Costs of IS

| ID | Costs | Description | Source |
|---|---|---|---|
| C1 | Infrastructure | A company has to maintain the infrastructure to use IS. | |
| C1.1 | Hardware | The server where e.g. the repositories and wikis are hosted. | |
| C1.2 | Software | The software required to share code and knowledge. | |
| C1.2.1 | Repository | A central space to share code; Preferred a version control system (e.g.git, svn). | Stol et al., 2014; Capraro and Riehle, 2016; Smith and Garber-Brown, 2007 |
| C1.2.2 | Wiki | A central space to share knowledge. | Capraro and Riehle, 2016; Smith and Garber-Brown, 2007; Martin and Hoffman, 2007 |
| C1.2.3 | Bugtracker | A global platform for task accumulation is needed to present tasks to foreign developers. | Capraro and Riehle, 2016; Vitharana et al., 2010 |
| C1.2.4 | Mailinglists, message boards | To achieve the decision making process; Enhances global view for developers. | Stol et al., 2014; Lindman et al., 2013; Lindman et al., 2010; Riehle, 2016 |
| C1.2.5 | Tools | The developers need tools for developing in teams and for foreign code exploration. | |
| C2 | Employees | IS requires additional personal for maintenance | |

Continued on next page...

**Table 2.2 – continued from previous page**

| ID | Costs | Description | Source |
|---|---|---|---|
| C2.1 | Committer | Each project needs one or more committers who decide if a patch is accepted and included to the project or not. Maybe this task can be done from active developers inside the project. | Capraro and Riehle, 2016 |
| C2.2 | Hard- and software administrator | The hard- and software have to be maintained. If the company buys the software as a service no personal is required. | |
| C3 | Training for employees | The company has to provide training for the employees on how to use the tools and how to collaborate successfully. | Capraro and Riehle, 2016; Melian et al., 2002 |
| C3.1 | Foreign code exploration | When a company uses IS processes the developers often have to work on different projects. A special training to simplify the familiarization process is required. | Capraro and Riehle, 2016 |
| C3.2 | Tools | The developer have to be trained on how to use the new tools (Wiki, IDE, VCS). | Gurbani et al., 2010 |
| C4 | Coordination of resources | With IS processes the developers can be deployed easier. This flexibility has to be controlled. | |
| C5 | Documentation of code and knowledge | To simplify the reuse of code and sharing of knowledge the developers have to invest more time e.g. to enhance the documentation of the code or to write a new entry in a wiki. | |

We did a multiple step approach to improve the model.

### 2.2.1 Column Names

At first we changed the columns. We dropped the indicator column. The data presented in this column was partly used to create the formulas in the quantitative model. The columns "Category" and "Perspective" were dropped as well. In table 1.2 the information is used again.

### 2.2.2 Completion of the Preliminary Model

In a second step, the model was extended by missing gains and costs. The gains missing were the "reduced process overhead" and "less time spent in meetings". The costs missing were the committee related costs and the legal costs.

### 2.2.3 Deduplication

In the next step, some entities were removed or merged due to duplicates. The duplicates occurred because of some points overlapping with others.

### 2.2.4 Omit Entities

Finally, we decided to omit several entities. We omitted mainly the local gains, affecting each employee on its own. These gains are irrelevant for company-wide decisions. The local gains serve as arguments to win unpersuaded employees over. Therefore, no quantitative analysis is needed.

We developed the model by retracing these steps. The new model is the model presented in the section "The Qualitative Model". The step "Column Names" was already applied to the table below in order to increase the readability.

**Table 2.3:** Changes of Preliminary Model

| ID | Gains/Costs |
|---|---|
| G1 | Reuse of code |
| The gain was only renamed and renumbered | |
| G6.1 | Increased code reuse |
| G2 | Faster time-to-market |
| This gain is a collection of the gains contained in the quality model; To prevent the overlapping of gains this gain is omitted | |

Continued on next page...

**Table 2.3 – continued from previous page**

| ID | Gains/Costs |
|---|---|
| G3 | More innovative development |
| The gain was only renamed | |
| G3 | More innovative development |
| G4 | More efficient development |
| This gain is a result of a majority of other gains; To prevent the overlapping of gains this gain is omitted | |
| G5 | More efficient use of resources |
| This gain is a result of a majority of other gains; To prevent the overlapping of gains this gain is omitted | |
| G6 | Simplified developer deployment |
| The gain is an effect of G2 and G4; to prevent imbrication the gain is omitted | |
| G7 | Collaboration of detached developers |
| The gain partly induces G3; to prevent imbrication the gain is omitted | |
| G8 | Higher employee motivation |
| This gain was split into two gains, because the increased motivation has two different effects; Furthermore the gain was renamed and renumbered | |
| G5 | Employee Happiness |
| G5.1 | Higher employee satisfaction and motivation |
| G5.2 | Higher employee loyalty |
| G9 | Cost and risk sharing |
| This gain is no monitary gain of IS; The gain was renumbered | |
| G11 | Cost and risk sharing |
| G10 | Internal information exchange |
| This gain is summarized in G11 | |
| G11 | Community-Based learning |
| This gain became a collection for G10; Furthermore the gain was renumbered | |
| G7 | Community-Based learning |
| G12 | Openness of knowledge |
| This gain overlaps with G11; Thus it is omitted | |
| G13 | Improved global perspective |
| This gain was renumbered | |
| G4 | Improved global perspective |
| G14 | Independence between reuser and provider |
| This gain was renumbered | |
| G12 | Independence between reuser and provider |
| G15 | Relief of component providers |
| This gain was renumbered | |
| G13 | Relief of component providers |

Continued on next page...

**Table 2.3 – continued from previous page**

| ID | Gains/Costs |
|---|---|
| G16 | Increase code quality |
| This gain was renamed and renumbered | |
| G2 | Higher static code quality |
| G17 | Decrease of maintenance costs |
| This gain is an effect of gain G1; Thus the gain is omitted | |
| G18 | Less build breaks |
| This gain is an effect of gain G20; Thus the gain is omitted | |
| G19 | Faster implementation of bug fixes |
| This gain is summarized in G20 | |
| G20 | Less bugs shipped to customer |
| This gain was put together in summary G19; Furthermore the gain was renamed and renumbered | |
| G1 | Better detection of bugs |
| The collaboration across OrgUnits enables developers to omit the use of complex processes by providing the desired changes in a foreign source code of the project themselves | |
| G6.2 | Reduced process overhead |
| The all-inclusive archiving enables developers to discuss things via mailing list before the meeting starts, thus the meeting is shorter | |
| C6.3 | Less time spent in meetings |
| C1 | Infrastructure |
| This cost summarizes all subordinate costs | |
| C1 | Infrastructure |
| C1.1 | Hardware |
| This cost is summarized in C1; Thus it is omitted | |
| C1.2 | Software |
| This cost is summarized in C1; Thus it is omitted | |
| C1.2.1 | Repository |
| This cost is summarized in C1.2; Thus it is omitted | |
| C1.2.2 | Wiki |
| This cost is summarized in C1.2; Thus it is omitted | |
| C1.2.3 | Bugtracker |
| This cost is summarized in C1.2; Thus it is omitted | |
| C1.2.4 | Mailinglists, message boards |
| This cost is summarized in C1.2; Thus it is omitted | |
| C1.2.5 | Tools |
| This cost is summarized in C1.2; Thus it is omitted | |
| C2 | Employees |
| Contrary to the previous opinion, no additional employees are required to conduct the subordinated tasks; Thus the cost is omitted | |

Continued on next page...

**Table 2.3 – continued from previous page**

| ID | Gains/Costs |
|---|---|
| C2.1 | Committer |
| This task is conducted by developers assigned to each project | |
| C2.2 | Hard- and software administrator |
| This cost is overlapping with C1; Thus it is omitted | |
| C3 | Training for employees |
| The subordinate costs are summarized; Furthermore the cost was renumbered | |
| C2 | Training for employees |
| C3.1 | Foreign code exploration |
| This cost is included in C3; Thus it is omitted | |
| C3.2 | Tools |
| This cost is included in C3; Thus it is omitted | |
| C4 | Coordination of resources |
| This cost is included in the costs "program office"; Thus it is omitted | |
| C5 | Documentation of code and knowledge |
| The documentation is done autonomously; Thus it is omitted | |
| The preliminary model doesn't include the costs for the committees; Thus they are added (Riehle, 2016) | |
| C3 | Committees |
| C3.1 | Program Board |
| C3.2 | Program Office |
| The preliminary model doesn't include the costs for legal issues; Thus they are added | |
| C4 | Legal Issues |

## 2.3 Decision Making

### 2.3.1 Gitlab Plan

Gitlab offers two different plans for enterprises. *Enterprise Edition Starter (EES)* and *Enterprise Edition Premium (EEP)* [1]. In the qualitative model (1.6.4) we stated the costs of Gitlab with $\$199/_{year*user}$. This is the price of the model EEP. EES is much cheaper (around $39\$/_{year*user}$).
The reasons for choosing the EEP plan were its features. Companies considering the adoption of IS are mostly relying on being able to develop software. With EES, a company cannot rely on this ability. If a problem occurs, the Gitlab support will start to handle the issue not until the next business day. With EEP, the support has to react within four hours.

[1] https://about.gitlab.com/products/, effective at 2017-09-02

This is assumed to be essential for companies adopting IS. Thus, the EEP plan was used.

### 2.3.2 Software Complexity Measurements

In section 1.6.1, we recommended one of the functional size measurements described in ISO 14143 to measure the complexity of software. We suggested this metric based on the result of a short, exploratory literature review were the following metrics (Yu and Zhou, 2010 ISO14143-6):

- Halstead complexity measures
- Cyclomatic complexity measures
- Object-oriented class metrics
- Software package metrics
- Functional size measurements (ISO 14143-6:2012)
  - ISO 19761 (COSMIC method)
  - ISO 20926 (IFPUG method)
  - ISO 20968 (MkII method)
  - ISO 24570 (NESMA method)
  - ISO 29881 (FiSMA method)

A well funded comparison of these methods is not in the scope of this paper. We suggest the measurements defined by the ISO, as the publication via ISO implicates a continuously high quality. The method to measure the complexity of the software might be substituted by the company's favorite complexity measurements.

### 2.3.3 Better Global Perspective or Simpler Code

During the elaboration of the paper we recognized the main benefit of an improved global perspective is a simplification of the code. Some effects of the gain "Higher Static Code Quality" affect the simplicity of the code as well. It was not possible for us to validate this development enough to mention it in the paper. Further validation is necessary to add the gain "Simpler Code" to the model or to substitute the gain "Better Global Perspective".

# Appendix A Abbreviations

| | |
|---|---|
| IS | Inner Source |
| RoI | Return on Investment |
| OS | Open Source |
| ROSI | Return on Investment for a security investment |
| OrgUnit | Organizational Unit |
| LoC | Line of Code |
| LLoC | Logical Line of Code |
| TtM | Time-to-Market |
| EES | Enterprise Edition Starter |
| EEP | Enterprise Edition Premium |

# Appendix B List of Tables

1.1 Qualitative List of Gains of Inner Source . . . . . 7
1.2 Assiging Gains to Perspective and Group . . . . . 9
1.3 Qualitative List of Costs of Inner Source . . . . . 10
1.4 List of Gains Improving the Quality . . . . . 25
1.5 List of Gains Omitting Costs . . . . . 26
1.6 List of Gains Improving the Time-to-Market . . . . . 26
1.7 List of the Costs of Inner Source . . . . . 27
1.8 Validation of Qualitative Gains . . . . . 28
1.9 Validation of Qualitative Costs . . . . . 29

2.1 Preliminary Model of the Gains of IS . . . . . 32
2.2 Preliminary Model of the Costs of IS . . . . . 36
2.3 Changes of Preliminary Model . . . . . 38

# References


Baum, H.-G. (2013). *Strategisches controlling* (5., überarb. und erg. Aufl.). Elektronische Ressource. Stuttgart: Schäffer-Poeschel.

Bliss, W. G. (2004). Cost of employee turnover. *The Advisor*.

Capraro, M. & Riehle. (2016). Inner source definition, benefits, and challenges. *ACM Computing Surveys (CSUR)*, *49*(4), 67.

Chi, C. G. & Gursoy, D. (2009). Employee satisfaction, customer satisfaction, and financial performance: an empirical examination. *International Journal of Hospitality Management*, *28*(2), 245–253.

Devanbu, P., Karstu, S., Melo, W. & Thomas, W. (1996). Analytical and empirical evaluation of software reuse metrics. In *Proceedings of the 18th international conference on software engineering* (pp. 189–199). IEEE Computer Society.

Dinkelacker, J., Garg, P. K., Miller, R. & Nelson, D. (2002). Progressive open source. In *Proceedings of the 24th international conference on software engineering* (pp. 177–184). ICSE '02. Orlando, Florida: ACM. doi:10.1145/581339.581363

Elaug, M. (1965). The rate of return on investment in education in great britain. *The Manchester School*, *33*(3), 205–251.

Emam, K. (2003). Return on investment models.

Frakes, W. & Terry, C. (1996). Software reuse: metrics and models. *ACM Computing Surveys (CSUR)*, *28*(2), 415–435.

Gaughan, G., Fitzgerald, B. & Shaikh, M. (2009). An examination of the use of open source software processes as a global software development solution for commercial software engineering. In *Software engineering and advanced applications, 2009. seaa'09. 35th euromicro conference on* (pp. 20–27). IEEE.

Goldin, D. (1999). Goldin stands by 'faster, better, cheaper' credo. https://web.archive.org/web/20080723151633/http://www.space.com/news/goldin_nasa_991214.html. Accessed: 2017-09-09.

Gurbani, V. K., Garvert, A. & Herbsleb, J. D. (2006). A case study of a corporate open source development model. In *Proceedings of the 28th international conference on software engineering* (pp. 472–481). ACM.

Gurbani, V. K., Garvert, A. & Herbsleb, J. D. (2010). Managing a corporate open source software asset. *Communications of the ACM*, *53*(2), 155–159.

Hansen, W. L. (1963). Total and private rates of return to investment in schooling. *Journal of Political Economy*, *71*(2), 128–140.

Harter, J. K., Schmidt, F. L. & Keyes, C. L. (2003). Well-being in the workplace and its relationship to business outcomes: a review of the gallup studies. *Flourishing: Positive psychology and the life well-lived*, *2*, 205–224.

Höst, M., Stol, K.-J. & Orucevic-Alagic, A. (2014). Inner source project management. In *Software project management in a changing world* (pp. 343–369). Springer.

Jones, T. C. (1984). Reusability in programming: a survey of the state of the art. *IEEE Transactions on Software Engineering*, *SE-10*(5), 488–494. doi:10.1109/TSE.1984.5010271

K., A. (2006). Google's "20 percent time"in action. https://googleblog.blogspot.de/2006/05/googles-20-percent-time-in-action.html. Accessed: 2017-01-28.

Kirkpatrick, D. L. (1998). *Another look at evaluating training programs: fifty articles from training & development and technical training: magazines cover the essentials of evaluation and return-on-investment*. American Society for Training & Development Alexandria, VA.

Lee, S., Aos, S., Drake, E., Pennucci, A., Miller, M. & Anderson, L. (2012). Return on investment: evidence-based options to improve statewide outcomes. *Olympia: Washington State Institute for Public Policy*.

Lindman, J., Riepula, M., Rossi, M. & Marttiin, P. (2013). Open source technology in intra-organisational software development—private markets or local libraries. In J. S. Z. Eriksson Lundström, M. Wiberg, S. Hrastinski, M. Edenius & P. J. Ågerfalk (Eds.), *Managing open innovation technologies* (pp. 107–121). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-31650-0_7

Lindman, J., Rossi, M. & Marttiin, P. (2010). Open source technology changes intra-organizational systems development - a tale of two companies. (151ff). EICS.

Liston, K. (2006). Literature review methods: point of departure. *Retrieved July*, *31*, 2012.

Martin, K. & Hoffman, B. (2007). An open source approach to developing software in a small organization. *Ieee Software*, *24*(1).

Melian, C., Ammirati, C. B., Garg, P. & Sevon, G. (2002). *Building networks of software communities in a large corporation*. Citeseer.

Melian, C. & Mähring, M. (2008). Lost and gained in translation: adoption of open source software development at hewlett-packard. In *Ifip international conference on open source systems* (pp. 93–104). Springer.

Mohagheghi, P., Conradi, R., Killi, O. M. & Schwarz, H. (2004). An empirical study of software reuse vs. defect-density and stability. In *Proceedings of*

*the 26th international conference on software engineering* (pp. 282–292). IEEE Computer Society.

Morgan, L., Feller, J. & Finnegan, P. (2011). Exploring inner source as a form of intra-organisational open innovation.

Neus, A. & Scherf, P. (2005). Opening minds: cultural change with the introduction of open-source collaboration methods. *IBM Systems Journal*, *44*(2), 215–225.

Nguyen, V., Deeds-Rubin, S., Tan, T. & Boehm, B. (2007). A sloc counting standard. In *Cocomo ii forum* (Vol. 2007, pp. 1–16).

O'Neill, C. (2009). Calculating roi for process improvement. https://www.ibm.com/developerworks/rational/library/edge/09/mar09/oneill/. Accessed: 2017-01-28.

Pearce, J. M. (2015). Return on investment for open source scientific hardware development. *Science and Public Policy*, *43*(2), 192–195.

Pearce, M. (2017). How to conduct a literature review: types of literature reviews. http://guides.lib.ua.edu/literaturereview. Accessed: 2017-09-03.

Phillips, J. J. (2012). *Return on investment in training and performance improvement programs*. Routledge.

Phillips, J. J. & Schirmer, F. C. (2008). *Return on investment in der personalentwicklung: der 5-stufen-evaluationsprozess*. Springer-Verlag.

Pine, R. & Tart, K. (2007). Return on investment: benefits and challenges of a baccalaureate nurse residency program. *Nursing Economics*, *25*(1), 13.

Plaza, B. (2006). The return on investment of the guggenheim museum bilbao. *International journal of urban and regional research*, *30*(2), 452–467.

Poulin, J. S. & Caruso, J. M. (1993). A reuse metrics and return on investment model. In *[1993] proceedings advances in software reuse* (pp. 152–166). doi:10.1109/ASR.1993.291707

Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy*, *12*(3), 23–49.

Riehle. (2016). *An example charter for inner source programs* (tech. rep. No. CS-2016-05). Technische Fakultät.

Riehle, Capraro, M., Kips, D. & Horn, L. (2016). Inner source in platform-based product engineering. *IEEE Transactions on Software Engineering*, *PP*(99), 1–1. doi:10.1109/TSE.2016.2554553

Riehle, Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi, Y., Naveh, B. & Odenwald, T. (2009). Open collaboration within corporations using software forges. *IEEE software*, *26*(2), 52–58.

Riehle & Kips, D. (2012). Geplanter inner source: ein weg zur profit-center-übergreifenden wiederverwendung.

Schmalen, H. & Pechtl, H. (2013). *Grundlagen und probleme der betriebswirtschaft*. Schäffer-Poeschel.

Smith, P. & Garber-Brown, C. (2007). Traveling the open road: using open source practices to transform our organization. In *Agile conference (agile), 2007* (pp. 156–161). IEEE.

Sonnenreich, W., Albanese, J. & Stout, B. (2006). Return on security investment (rosi)-a practical quantitative model. *Journal of Research and practice in Information Technology*, *38*(1), 45–56.

Soora, S. K. (2014). A framework for software reuse and research challenges. *Int J Adv Res Comput Sci Softw Eng*, *4*(10).

Stol, K.-J., Avgeriou, P., Babar, M. A., Lucas, Y. & Fitzgerald, B. (2014). Key factors for adopting inner source. *ACM Trans. Softw. Eng. Methodol. 23*(2), 18:1–18:35. doi:10.1145/2533685

Stol, K.-J., Babar, M. A., Avgeriou, P. & Fitzgerald, B. (2011). A comparative study of challenges in integrating open source software and inner source software. *Information and Software Technology*, *53*(12), 1319–1336.

Stol, K.-J. & Fitzgerald, B. (2014). Inner source—adopting open source development practices within organizations: a tutorial.

van der Linden, F., Lundell, B. & Marttiin, P. (2009). Commodification of industrial software: a case for open source. *IEEE software*, *26*(4).

Vitharana, P., King, J. & Chapman, H. (2010). Impact of internal open source development on reuse: participatory reuse in action. *J. Manage. Inf. Syst. 27*(2), 277–304. doi:10.2753/MIS0742-1222270209

Wesselius, J. (2008). The bazaar inside the cathedral: business models for internal markets. *IEEE Software*, *25*(3), 60–66. doi:10.1109/MS.2008.79

Wiegers, K. E. (2002). Seven truths about peer reviews. *Cutter IT Journal*, *15*(7), 31–37.

Yu, S. & Zhou, S. (2010). A survey on metric of software complexity. In *Information management and engineering (icime), 2010 the 2nd ieee international conference on* (pp. 352–356). IEEE.