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

Julian Hirsch
MASTER THESIS

Conceptualizing a Management Accounting Solution for Inner Source Software Engineering

Submitted on May 27, 2022

Supervisor: Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander University 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.

_____

Nürnberg, May 27, 2022

License

_____

Nürnberg, May 27, 2022

Abstract

Open Source represents a non-hierarchical and collective approach to software development, that has seen unprecedented success and rapid adoption across a multitude of fields. Inner Source is an attempt to siphon some of the advantages of the Open Source paradigm by applying it to internal closed development projects.

However, the well-established uncontrolled and free-flowing nature of Open Source poses a challenge for conventional management systems, that are traditionally well-monitored and run in a top-down manner.

Using both sources from literature as well as interviews with industry practitioners, this work investigates potential approaches to combine management accounting and Inner Sourcing in a meaningful way and presents two design artifacts. These prototypes are demonstrated to be used for monitoring and controlling active projects, and to determine the viability of Inner Sourcing potential future projects, respectively.

Agenda

1 Introduction

1.1 Open Source and Inner Source Software Development

Open Source software development is **based on the principles of free access and open collaboration**. The **collective and non-hierarchical** nature of work has helped propel Open Source from a niche phenomenon to a dominant force in a number of markets with many years of double or triple digit growth rates and companies worth billions of US-dollars in market capitalization (Deshpande & Riehle, 2008; Volpi, 2019; Ahlawat, Boyne, Herz, Schmieg, & Stephan, 2021).

The underlying **Open Source definition** that this branch of software development is based on takes two forms, which are deeply interconnected. They are, on one hand, the rights granted by the software's license. On the other hand, following from these basic rights emerged a way of collaboration that is distinct from other proprietary software development and that has become part of the term Open Source as much as the licensing itself.

Generally, in order to qualify as Open Source, a software license must be permissive enough to allow open access, adaptation and improvement on a piece of code. Thus, the set of rules in a license also act as universal rights for any contributor.

Perens (1998) points out **three fundamental rights** in particular that make programmers want to contribute to Open Source. They are:

- The right to have access to the software's source code
- The right to make and distribute copies of the program
- The right to make improvements to the program

These user (and by extension, developer-) rights established in Open Source create a **parity between all contributors** that is unlike traditional software projects. **This parity is then directly reflected in the way the collaborators work together.** The bottom-up approach to software development creates an inversion of power from traditional guided development processes. With parity established, collaboration must be structured around shared principles, so-called best practices. In Open Source development these best practices revolve around the **collaboration principles of egalitarian, meritocratic, and self-organizing work** (Riehle et al., 2009).

Stol, Avgeriou, Babar, Lucas, and Fitzgerald (2014) further collected **seven Open Source best practices**:

- Universal access to development artifacts
- Transparent development environment
- Peer-review of contributions
- Informal communication channels
- Self-selection of contributors
- Frequent releases and early feedback
- "Around the clock" development

Instead of being assigned a single, pre-determined task, as is usually the case in traditional development, Open Source enables contributors to choose what and how they contribute to a

project, and are a lot more flexible in what way they choose to utilize their time and skills (Gillies, 2016). **Peer-review, unobstructed communication and transparency** of development allow for frequent releases while mitigating flaws and revisions after release. In fact, it has long been established that **Open Source processes can have positive impacts on software quality and reliability**, due to the non-standard work structure and collaborative spirit (Bosio, Bev Littlewood, L. Strigini, & M. J. Newby, 2002; Lawrie & Gacek, 2002).

With these advantages to Open Source software development, **private companies quickly became interested in branching into the Open Source space**. Their goals, however, appear to be diametrically opposed, as companies would naturally still seek to extract economic value out of their development and protect their intellectual property (Riehle, 2009). Permissive licenses would not allow for direct monetization, as per the Open Source definition, any Open Source licensed software has to be freely distributed, along with all of its derivative works (Open Source Initiative, 2007). This **mismatch between Open Source working principles, proprietary licensing and community-driven development will probably remain unresolvable**, which **forces companies to choose either two of the three**, while mitigating the impact of losing the third.

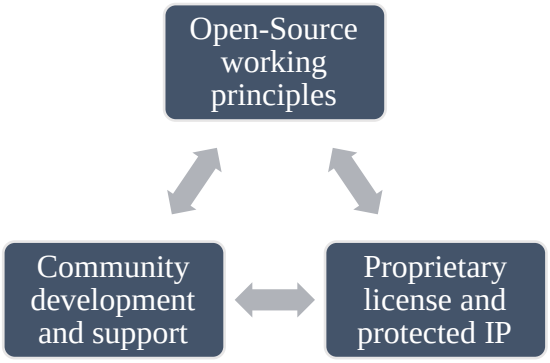


Figure 1: Decision triangle in Open and Inner Source

These contrasting ideas couldn't easily be combined. As a result, a number of new approaches and business models were developed to solve this perceived dichotomy between inherently open development and profit-driven private markets. A number of solutions were formulated under the umbrella term Commercial Open Source (Riehle, 2009). The innovation brought about provided a platform for economic disruption in the software marketplace and gave rise to corporate Open Source collaboration as well as entire Open Source departments. The participating companies are forced to forego revenue and give up some of their intellectual property but may receive other benefits that compensate for this. In some cases, their business models are hybrids between Open and Closed Source, and create revenue from software adjacent services, such as an improved distribution system or adapting the codebase to a specific customer on demand (Watson, Boudreau, York, Greiner, & Wynn, 2008).

Another approach to capture some of the benefits of Open Source is the **limitation of Open Source practices and principles to a closed system within organizations – appropriately named Inner Source** (Capraro & Riehle, 2017). This limitation to a single organization seems to be a stark contrast to the ideas of open collaboration described before. However, it enables practitioners to **benefit from many of the upsides** of collaborative development, **without using Open Source licensing** and **without risk of losing their intellectual property** in software. Of course, at the same time**, Inner Source sacrifices mechanisms like community development** and crowd-sourced support, in favor of the aforementioned work principles and best practices. In supplementing their closed source development processes with these Open Source

principles, companies hope to improve compliance with schedules, higher reliability and more efficient development practices, and to generally further develop and improve their software product platforms (Riehle, Capraro, Kips, & Horn, 2016; Morgan, Gleasure, Baiyere, & Dang, 2021).

With these rather disruptive changes to their processes, **a number of (potential) problems when moving towards Inner Source have been identified** by the likes of Riehle et al (2016) and Stol et al (2011) and (2014), respectively. To combat these**, steering and controlling subsystems are most likely required to make Inner Source a useful integration into most companies** – either as part of their overarching management accounting software, or as a standalone component used and maintained by Inner Source practitioners.

1.2 Dimensions of Management and Management Accounting

Management provides the core functions of planning, organization, commanding, coordination and monitoring, as described by Charifzadeh and Taschner (2017). Among these management responsibilities, all monitoring and documenting of processes relating to income, expenditures, and cash flow is handled by an accounting subsystem.

Within accounting, **financial accounting represents the external reporting within clear authoritative guidelines**, i.e., within the framework defined by accounting standards such as US GAAP (USA), HGB (Germany), and IFRS (International). Its purpose is to be a documentation process to ensure compliance with these regulations. Recording only quantitative, financial information, accounting systems track the flow of all business transactions that directly affect stocks of goods and tangible resources. As a rule of thumb, **financial accounting aims to disclose only what is strictly necessary and mandated**, because this information may be available to external parties, such as competitors, creditors and authorities.[1]

Its counterpart, **management accounting acts as the internal reporting and controlling tool** used to supplement and guide management decision making. **It is a pure management subsystem**, and unlike financial accounting, management accounting is **not bound by country-specific laws and tax codes.** As such, the reporting dimension of management accounting can go beyond documentation of facts and include management-specific information like projections, forecasts, deviations from prior forecasts as well as error margins.

This distinction can be most easily illustrated using one of the most common forms of financial accounting documentation – the balance sheet, which includes financial information documenting a company's total assets, equity, and liabilities. And **where the balance sheet in financial accounting represents solely the financial status at the time of annual reporting, management accounting extends it to serve its own function**. To fulfill internal information requirements, the balance sheet in this form includes prior forecasts, deviations, and comparisons to last year's items.

In the following example, columns highlighted in grey are found in reporting by financial accounting, whereas ones in blue are customarily added for internal purposes at management accounting.

[1] Publication obligations are subject to local laws and can vary depending on the legal form (§325 HGB), structure of liability (§ 264a HGB), and the size of a company (§ 1 PublG), as well as jurisdiction – examples are taken from German tax law.

	CURRENT YEAR				
	Actual	**Forecast**	**Diff**	**Diff%**	**Prior Year**
ASSETS					
Non-Current Assets					
Value, Depreciation, Amortization, Investments	415 228	414 500	728	0,2%	379 250
Current Assets					
Inventory, Debtors, Loans, Bank, Cash	958 756	833 589	125 167	15,0%	618 533
Total Assets	**1 373 984**	**1 248 089**	**125 895**	**10,1%**	**997 783**
EQUITY & LIABILITIES					
Equity					
Shareholders' Contributions + Retained Earnings	769 452	789 289	-19 836	-2,5%	543 889
Non-Current Liabilities					
Long Term Liabilities	216 250	168 800	47 450	28,1%	200 000
Current Liabilities					
Creditors, Accruals, Sales Tax, Dividends	388 283	290 000	98 283	33,9%	253 894
Total Liabilities	**604 532**	**458 800**	**145 732**	**31,8%**	**453 894**
Total Equity & Liabilities	**1 373 984**	**1 248 089**	**125 896**	**10,1%**	**997 783**

Figure 2: Illustrative Balance Sheet Statement used in Financial Accounting and Management Reporting[2]

Beyond the monitoring of a company's overall financial information, management accounting is also directly involved in a firm's pricing process. Evidently, pricing is critical to sustain a business' profitability, as setting prices incorrectly risks losing out on revenue in the short-term, and market share in the long run – making pricing strategy a crucial part of a firm's overall strategic alignment (Lancioni, 2005; Johansson, Hallberg, Hinterhuber, Zbaracki, & Liozu, 2012). In addition to the above described financial data, **the pricing process in management accounting also includes imputed costs such as markups for risk and investments, as well as profit margin to determine the product's final price to the customer** Charifzadeh & Taschner, 2017; Jung & Han, 2017).

Pricing methods can take three basic forms: cost-plus-pricing, value-based pricing, competitive pricing, as well as variations thereof. Barring the last, which focuses on the market to derive one's own price tag, **pricing generally requires clear knowledge of one's production cost and the customer-value created.** Since although companies may choose one method over another to set their prices, **if cost of production is higher than the value towards the potential customers, the product cannot be viable**, making both metrics vitally important regardless of what pricing method is chosen (Charifzadeh & Taschner, 2017).

Further, in addition to the aggregation of a company's financial information, management accounting defines, documents, and monitors non-financial business metrics, depending on a company's specific use cases and management requirements. Since reporting is usually only

[2] Typically, a balance sheet in management accounting would include one or more further statements. These do follow the same structure and items as Figure 2 and display the same financial overview for a different period of time, usually the current month as well as year-to-date or year-by-year views. To ensure readability, these additional views were truncated, and the individual asset, equity and liability items consolidated. The balance sheet as presented is only intended to illustrate the difference in information between financial and management accounting.

made available to internal stakeholders, it often readily includes business-critical and confidential information like internal good and IP flows, resource allocation (as well as management) and resource utilization (Charifzadeh & Taschner, 2017).

One area, where both the financial and non-financial aspects of management accounting are combined, is during strategic planning and sourcing. More specifically, **management accounting aids in the strategic decision between designing and developing products in-house or buying them from a third party – also known as a make-or-buy decision**. Most often used to avoid higher cost, management may choose to outsource some production to stay competitive or increase potential margins (Cánez, Platts, & Probert, 2000).

According to an analysis run by Medina-Serrano, González-Ramírez, Gasco-Gasco, and Llopis-Taverner (2020), further reasons for purchasing ready-made solutions may include higher quality and reliability, lack of in-house expertise, resources or flexibility, insufficient scaling capabilities or disproportionate required investment in research and development. On the other hand, outsourcing can risk draining in-house competences (human factors) and degrade comparative advantages when used for one's core processes (technological factors). Crucially, the researchers further argue that there is **no need to constrain sourcing to either "Make" or "Buy"** and argue that **hybrid solutions like "make-and-buy" should be considered** as possible outcomes – opening the door to fruitful strategic partnerships in sourcing and development.

To find the correct course of action for a sourcing decision, **any tool analyzing the impact of a potential make-or-buy decision has to account for both its financial and non-financial factors**, as well as weighing their potential effect on the long-term strategic outlook. The make-or-buy framework elaborated by Medina-Serrano et al. (2020) elaborates the following general-purpose areas and factors that should be considered.

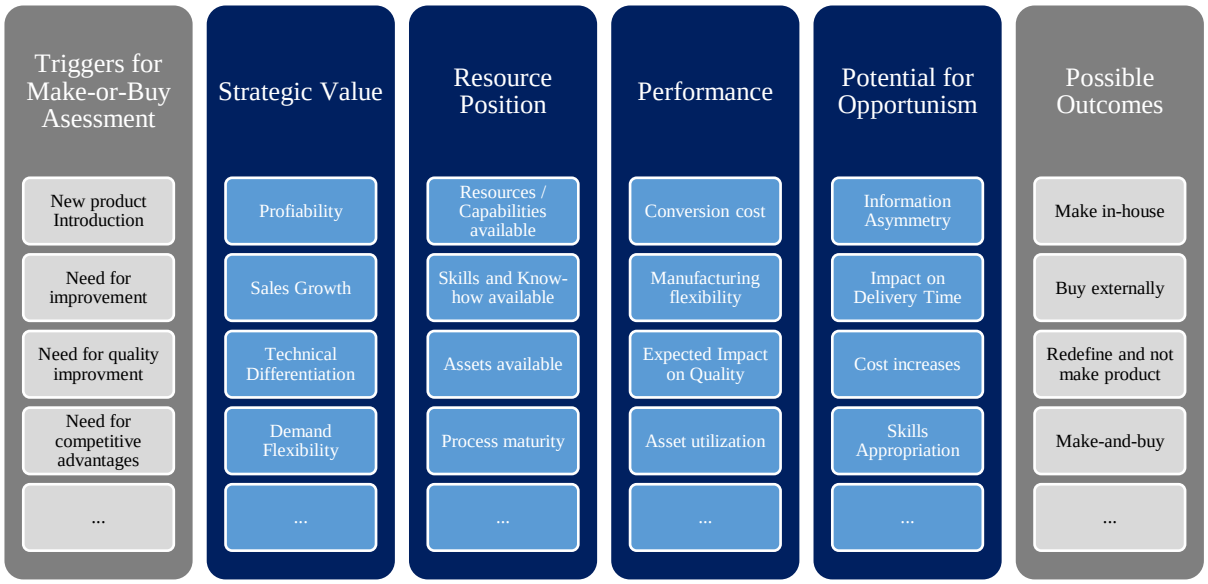


Figure 3: Make-or-buy framework
Source: Own illustration adapted from Medina-Serrano et al., 2020)

Due to the legal ambiguity surrounding management systems, the **roles and responsibilities of management accounting don't follow a narrow definition**, which is also reflected in differing nomenclatures. In Central Europe, for instance, management accounting is generally referred to as controlling, giving higher importance to the control and feedback aspect of management accounting (Charifzadeh & Taschner, 2017). Most traditionally however, **management control is considered a subsystem, that, together with management reporting, builds**

the function of management accounting. As responsibilities of controllers have expanded, this distinction has become less clear over time. This research aims to use the specific terms management accounting, (management) reporting, and (management) control respectively, where applicable. However, as concepts and responsibilities may overlap, the distinction cannot always be resolved finally. The following image illustrates the definition used within this academic work.

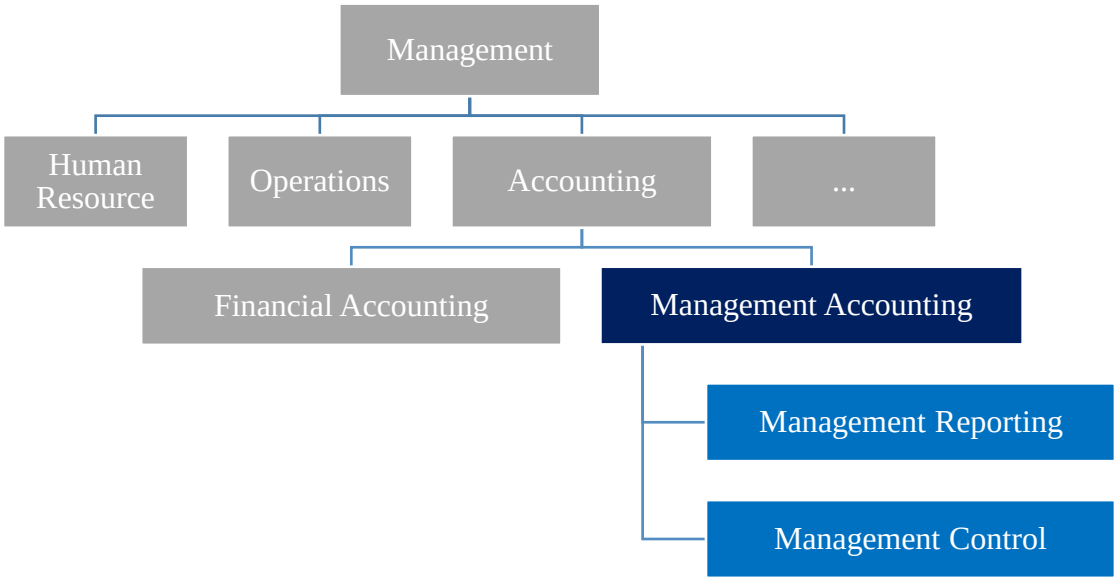


Figure 4: Classification and differentiation of management accounting within management
Source: Own extended illustration modeled after Charifzadeh, Taschner (2017)

To prevent confusion by the illustration above, it has to be noted that, from an organizational viewpoint, **management accounting and control are rarely centrally organized** in the same way financial accounting usually is. Information and responsibility for management accounting **tasks are spread across departments and hierarchical levels**, such as project management, product management, sales, and sometimes individual low-level employees. And as a management tool, management accounting supports and reinforces the general hierarchical structure most management systems have, with information flowing upward towards middle and upper management and being aggregated and generalized along the way. Complementing this bottom-up movement of information, management uses the information provided to steer and control the process in a top-down manner.

Since management accounting is not subject to any pre-defined regulation, **managers and controllers are left to decide on the nature of their reporting**, which information to include in reports for which stakeholder, how to organize it and at how often to produce it.

Specifically, managers and controllers are free to decide on the:

- contents and level of detail
- scope
- form and structure of the reports
- frequency of reporting
- target audience and authorized viewers

As a commonly used example, management accounting can create a regular reporting of business performance – either of the business as a whole or an individual unit.

Items	Budget	Actual	Diff	Diff%	Variance
Revenues	57 000	60 000	3 000	5,3%	Favorable
Cost of Goods Sold	40 000	43 400	3 400	8,5%	Unfavorable
Wages	6 700	7 000	300	4,5%	Unfavorable
General & Admin	1 300	900	-400	-30,8%	Favorable
Other fixed costs	2 500	2 500	0	0,0%	-
Operating Income	**6 500**	**6 200**	**-300**	**-4,6%**	**Unfavorable**

Figure 5: Weekly Performance Report
Source: Own adaptation after Charifzadeh and Taschner (2017)

In this instance, the amount of content and the level of detail are relatively limited, which allows an easy overview of the most important and consolidated financial information to detect deviations from the budget.[3] The scope is either the whole organization or an organizational unit, with the form and structure serving as a quick overview for the target audience – upper management. The frequency of reporting is weekly, which leaves ample room for more in-depth analyses when irregularities are detected.

With this **decision power over what, when and how often to report**, management reporting also heavily influences categorization and prioritization of available information, **which directly affects operations and strategic decisions**.

As a decision making tool, a management accounting system provides the necessary information for analyzing the raw data collected from the subgroup to measure performance of current business operations and serves as the basis for further planning. This **duality of purpose is often reflected in the distinction between operative and strategic controlling, i.e., the analysis and control of performance compared to the long-term alignment with strategic goals.** Evidently, in some cases it may be preferable to accept suboptimal performance on any one particular endeavor in order to guarantee long-term strategic success. As a standard example, it is reasonable to risk a loss on a project to avoid losing a customer that is high-revenue or otherwise strategically relevant.

As mentioned, risks should be mitigated during the planning phase and thus already considered by management accounting – such as during the markup in price determination. However, when these challenges arise during the execution phase, they have to be detected and dealt with as little disruption as possible. In order to ensure that an operation still reaches its strategic goals, operative management accounting goes beyond documentation and reporting – which is where the controlling element of management accounting takes center stage. **Where management reporting acts as an aggregation and analysis tool for monitoring and planning business processes, management control provides the tools to enact corrective action to achieve the goals when necessary.**

Strategic controlling is an integral part of the planning process of any major project or change in processes as it helps in analyzing the different options and offers input for continuous improvement. This action of monitoring and providing corrective feedback into a system is usually referred to as the controlling dimension of management accounting.

Structurally, any controlling system consists of three components, that interact with and act on the controlee: A measuring device (detector), an assessor that determines discrepancies between

[3] Note that the report does not include details on the cause of the deviations and the decrease in income while revenues are up. This would have to be the subject of further reporting, if the decisionmakers find it warranted.

the observed and planned values, and an effector that provides feedback and takes corrective action if necessary.

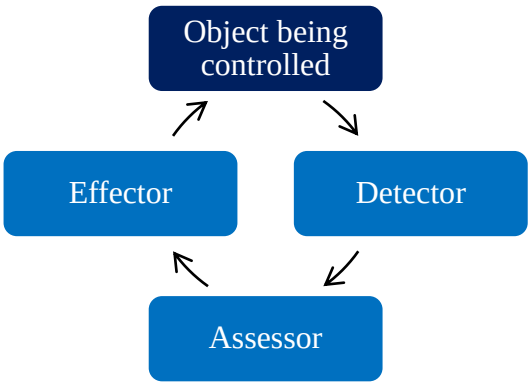


Figure 6: Operative Control System
Source: Own illustration after Charifzadeh & Taschner, 2017; Anthony & Govindarajan, 2014

Summing up, management accounting tasks are deeply interwoven with an organization's entire operation and are carried out across several organizational units. The main distinction identified within the tasks is time – with some tasks taking place in the planning phase before a project is started, and others used for reporting and controlling currently running projects.

The main aspects of a management accounting and controlling system, grouped by when in the project lifecycle they take place, are:

1. **Monitoring, Performance Measurement and Corrective Action** – for ongoing projects
2. **Planning, Preparation, Budgeting, Resource Allocation** – for potential future projects

1.3 Management Accounting in Software Engineering

Management accounting for software engineering most traditionally takes the form of general project controlling, as a project from a management perspective mirrors what the development process entails from an engineering point of view. This seems logical, as most management accounting action in software engineering is operative.

The necessity for a controlling infrastructure in software is quite apparent: **software development processes can quickly become complex and strenuous to monitor**. Projects are **rarely finished as originally planned, in schedule, effort and result** – and this observation is neither new nor expected to change dramatically. For this reason, organizations engaged in software development are always looking for ways to improve their software development processes and iteratively improve upon their development projects (Deephouse, Mukhopadhyay, Goldenson, & Kellner, 1995; Bloch, Blumberg, & Laartz, 2012; Budzier & Flyvbjerg, 2012; Dingsoyr et al., 2019).

Time, cost, and quality – it's these three key performance criteria named the Triple Constraint, also known as Iron Triangle, that determine the priorities and boundaries of project management (Pollack, Helm, & Adler, 2018). An aggregation analysis of project cost, schedule and outcome deviations elaborated by McKinsey's Bloch et al. (2012) shows that software projects are often not finished within budget or schedule, but reports better average compliance with quality requirements than in non-software projects. These statistics are suggestive that software projects rather sacrifice deadlines and accept higher effort than lower-quality results.

Project Type	Average cost overrun	Average schedule overrun	Average benefits shortfall
Software	66	33	17
Non-software	43	3,6	133
Total	45	7	56

Table 1:Percentage of IT projects with given issue (budget > $15 million in 2010)
Source: Own illustration after McKinsey-Oxford study in Bloch et al., 2012

According to Kuster et al (2015), **project controlling's mission is to "describe the rules that are used in project management to ensure that the project goals are achieved**". Thus, its goal is, by extension, **to make different projects consistent across an organization** to improve overall project outcomes. This consistency of rules can, in turn, ease the transition between consecutive projects and allow for collaboration between concurrent ones.

The basic aspects of project controlling are well-established and remain largely unchanged: **every project has to be controlled in terms of its cost as well as concerning its processes**, including scheduling as well as project outcome and process quality (Larson & Gobeli, 1989). And – with some adaptation – these hold true in software development projects as well (Boehm & Papaccio, 1988; Rook, 1986; Filieri et al., 2015).

1.3.1 Cost controlling

Regarding the cost of software development, **there are two well-established approaches to control development projects** – namely the influence-function and cost-distribution approaches described originally by Boehm and Papaccio (1988). In conjunction, these two perspectives work together to create an understanding of software cost. In more detail:

1. The **influence-function – or black-box approach** – acts as an overarching **comparison tool between distinct software projects** and attempts to **identify the influence that non-financial project inputs have on the overall project outcome and cost**. As such, this perspective **describes the projects in terms of objectives, methods employed, time and personnel constraints, as well as skill level**. From this aggregated data, the method **tries to draw correlations and explore causations** with higher or lower software cost (or quality), which can in turn be used to identify and mitigate cost-drivers to improve processes long-term. It has to be noted that, with a reasonable number of predictors, even with just one outcome variable like cost, this type of multiple regression analysis requires a large number of input observations to yield statistically significant results at a satisfactory power.[4] Critically, if this method is used for more than one outcome metric, the solution space would require an even larger dataset, to make any correlations statistically significant.

   Another main drawback of this approach is its inability to create short-term improvements to a system, while requiring additional documentation and the establishment of

[4] Breakdown of the sample sizes required for a multiple regression analysis measuring the effect between an exemplary five explanatory and one outcome variable (like cost) and their resulting powers, anticipating an effect size (f^2) of either 0.15 or 0.35 and a desired probability level (p) of at least 0.05:

Expected Statistical Power	**Required Sample Size at f^2 of 0.35 (high)** Optimistic Estimate	**Required Sample Size at f^2 of 0.15 (medium)** Conversative Estimate
0.50	26	52
0.80	43	91
0.90	53	116
0.95	63	138

common evaluation factors. Due to this, practitioners may be hesitant to dedicate the necessary time to the required record-keeping.

2. The **cost-distribution – or glass-box approach** – focuses in more detail on the individual sources of project cost. As an **aggregative costing method**, it relies on detailed documentation of all cost factors, as well as when, how, and by whom the cost was incurred.

   Specifically, the glass-box approach puts **emphasis on analyzing the distribution of these types of cost**:

   a. **Value-added or corrective activity** – i.e., time spent on development, implementation and improvement vs. documentation and maintenance effort
   b. **Capital or labor cost** – Capital Expenditure vs. Operating Expenditure[5]
   c. **Project phase or activity** – planning, implementation, testing, maintenance, etc.

   It may be useful to adapt the and add to the classification types proposed by the original authors to apply the method to new development methods and potentially enable collaborative action.

   This collective cost approach is quite intuitive, but will struggle to capture the overall cost structure when the individual cost is not documented in enough detail. In these application cases, similar to the black-box approach, **the glass-box approach will require upfront investment and changes to the way development activity is documented** before achieving benefit.

Experimental approaches confirm the viability and applicability of both methods, showing that a better understanding of cost development and correlation with project factors results in increased effectiveness and better control over cost while maintaining quality standards. **Analyses of controlled experiments have shown similar gains to productivity for both methods**. The aggregation of historical project data when identifying cost factors and risk has also been shown to allow a significant cost reduction long-term. However, the black-box approach has been associated with high variance in its effect on productivity – due to differences in adaptation by the development teams.

1.3.2 Process Controlling

Processes can be distinctly more difficult to control, since they are often less clearly defined and improvement metrics less obvious. To classify these software development processes one could rely on the **Capability Maturity Model (CMM)**, first established by the Software Engineering Institute of Carnegie Mellon University in 1991 and expanded and added-to in later iterations (Paulk, 2009). The model **groups software development operations into five ascending levels, from entirely chaotic to highly organized** and optimized. They detail clear priorities on which improvement activities should be implemented first, as they will most likely prove to be most effective in leading to a successfully performing software operation.

The levels, in ascending order of maturity and complexity, are:

1. **Ad hoc / chaotic** – few processes are defined; success hinges on individual contributions and effort.

[5] Usually, any movement towards higher percentage of OpEx is welcomed, as it reduces upfront investment and hedges risk – which can also be seen in movement towards Cloud, IaaS and PaaS solutions in software engineering projects (Andreo, Calà, and Bosch (2021).

2. **Repeatable** – basic project management processes are defined to monitor cost, scheduling and product viability.
3. **Defined** – processes (both management and engineering) are standardized and integrated into the company's wider software environment. Projects use standardized company-wide software processes which focus on continuous learning and improvement.
4. **Managed** – comprehensive collection, reporting and controlling of process and product quality attributes; Quantitative analysis and statistics become paramount at this stage, aiming towards evidence-based management structures.
5. **Optimizing** – steady feedback into the processes and constant introduction of innovation enables continuous improvement. Changes in the processes are constantly evaluated using both practical experience and statistics to gauge whether the revisions were merited and are potentially worth expanding further.

In order to improve development performance and output long-term, **companies need to improve through these stages**, which is where the connection between development processes and the involvement of management through reporting and controlling can be drawn. **Level 1 is the only one of the five that may "function" without even a basic reporting and controlling mechanism** – it is also called chaotic after all.

Moving from a chaotic level to Levels 2 and 3 already requires, at the very least, some project management and managerial oversight to avoid non-compliant software as well as discrepancies in delivery time and cost. To achieve a reasonably stable development process, **these stages require technical, business, and financial control** which can be achieved by introducing a low-level management accounting system that quantifies the process and gives management handles for supervision and potential intervention, without completely re-designing the processes established.

To achieve any of the higher levels of organization (Levels 4 or 5) that foster effective development and high-quality output, Paulk, Curtis, Chrissis and Weber (1993) suggest moving towards a process that is standardized and monitored on an organizational level. This **includes establishing process value metrics** measuring both cost and quality parameters of each process unit which should be kept in a well-managed process database. The collection of these metrics should also extend to product quality measures and other adjacent metrics that are integral to the success of the software department or company. Importantly, this maturity level **also requires adequate processing and analysis of the data** collected by dedicated individuals to advise the project members and decision-makers on how to implement data-driven solutions in the process.

This extensive requirement for monitoring, analytics and direct corrective action **necessitates a comprehensive reporting and control system**. In combination with the requirement for company-wide standardization, oversight, and ability for intervention, this suggests the need for a central unit dedicated to the gathering, storage and analysis of software process and product metrics.

1.3.3 Metrics

For it to be able to properly measure performance, and thus success, of a development operation, such a management accounting tool needs to be tailor-made or adapted towards the task of monitoring software development. In order to properly monitor and steer a software project or process in a way that leads to satisfactory outcomes all round, **one needs to establish a set of metrics that define the project and process performance**. According to DeCotiis and Dyer

(2016), **recognizing what constitutes good project performance** is the first important step towards understanding and successfully controlling project operations, and requires rigorous definitions and accurate measurements.

For this reason, the authors established **five basic performance dimensions for engineering projects** in 1977:

1. Manufacturability and business performance
2. Technical performance
3. Efficiency
4. Personal growth experience
5. Technological innovativeness

These include technical, business and human resource dimensions of engineering endeavors and represent a good starting point, around which to structure the metrics for one's management accounting system. It is to be noted, that the metrics by DeCotiis and Dyer are not exhaustive, nor were they meant to be 45 years ago, and encompass a very general notion of 'engineering'. To adapt these to any modern software engineering context requires analyzing the demands of today's industry – extending and adding metrics accordingly.

For software engineering in particular, it can be difficult to gauge efficiency and technical performance specifically, as such aspects are often too complex to be measured directly (Kaner & Bond, 2004). Simple **Lines of Code (LoC)[6] metrics can be used when comparing the size of programs and coding contributions,** but they cannot fully account for the actual production of a software contributor. To create a more holistic benchmark for productivity, Mas y Parareda and Pizka (2007) have adapted this LoC approach to consider two further metrics. The authors **proposed an extension of LoC to "Redundancy-free Source Lines of Code per Effort" as well as an inclusion of "Defects per Source Lines of Code"**. This combination of metrics was found to be a fair and comprehensive measure of productivity in software engineering.

Other **performance and success metrics may be derived from sources such as compliance with legal requirements, industry standards, management and corporate demands** as well as from interviews with the development team and other important stakeholders (Nuseibeh & Easterbrook, 2000; Aurum & Wohlin, 2005). As one example, managers are increasingly concerned with skills management, skill building and employee retention, as they represent some of the most essential resources in software engineering (Bibi, Anwar, & Rana, 2021). **Deciding which of the metrics derived are to be integrated in a management accounting system can heavily influence the usability, scope, and usefulness of the software solution**. This trade-off situation needs to be evaluated as a major design choice in the construction of the software, and may potentially be left for configuration and customization by the system user.

Importantly, there is also a trade-off when opting for a one-size-fits-all system for software controlling. From real-world testing, such approaches either result in high variance estimations or require more inputs than are generally available at the start of a project. Particularly during the early stages of tool development, not all requirements and thus required metrics will be known, which is why the software needs to be adaptable and allow for iterative improvements upon the system (Hihn & Menzies, 2015). Ideally, a tool prototype demonstrates the basic functionality with commonly accessible metrics, which can then be extended with additional complex measures using more intricate processes.

[6] Sometimes also called Source Lines of Code – SLoC

1.3.4 Implementation options

Implementing such a tool can be realized **either as a standalone** software development reporting and control tool, **or by integrating the software engineering monitoring into the general-purpose management accounting environment of the company**. In most scenarios, this decision will hinge on the importance that software has within the company's value chain, i.e., whether software is the main selling point, or a complementary part of the end product.

For implementation of the advanced productivity and code value measures described beforehand, an analysis tool like the Open Source project GrimoireLab could be of use. With a great number of functions surrounding repository mining and software development analytics, the GrimoireLab platform could be used both for a post-hoc analysis as well as a monitoring solution for new code contributions and interface directly with a project's code repository (Dueñas et al., 2021).

To track and visualize all metrics associated with an endeavor, any number of database and business intelligence solutions could be used. However, even though these solutions are generally robust and highly customizable, any custom analysis and change or update thereto requires significant upfront development. As such, especially during early stages of development, a less complex implementation may be warranted. **In industry, project controllers still rely on spreadsheet-based solutions** to monitor and control their projects, even though they are generally replaced by more sophisticated project management tools and database implementations long-term (Raith, Richter, & Lindermeier, 2017). This is mainly because spreadsheets' **widespread adaptability and ease of use allow for rapid iterations** and make it an **ideal tool for prototyping** a robust software solution.

In addition, there are advantages from a change management perspective to implement any software solution within tools or frameworks familiar to the practitioners, i.e., active users. As per Hihn and Menzies (2015), who evaluated the introduction of software costing systems at NASA's Jet Propulsion Laboratory over the course of ten years: The larger an operation is and the more different stakeholders are involved, the harder it becomes for a new tool to be adapted readily and achieve the desired outcome. Therefore, actual adaptation should take precedence over technological perfectionism. **Particularly as a proof of concept, spreadsheet-based solutions may be helpful in convincing stakeholders and making the tool an effective addition to the overall management process**.

Average resolution / conclusion times in calendar days					
Status	Cosmetical	Minor	Major	Critical	**Total**
Assigned to resolved	10,8	16,6	18,2	9,1	16,5
Resolved to closed	6,7	14,1	11,0	5,2	11,5
Assigned to closed	17,4	30,7	30,0	14,3	28,4

Figure 7: Part of a software defects dashboard in project controlling

Akin to the previously-explained influence-function (= black-box) approach, efforts have been made to introduce advanced data analytics methods and machine learning models to estimate expected workload and predict budget overruns, although they remain a niche application (BaniMustafa, 2018; Arndt, 2018). Generally, these models require higher development effort and are often prone to high variance in prediction accuracy which, coupled with the already high variance connected to the black-box approach, could impact their usefulness. They are, however, an option for further development once the prototyping was successful and the models were shown to be practical.

1.4 Relevance of Topic

Any form of Open Source (and transitively Inner Source) development is making use of a wide range of development tools, to enable the collaborative and shared nature of the paradigm, while reaching its technological goals. As a general rule, the increasing number of programs that developers depend on does not seem to have stalled the advancement of Open Source projects (Dueñas et al., 2021).

However, stratifying the Inner Source software development paradigm within a monitoring and reporting system seems to restrict some of the nature of Open Source working principles itself, even more so a controlling system influencing and correcting the process. After all, its basic tenants appear to be opposed to the idea of quantifying, monetizing and being steered, instead focusing on open collaboration across organizational boundaries without major restrictions.

Additionally, introducing a new dimension to one's management accounting system is no small feat either. Especially when the target process does not conform with the structure and scope of a normal organizational unit. Where usually a manager assigns resources and registers expectations in the form of a project plan (and eventually results) in accordance with these inputs, this clear connection is no longer guaranteed in collaborative work environments like Inner Source. As established, many companies do not have access to highly customized management accounting solutions, nor are they required to have such systems at all. For a seemingly niche area such as Inner Source, the upfront investment for a custom management accounting tool, as well as the management overhead caused by such a system, may not be worth the expense. Considering the amount of effort required changing and adapting to a new system in general, it might be questioned whether the system could even achieve its theoretical potential in any given environment.

All of this means that working towards a specialized management reporting and control system for Inner Source despite these challenges requires comprehensible justification and a realizable execution plan. **In essence, why would it even be necessary to add the dimension of management accounting to the Inner Source process in the first place?** And how would it be done?

As touched on beforehand, Open Source methodologies and working principles can have a number of direct impacts when applied on a company's operations – many of which directly translate to Inner Source. Perhaps most crucially, it has long been known that open collaboration and interaction between different organizational units and functions can lead to increased productivity and overall better results (Deephouse et al., 1995). While in the past this has mostly been realized through interaction between cross-functional teams, increased collaboration is also one of the key tenants the Open Source method could bring to organizations. In addition, regarding product and development process quality, Open Source methodologies have the potential to improve security and availability, if properly implemented and managed (Lawrie & Gacek, 2002).

Generally, in the case of Commercial Open Source development, no special management reporting tools are required, because the involvement in Open Source projects does not create revenue directly. Supplementary or adjacent software, which the company sells under a proprietary license, can be handled the same way as any other software – its connection to Open Source is irrelevant from an accounting perspective. On a side note, while reporting may not be necessary, even in the case of Open Source software engineering, there are some advantages to be drawn from a management controlling system (Riehle, 2011).

Inner Source, on the other hand, requires a more formal and complete subsumption into a management accounting system, as **it affects resources and revenue streams more directly than other forms of Commercial Open Source**, differing only in the work methodology and organization. And as management accounting is generally less rigid and customizable and financial accounting, it lends itself more readily to be adapted to a new use case, which is important, as very rigid supervision systems may quickly lose or dampen the advantages of Inner Source.

Perhaps most crucially, accurate pricing and cost control are among the most important parts of any business' ability to continue its operation and sustain itself long-term. Involving Open Source principles and unmonitored Inner Source processes may create ambiguity and inaccurate pricing. **To prove the economic viability of Inner Source and to convince companies to venture into the field, solid management processes and tools are to be established** that enable tracking contributions across usual organizational boundaries.

Management accounting also goes beyond economic analysis in making sure to align the operations with the strategic outlook (Charifzadeh & Taschner, 2017). As such, **if companies seek to implement an Inner Source strategy that involves resource and IP-management, a monitoring and controlling device is without alternative**. In the case of knowledge management for instance, Open Source and Inner Source differ dramatically, and in turn, they have different management requirements attached to them. As Inner Source seeks to protect intellectual property and proprietary knowledge, those could be integrated in a company-wide knowledge management structure (Oun, Blackburn, Olson, & Blessner, 2016).

In most companies, management sets up a hierarchical structure through which it delegates resources and responsibilities. While this is not set in stone, it is somewhat inherent to the management function of commanding and leading, also described by Charifzadeh and Taschner (2017), which also makes it unreasonable (in most cases) to restructure an entire organization around Inner Source. Particularly traditionally operated organizations have a vested interest in keeping a close grip on their processes.

As Inner Source and its open, un-controlled and free-flowing nature does not naturally lend itself to being under strict supervision and control, it does not conform to the usual top-down system of management. A lack of integration with management accounting can pose a significant barrier to entry into the realm of Inner Source, since an absence of information makes any organizational change potentially high-risk to decision-makers. **Depending on the level of engagement and integration, it can cause inefficiencies and errors in existing Inner Source processes, hinder further expansion, or prevent companies from venturing into Inner Source at all**.

The specific reasoning behind connecting Inner Source development and management accounting can be manifold. However, in general, it can be traced back to management's information and control requirements. It is why companies have management accounting systems in place – to monitor and control their operations as well as their strategic alignment, which, as laid out before, can be the difference between continued success and failure of a development project (Deephouse et al., 1995). The all-encompassing nature of management accounting is paramount to this purpose. Only if all aspects of a company are monitored and controlled, can this strategic alignment be guaranteed. **It is only logical therefor, that Inner Source should be subject to the same level of scrutiny as other operations.**

This is further affirmed when viewing Inner Source in the context of the Capability Maturity Model (CMM) – **in order for Inner Source not to be treated like the non-conforming step-**

sibling of traditional development, it needs to adhere to the same process quality and maturity requirements. The CMM already suggests some basic monitoring and project management, as well as an effort of process standardization, to leave the chaotic level and reach one of the lower maturity levels. The **higher maturity levels require even further monitoring, stratification, constant feedback and corrective action**, which can (reasonably) only be achieved by integrating Inner Source processes into company-wide management processes, including accounting.

Recalling the performance dimensions of engineering projects, Inner Sourcing can make a powerful contribution to the success of any project or process, if the endeavor is not only well-understood and supervised with well-defined metrics but is also subject to a functioning corrective system using said metrics. If implemented properly, Inner Sourcing could lead to better business performance through quicker and less error-prone development as well as more efficient detection and correction of inefficiencies. A less restrictive development environment also has the potential to foster employee growth and create a more innovative work environment, creating long-term benefits for the business.

For some cases, **hesitation to implement Inner Source, despite its potential advantages, may stem from legal rather than organizational boundaries**. For many projects, such as government contracts, process compliance takes precedence over faster development and potential innovation. In other instances, **licensing requirements may make it difficult to justify unmonitored interactions and collaboration** with organizational units that are not part of the original certification as is the case for medical products or critical infrastructure. Still, with the advantages of Open Source methods becoming apparent, there is a significant push towards acceptance of FLOSS software products in medical procurement (Reynolds & Wyatt, 2011). With this growing recognition of Open Source solutions in the medical field, it is only reasonable to assume that there are tangible benefits to applying Open Source principles within medical companies as well. Using Inner Source in combination with an adapted management accounting system, many of the risks and certification problems could be mitigated or solved entirely.

Evidently, Inner Source will never be able to capture all the benefits of Open Source – some trade-offs will have to be accepted. By definition, it is constricted within the boundaries of an organization, and it cannot have all the same freedoms as Open Source. But in creating tools that satisfy the management requirements as well as harness the power of Inner Source collaboration, companies get the option to diversify their operations and take advantage of this relatively new development in the realm of software engineering.

Aiming to integrate Inner Source into a management accounting environment to capture the postulated benefits while maintaining flexibility and working advantages is certainly a challenge. Yet, if successful, this evolution and adaptation of existing instruments could prove helpful both for small companies attempting to find their competitive edge, as well as for bigger players to strengthen their position on the market.

2 Requirements Identification

2.1 Interview Review

As **this research aims to conceptualize a practical solution artifact**, it is paramount to identify the problems that real-world applicants of Inner Source are faced with. Thus, and **to derive the requirements they have towards any business or software solution, two interviews with Inner Source practitioners were conducted**.

Both interviewees are involved with the Inner Source processes in medical engineering and **have cooperated with researchers in the past**, indicating their openness to new developments within the growing field of Inner Source. Both of the interlocutors **also have management responsibilities** within what is internal known as "Shared Software Components" and can provide valuable insights into the practices and challenges of an industry example of Inner Source. They have expertise on the development, the inner source processes, as well as the approval of changes to said processes.

As such, **they could be considered the target audience for the proposed management accounting artifact**. In any case, they are important stakeholders, whose requirements should be heard and met to increase the likelihood of adoption.

The first interview, conducted on April 1st 2021 shed light on the perspective of a software architect on the Inner Source topic. He is part of the company's proprietary platform for medical equipment. As a domain platform, it serves as a central hub for the research and development efforts of multiple medical devices and software products of the company, as well as the service associated with any of the solutions. Additionally, the unit is also a user of its own services, meaning its products and software products are also hosted on this platform.

He is the head of the department of software architecture, which also includes the system engineering group, test automation, analytics and testing strategy. His responsibilities are two-fold: They include the strategic orientation, both in technology and software architecture, but he is also in charge of reliability to guarantee seamless operations.

To that end, he is basing his decisions on analytics that help give faster feedback to both developers and management as well as to enable them to find flaws in the process and improve the general workflow. The data they collect include information about code commits, test execution and preparation, for instance, and is then used to give rapid feedback to the developer about their most recent commits and changes. As of the time of the interview, this analytic process is mostly descriptive, with the goal of including predictive monitoring and advanced feedback as well.

Regarding how the development process is measured and controlled, i.e. the metrics that are used, the measurement process is quite openly defined. The main metric used to measure product success is value to the customer – represented by the number of complaints, qualitative analysis of the complaints, extent of the necessary rework, and time until a complaint is finally resolved. Process quality is measured by the general velocity of development and how much the time deviates from prior planning. Feedback is received through two channels, either directly from the platform user (i.e., an internal customer) or indirectly through the end customer using the platform solution. This feedback is used qualitatively to identify flaws.

Regarding code completeness and correctness, a commit is considered correct when all of its assigned tests have been passed. Regarding a potential way to assign value to a commit, the

interviewee proposed assigning a value to the respective user story and considering each commit an equally valuable part of this user story.

When asked about the granularity of code documentation, he explained that there is no consistent style of documentation and coding, even though such practices are encouraged, and positive developments have been observed. However, regarding commit history, and documentation, there is significant architecture and resources available for development teams which makes code contributions viewable, and in theory, traceable.

The interviewee was personally involved in the introduction of Inner Source at the company after identifying the need to exchange code between dedicated business lines and has in-depth knowledge on the internal processes established at the company.

The Inner Source infrastructure at the company is divided in two. On one side, there is "true Inner Source", which is mostly used for internal development tools – open collaboration and communication between software engineers and development staff is encouraged with very little oversight. On the other side, Inner Source is adapted as Healthineers' own Shared Software Components system, SSC for short. What makes SSC different is that these software components are shared and used in the final products that are sold to the customers, not just in internal development. As such, especially in the biomedical engineering field, more oversight is required.

The whole Shared Software Component process is headed by the so-called Steering Committee, also known as SteeCo, which is made up of members of all business lines. If a business unit, department or similar wants to join SSC, they must fill out a fact sheet, which evaluates whether there is potential to Inner Source this component or not. If the result is positive, the proposal is presented to the SteeCo to make the final decision and, if accepted, decides on two guardians for the component, one from research and development and one from project lifecycle management. The guardians are in charge of the component and responsible for all artifacts associated with it, which consists of source code and documentation, but can be extended depending on the type of project – special licensing requirements are typical and manifold in medical device production. When all artifacts are assembling, and a final assessment is passed, the Inner Source component is released as version 1.0 and made part of the company's shared software components.

If developers and stakeholders want to make or request changes to a component within SSC, they cannot make that change directly or even fork the source code. Every potential change has to be proposed or requested through an SSC forum where it is then discussed between stakeholders. If the community decides to accept this proposal, the contribution to the project can be made. Any shared software component that is pulled to a specific domain product or platform (i.e., a product sold to the end customer) is responsible for their version of the code, which means that is takes care of any adaptations to the code and potential bugfixes – pulling a component from SSC creates a separate entity of source code singular to the product. Beyond the previously mentioned discussion forum, there is no direct feedback from the live product to the SSC component and no connection between the different iterations of the SSC component across product lines.

The interviewee regards the topic of Inner Source highly and values its positive effect on product quality and the changes it brings to collaborative processes. He identifies higher development speeds, removal of bottlenecks and smoothing of development efforts through quicker bugfixes as the main advantages. In addition, the Inner Source process gives rise to better development ideas and sparks cooperation in areas, that previously had little collaboration – like unified components for various products across business lines.

The development efforts and contributions made through SSC are projected and organized during the planning process – both when designing the features that have to be realized for the final

product as well as when scheduling developer time. The time spent on an Inner Source component is thus budgeted towards the project first developing it – there is no formal distinction between an Inner Source component or a platform or product component.

At the time of the interview, there was no deliberation between complexity of contribution from different contributors. However, he acknowledges that there is management interest in quantifying IP flows – particularly because of tax implications. Asked about where the value creation of an Inner Source component should be quantified, he argued that only the internal user selling a product to the end customer can actually create value for the company, which is why the valuation should be done there.

On the topic of technical debt, the interviewee sees no direct correlation between Inner Source and higher accrual of technical debt, as the Inner Source process at the company is highly stratified and controlled, much like any other development efforts. As such, technical debt isn't more (or less) of a problem than traditional development, as their Inner Source style is still quite restrictive. There were discussions to loosen this system, by having contributors gather "trust" through good commits, which could give them a preferential status in contributing to some components, but it is not expected to be implemented soon.

On the topic of estimating effort and value of a proposed component through the use of comparable historical data, he considered the idea interesting, but questioned its feasibility in the context of software components that are not sold as standalone products. He contrasted the idea of an Inner Source valuation to the purchasing of software licensing, which gives an immediate plannable cost and value. Inner Source, on the other hand, has rather indirect influences on the cost of a product, mostly through its influence on short-term and long-term resource usage. He mentions the increased efficiency and output of Inner Source components, whose licenses counterpart had high cost in the past, but struggles to reason a way to quantify this change.

The valuation of a component is also different depending on its intended use – commodity features are mostly implemented to maintain customer value, not adding to it. For differentiating and innovative features there is potential for added market value, but its realization depends largely on the acceptance of the customer on the market.

The second interview was conducted on April 7th, 2021, with another key employee behind the Inner Source process at the company . In his main occupation, he is a project manager for a medical device used in radiology. In this position, he is an active user of Shared Software Components. In addition, he is the general program manager for SSC, which entails moderating the Steering Committee, which in turn oversees the addition of components to the SSC program and facilitates continuous improvements to the process involved.

As of the time of the interview, the SSC process does not include a method to valuate each contribution. However, the contributor of an SSC component can make an annotation on their contribution to mark it as strategically important and prevent other internal users from giving it away for free and thereby eroding the value the component gives to another product. Within the general context of software engineering at the company, there are instances of valuating software – most commonly by Lines of Code via percentage breakdowns, but also including code complexity, comparable products and comparable functions.

He is interested in the idea of measuring who contributes and profits to what extent from the Inner Source components in SSC and bringing understandability to the internal transfers of code through transparent pricing between business lines. Currently, there is some traceability of who uses which SSC code through internal process, especially since most code is shared between similar products. However, the system of signing in and indicating when using an SSC component isn't always used properly and usage is not enforced. Additionally, this is only meant to monitor where the code is used in case of bugs and vulnerabilities, or when software-adjacent

work like regulatory studies for medical licenses can be done only once and shared between all instances of that software component. The interviewee argues that using this system for remuneration would stifle adoption of SSC and lose many of the non-monetary benefits. Still, there is vested interest within the parties responsible for SSC at the company to create better information density of Inner Source usage across business lines.

Regarding the granularity of code documentation and traceability, the interviewee points to the extensive legal requirements for documenting and validating software components in the biomedical field. These regulations are not only far-reaching and strict but also changing across different markets. For medical device providers to comply with all of these requirements means building a strict framework for any of their processes, including the Inner Source process realized in SSC. However, he also mentions the different management structures and systems that are present in an organization spanning multiple business lines, resulting in incompatible software development processes, that have to be united when collaborating in Inner Source. In addition, only officially trained personnel is allowed to work on specific medical software, making the integration of Inner Source into the process even more complex and posing a significant barrier to entry for each business line and product team intending to enter the SSC realm at the company.

The documentation of Inner Source contributions at the company is rather loose and incomplete. If documentation about SSC components are created at all, this is not stratified within in a process or generated through tools – meaning the documentation is done manually.

The concrete Inner Source process at the company is highly regulated and stratified, and headed up by the previously mentioned Steering Committee, which evaluates proposed SSC components and appoints guardians for these projects, in the form of a product owner and a guardian. And while there is open access to the Inner Source components, there is no free-for-all development, as it might occur in Open Source. There are strict quality requirements and checks before changes are to be made on a component to guarantee continued functioning of components as well as compliance with regulation. Also, participation in an SSC component at the company makes the users responsible to participate in its maintenance and further development – which has led to internal criticism of the Inner Source implementation.

Each SSC component comes with a number of mandatory documentation objects, such as an engineering requirements specification, test records, code reviews, and other development artifacts, as well a distribution notification, which acts as a guideline for potential users of the component. This includes if and under which conditions the component is published, e.g. for which software environment the component is validated. If other versions are adapted or developed from this component, validation and documents may have to be generated anew.

The interviewee considers Inner Source an important aspect of the company's business, but laments inconsistent acceptance, with some business lines being very active in the process and contributing readily, while others would prefer a ready-made solution without having to contribute themselves.

The contributions are, unlike in Open Source, planned and scheduled for and are part of regular project management and software maintenance in the same development team. Usually, there is no collaboration in the initial build of a component, meaning that the first version is normally built by one org unit and then published to the broader audience. There have been instances where two business lines have collaborated on a development project, however, the organizational structure at the company requires there to be one unit in charge of the development, nevertheless. Again, as the collaborative process is rather loosely structured and supervised, there is no direct way to determine who made which contribution. Version control in Inner Source is realized within Microsoft Team Foundation Server (TFS), with a wish to migrate towards GitHub.

Asked whether a potential quantification of software contributions should be done on the contributing or using side, he expressed a preference for the former. Referring to his colleague's prior (contrary) answer, he commented that quantifying on the side of the user of the component required a lot of confidence and honesty, and expressed doubt whether this approach would work. Rather, he envisioned a subjective quantification procedure that creates trust in all participants.

Regarding technical debt, the interviewee is of the opinion that having guardians overseeing each SSC component is advantageous and avoids many problems. However, he laments lacking quality of accompanying documentation, which, in some cases, can cause problems when auditing and licensing the software. This system is continuously being improved upon, with attempts to make the process more transparent and traceable.

On the topic of potentially using historical data about the output of development teams to estimate the future planned Inner Source projects, he states that there is clearly a connection to be drawn, as similar projects often have similar performance and problems. Nevertheless, he voiced concerns if this could be used to estimate an Inner Source component's contribution to the overall worth of the software, as a similar component can have very different importance to the valuation. He argues for a valuation of Inner Source components as a pool of software, and for using the contributions made to it from different parties to value their individual part in that value.

2.2 Summary and Critical Evaluation

To enable analyzing the information collected, the following is a short summary of the interviewees' points of view. To avoid repetition, the second interviewer's answers are only included where they differ from the first.

Notably, despite the fact that both interviewees are not designated controllers, their management responsibilities are interwoven with the tasks of management accounting. In the first interviewee's function as head of software architecture, his responsibilities are divided between strategic planning and ensuring seamless operation, using analytics to continuously improve upon the processes at the company. The other interviewee is a project manager whose work involves monitoring projects that are connected to Inner Source. And as head of the SSC program at Healthineers, he is partly responsible for the planning and viability of new additions to the SSC pool.

Conclusions according to the interviewees:

Regarding **Inner Source Planning and Execution**, IS projects are usually run by one team and then made available to the SSC pool. Particularly during the first iteration of development, there is no direct emphasis on collaborative development, also due to the fact that internal processes can vary greatly across software engineering teams and departments. In any case, any planning process is to be supported through data, which should be understandable and consistent across project and department lines. Generally, at the company, IS projects are treated like any other development project in both planning and execution. As such, participation in an IS project also mandates the contribution to maintenance efforts.

Feedback into the Inner Source Process is very loosely organized and not a priority at the time of the interview. There is no mandatory traceability of who uses an IS component – and critically - once components are copied from the SSC pool, there is no direct connection between the code repositories and projects. Still, continuous feedback is welcome and evaluated

quantitatively. Nevertheless, adoption of said feedback may be slow as reliability generally takes precedence over innovative changes. The IS practitioners are aided in their work by descriptive analytics tools, with the goal of introduction predictive monitoring and controlling aspects as well.

There is a variety **of metrics used in the documentation of Inner Source** contributions and a number of tools available, even though their actual usage and granularity is inconsistent. Both interviewees agree that subjective and consistent measurement are paramount if a valuation is done. The IS process focuses mainly on measuring product quality and process quality. The main metrics for the product is customer value – measured by the number of complaints, extent of complaints and time until correction and the number of flaws and amount of rework necessary. Whereas the main metrics for the process quality are velocity and accuracy of prior planning predictions as well as percentage of passed unit test.

Importantly, a single code commit rarely has value on its own. Rather, its value is tied to its correctness (verified through tests) and the overall value of the respective user story. Within IS at the company, there is no differentiation between different Inner Source code contributions or evaluation of complexity, however, there are instances of such valuation at the wider company. One possible differentiation of code is to classify whether contributions were made either as corrective action, to provide commodity functionality or to create innovative features.

The **clear advantages of Inner Source** are in smoothing the development process through the removal of bottlenecks, as well as the generally increased development speed and quicker bugfixes. IS is also credit with creating better ideas and heightened potential for cooperation, which in the past has provided high-quality contributions to replace formerly licensed software products.

The **challenges associated with Inner Source** are very particular to the environment at the company. As the company operates within biomedical engineering, their production processes require significant oversight to assure legal compliance, which can be difficult to combine with collaborative development. In addition, there is internal criticism that engagement in an IS project requires the continued participation in its maintenance efforts. If a tracking and remuneration system where to be established, the IS process runs the risk of losing participants and missing out on many of the non-monetary benefits.

2.3 Literature Review

The previously conducted interview review gives a deep but relatively narrow view on the Inner Source process at one company. In order to reflect requirements that pertain to the wider audience of Inner Source software engineering, a **literature analysis** was used **to create more generalizable requirements for a management accounting tool** in this environment. These are to be used in conjunction with the general attributes of a management accounting system in software engineering to build a robust system prototype.

The rights generally bestowed on any contributor and user in Open Source are technically at odds with the goal of quantifying contributions with the aim of determining the appropriate compensation, usually monetary. As one of the fundamental principles of Open Source economics, no royalties or license fees are collected (Perens, 1998). Many, if not all of the advantages of the Open Source (and transitively those of Inner Source) are intrinsically linked to the free and open access, that allows rapid innovation and collaborative success beyond the boundaries of a single organizational unit.

The issue of accounting systems that are incompatible with what they are used for is nothing new. In fact, **accounting principles and systems often lag behind the technological processes required**. Gietzmann (1996) goes as far as alleging that these systems can cause counteracting incentives that "stifle attempts to work cooperatively", and argues for an **increase in flexibility to improve relationships and collaboration**. This suggests that it must not necessarily be a case of stratifying and potentially constricting Inner Source into a management accounting context, but that **there is potential of adapting management accounting systems to become more flexible in dealing with non-traditional development practices**. Transposing the idea of Gietzmann (1996) to the Inner Source realm, the goal has to be to create a management accounting tool that gives management control over the process without wrestling control from the developers and without obstructing Inner Source collaboration.

In any collaborative project approach, one of the main questions posed by accounting concerns cost centers. More specifically, whether the collaborative development should be considered its own internal entity with costs and potential profits, or if the endeavor should be hosted under the umbrella of one of the participating units (Link, Teece, & Finan, 1996). Further research suggests that companies aim to improve their operations using firm management accounting systems rather than deviating too much from the norm to potentially gain a strategic advantage (Granlund & Lukka, 1998). **Economic and competitive stability seems to take precedence** over potential short-term gains at higher risk.

Research also registers an increasing call for simplicity from industry (Cappelli, 2016; Driscoll, Webb, & Schmidt, 2015). Evidently, it is expected that any tool has sufficient complexity and potential customization to ensure it produces useful results. At the same time, **to guarantee user acceptance, ease of use and simplicity are paramount** and can be achieved through **high levels of abstraction**, and ideally automation of manual processes (Bueno & Salmeron, 2008). In practice, this would mean that key decision makers have access to detailed metrics about the Inner Source software engineering process, while programmers and project managers only have to be involved with the management overhead as little as possible or as necessary. This is in line with the unobstructed working principles of Open and Inner Source, while satisfying the information and steering requirements of management. Making input mechanisms intuitive and standardized can further help both management and employees in their use of the tool.

Recalling the CMM, a management accounting system that allows the advantages of Inner Source to shine while enabling it to reach high levels of maturity, **requires well thought-out and clearly measured metrics, that are updated regularly, or imported directly from project management and repository analysis tools**. Such metrics are found across software engineering management literature and may have to be adapted to suit the broader Inner Source and collaborative approach. Possible metrics are both financial and non-financial in nature to accurately depict the value of any project, not only in the short-term but within the long-term strategic outlook of a company or department (Anderson, 2003; Bibi et al., 2021).

To create more robust predictions when human input for a continuous variable is needed, estimation **techniques like the three-point-approximation** work with three values instead of one value for each input. These values – one pessimistic, one optimistic, and one realistic – are then weighed to achieve a more dependable estimation (Keefer & Bodily, 1983).

Whereas financial values are intrinsic to any accounting system, non-financial metrics are often recorded qualitatively, making analysis more difficult. In many cases, this leads to suboptimal understanding and even disregard of qualitative data – simply because it is more difficult to handle (Dul & Hak, 2007). To solve this and **to make qualitative data usable, management**

accounting systems have to rely on quantification mechanisms, such as the Likert-Scale which was pioneered in psychological research and continues to be used for social science work to this day (Likert, 1932; Joshi, Kale, Chandel, & Pal, 2015).

3 Objective Definition

3.1 Research Question

How can Inner Source collaboration be quantified and controlled in a way that is unobstructive and enables and supports free collaboration in a way that is productive to both the participants and the organization? How can this quantification and control be captured in a design artifact, i.e., a management accounting system?

3.2 Research Approach

This thesis paper follows the Design Science research method laid out by Pfeffers, Tuunanen, Rothenberger, & Chatterjee (2014) which serves as a framework for Information Systems researchers in the successful building of artifacts.

This master's thesis contextualizes the current characteristics and flavors of Open Source as well as their adaptation practices within competitive markets. It also highlights the growing importance of establishing management accounting processes to guide and measure general software development processes and including cost control mechanisms. Following from that, the author motivates the need to introduce this type of comprehensive tooling and practices to quantify and control Inner Source projects and processes (Chapter 1).

In order to explore concrete requirements for the desired artifacts, a two-fold analysis is conducted – two interviews with Inner Source practitioners from industry to identify practical challenges and expectations, as well as a literature review to derive general requirements from a software engineering and information systems management point of view (Chapter 2).

Following this process of establishing requirements, the theoretical solution designs as well as the basic assumptions made prior are laid out. Namely, a conceptual solution is finalized that includes a valuation model for running Inner Source projects or processes, as well as a planning tool to gauge the viability of introducing a component to Inner Source (Chapter 4).

The implementation of this conceptual design is shown as a proof of concept using traditional spreadsheets (Chapter 5), and subsequently demonstrated using an example data set and significant use case scenarios (Chapter 6).

Finally, the thesis then concludes with listing potential shortcomings, limitations, approaches for further research and subsequent artifact iterations (Chapter 7) as well as closing thoughts (Chapter 8).

4 Conceptual Solution Design

4.1 Basic Assumptions and Design Decisions

Any economics model or simulation will not be a perfect representation and prediction of reality, as certain simplifications or assumptions have to be made. To that end, these assumptions are based on the previous requirements established from industry interviews as well as a literature review and try to reflect the current standards in software engineering and management accounting.

One of the main assumptions made within the models is that **all code added to a project is of equal value, measured by Lines of Code** and **entered manually**. Initially, there will be no real distinction by code complexity, although there is potential for later inclusion of this mechanism, powered by the tools mentioned earlier. Still, there is value in differentiating contributions by project phase, the results of which have to be evaluated qualitatively.

For the early prototypes of the model, it also requires full transparency, honesty and diligence by the contributing party, so that all contributions are recorded at the necessary level of detail. For later iterations, there is potential for integration with ERP systems, or analytics tools interfacing directly with version control systems and code repositories.

Access restriction in the exemplary model is represented by naming the sheets with a shorthand for the parties able to access them, as follows:

- PM1 – Project manager of team 1
- D2 – Developer team 2
- C – Controller

The **aim is to limit the apparent complexity of the model to simple and consistent inputs**, coupled with **high level of abstraction** to the individual user. I.e., managers and developers are granted access to separate input panels to enter their data. In order to design a model that can produce clear and comparable outputs, **all inputs are required to be quantitative** in nature. For descriptive financial information this is rather straightforward, whereas predictive financial data will have to be provided using the three-point-estimation technique. Any non-financial user input is required to be quantified within a seven-point Likert-scale.

To further increase the resilience of the projection, the estimations entered by participating parties will be **checked automatically for inconsistencies**. In the case of irregularities, the system is supposed to provide the project controller or project manager with a note, who can run a manual revision and, if necessary, advise the project members.

4.2 Model Selection and Design

It was deemed unfeasible and against the caution for simplicity, to implement all of the established requirements into one design model. Therefore, the decision was made **to create two distinct models**, mirroring the two-fold responsibilities associated with management accounting systems – **both planning and monitoring**. As such, **this design paper aims to prototype both a useable a-priori planning tool as well as an a-posteriori monitoring solution**.

The models are prototyped to be financial in nature but aim to derive some information about the development process and non-financial metrics as well. The tool will provide visualizations for some of these whereas the required qualitative analysis will ultimately be left to the tool user.

Particularly for a planning model, it can be useful to include non-financial metrics to determine the viability of a project beyond its direct monetary value. Whenever applicable, this data will have to be provided (and quantified within a set framework) by the user. As these models are prototypes and subject to change, user feedback will be needed, particularly for the implementation of these non-financial metrics.

4.2.1 Model for Monitoring and Direct Compensation

This model will assume that the decision to Inner Source a component has already been made and that **Inner Source development has either started or already been completed**. Hence, this can be seen as an **a-posteriori controlling device**, when cost associated with the development is accounted for, fulfilling the monitoring function of management accounting described in Chapter 1.2.

For each individual project, this model adapts the glass-box approach outlined by Boehm and Papaccio (1988), which aims to document each low-level cost item and its associated cost type, during which project phase it was incurred, and whether the action added direct or indirect value to the endeavor.

This very simplistic model takes as input the contributions by each team – ideally drawn automatically from the project's version control system. These inputs can be:

- Code additions by LoC and Number of Errors
- Revisions by LoC and Number of Errors resolved

From this data, the model computes the overall contribution by each party to the project and visualizes the data. It also gives the controllers and project managers an indication if an imbalance in contributions has occurred.

The **overall contribution in percentage points for contributor A** is thus calculated as follows, using:

- LoC = Lines of Code
- E = Errors produced
- R = Error Resolutions
- m = Custom Error Multiplier

$$Contribution_A = \frac{\sum LoC_A - (E_A - R_A) \times m}{LoC_{Total}} \times 100\%$$

Using the error multiplier, the participants or the controller can decide what emphasis to assign code quality. Using a higher multiplier rewards contributions without errors and **shifts the net-contribution percentage slightly towards contributors that resolve more issues than they create**. For instance, a multiplier of 10 subtracts ten Lines of Code from a contributor's overall per error introduced and adds the same amount for a resolution. This assures that resolving one's own errors has no negative impact on the total and that reworking faulty code of others is recognized in the overall calculation. Setting the multiplier to zero cancels this rebalancing effect out.

As per this implementation, the model will not be considered its own accounting post or cost center. Rather, **it is aimed to be a net-zero mechanism** to measure and coordinate collaboration

between two or more developing parties and to assure mutual understanding of effort and suggest equal contribution. As collaborators within a company will likely want to avoid actual payment, the attributed value can simply be used as a running tally of collaboration to ensure balanced cooperation between two sides.

To ensure an adequate level of comparability between project reports and to guarantee easier understanding for management decision-making, the form and structure of the reports should remain fixed as follows.

The model includes:

- **Management Overview:** Input masks for the respective project managers including a management overview of the running total and potential deviations – one per contributing organizational unit
- **Development Team Input:** Input sheets to track additions and revisions – one per project
- **Control:** Controlling sheet only available to management accounting – one per project

Outlining the reporting decisions made for the tool, this development follows the previously introduced structure described by Charifzadeh and Taschner (2017):

- **Contents and level of detail:** Recording each contribution made to a singular project – granularity at discretion of the contributor
- **Scope:** Restricted to a single Inner Source development project
- **Form and structure of the reports:**

  1. Consisting of a simple input sheet for each contributor, where single contributions are recorded – input manually during prototyping, with ERP-integration as a possibility long-term
  2. Fixed – to ensure comparability between distinct projects
- **Frequency of reporting:** Continuous – standard view: running weekly report
- **Target audience and authorized viewers:** Granting each contributor only access to their own contributions – analysis sheets made available to decision makers, project managers and possibly controllers

4.2.2 Model for Viability of Inner Sourcing

As a derivative of the compensation model described in the previous section, this model should seek to identify the viability of Inner Sourcing a specific component, before development has started. More concisely, **it should determine whether it is worth combining development efforts with another unit** to work on related components. As such, this can be considered an **a-priori planning device**, for when cost and resource expenditure of development can only be projected.

In essence, any decision whether to Inner Source a component or to develop it yourself, comes down to a **modified Make-or-Buy decision**, as laid out in the earlier chapters. In this context, **"making" as the internal sourcing mechanism** is the equivalent of independent closed development – i.e., traditional software engineering. **"Buying" on the other hand, represents external sourcing** – usually done through a one-time transaction between companies – but in this case through Inner Source collaboration. The price paid for an Inner Source component can be considered the cost of one's own contribution as well as the cost of adoption and integration in

one's own environment. If the company wishes to implement as transfer pricing scheme into its Inner Source process, the departments would also account for licensing fees towards the bigger contributors.

The main difference between traditional Make-or-Buy and this **Make-or-Inner-Source** approach is that **there is no clear distinction between seller and buyer**, as **all parties involved are contributing** and taking advantage of others' contributions at the same time. Therefore, there is **usually no passive buyer role** – as everyone is also a contributor – and solutions are built on demand. Advantageously, this mean that the solution is not only custom made, but every contributor has direct insight and influence over the development efforts. However, unlike in Make-or-Buy, the company cannot outsource externalities such as liabilities, potentially requiring additional spending for quality control and certification.

It is evident that there are multiple factors beyond pure financial reasoning that might lead to the decision for collaboration (or against it). For instance, a business unit might choose to contribute to or license an Inner Source component developed by another unit to comply with scheduling of their developers or because they lack know-how in a certain area, even though it causes higher upfront cost than with own development. As is, many project planning tools only allow for quantification of financial and scheduling aspects, which disproportionately affects other success metrics, such as quality and long-term maintenance effort.

The following is an attempt to **adapt the established make-or-buy template to a template displaying the factors important for Inner Source** considerations. Many of the general decision criteria between a make-or-buy and a make-or-contribute decision don't change much. However, as this model is not **supposed to evaluate** whether a development should be pursued or not – simply **how development should be carried out** – the criteria were adapted and rearranged to reflect this change.

In any case, this template should help decision-makers by **establishing the difference in considerations between own development and Inner Sourcing** – such as the **impact on performance**, measured in delivery time, cost or quality. It also helps evaluate the potential synergy effects regarding **resources** and **strategic value** and provides a framework to gauge the **financial implications** of an Inner Source decision.

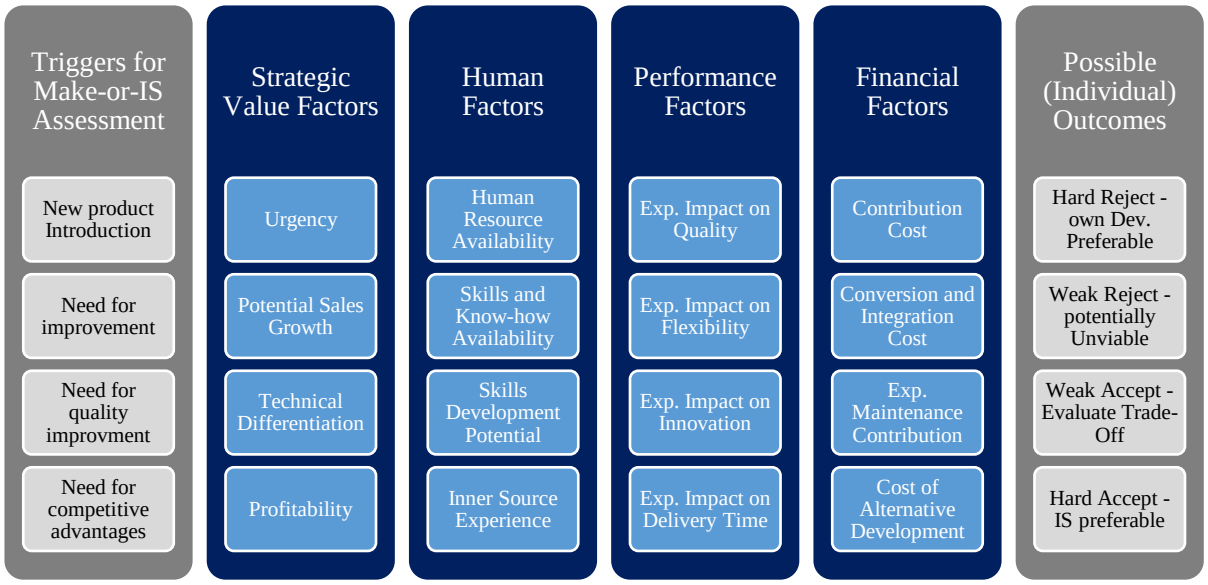


Figure 8: Make-or-Inner-Source Decision Factors
Own depiction modeled after Medina-Serrano et al., 2020

Each of the factors in the first three columns highlighted in light blue is to be rated on a seven-point Likert scale. In this context, “1” represents a low or negative value, “4” an average or neutral value, and “7” a high or positive value.

As established beforehand, the **planning process should enable the participants to prioritize certain decision factors** over others – which is why **managers are not only asked to rate each factor subjectively, but also to assign weights** in accordance with their importance for the specific project.

To the end of specific analysis, the factors can be arranged into a matrix that visualizes the different approach dimensions. These dimensions are:

- **Company Dimension**: Long-term Strategic Value as well as Human Development factors
- **Project Dimension**: Short-term value Performance factors regarding deliverables and Cost impact of Inner Sourcing
- **Quality Dimension**: Concerned with the impact of Inner Source on Product and Process Quality factors
- **Resource Dimension**: Assesses the status quo and the expected impact of Inner Source on company assets

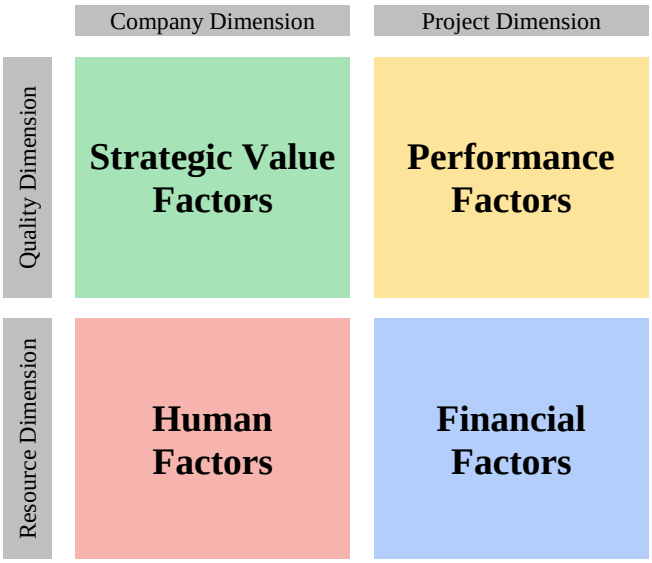


Figure 9: Make-or-Inner-Source Decision Matrix

Using this combined information, **the model returns a judgement of viability of Inner Sourcing**. Importantly, the model carries out **both an overall recommendation of viability** of inner sourcing the project in terms of each of these factors, **as well as a recommendation for each of the participating business units considering their specific delivery time and financial impact situation**. Combined with the fact that this model represents a proof of concept, and as laid out earlier, may need subsequent updates and refinements to properly reflect a company’s needs.

Here, the model takes as inputs all of the potential contributors’ judgments and returns partial assessments for each of the value dimensions. As established, the model also takes some special considerations into account, that have to be assessed individually. High urgency, for instance, does not immediately affect a project’s quality outlook. However, combined with an expected negative impact on delivery time due to Inner Sourcing, the project is at a high risk of going

over schedule. In this case, the model issues an additional warning addressing this shortcoming and leaves the final decision up to the affected user. If, on the other hand, the expected impact on delivery time is a net positive, the time consideration weighs favorably into the overall calculation. Similarly, the model issues a warning if Inner Sourcing is expected to be a net negative on a project's budget, only if the rest of the parameters are favorable.

As for this early model, the **possible outcomes of the assessment** are:

1. **Hard Disapproval** – the currently entered financial and non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after re-defining the project parameters significantly.
2. **Weak Disapproval** – the project as presented is either financially or non-financially unviable for one of the parties. The project could potentially be reworked, and parameters redefined to achieve a more positive result.
3. **Weak Recommendation** – the project is viable from an objective point of view, but not unconditionally. The specific trade-offs should be evaluated by the participants to reduce risk.
4. **Full Recommendation** – the collaboration appears viable on all accounts and can go ahead if all participants agree on the terms.

Analogous to the monitoring tool, this development has the following report characteristics:

- **Contents and level of detail:** High-level approach – effort estimation and assessment of non-financial aspects has to come from each potential contributing party
- **Scope:** Restricted to a single Inner Source development project
- **Form and structure of the reports:** One input sheet for each contributing party consisting of Financial Aspects, Non-financial aspects and a result and recommendation section displaying the analysis of viability regarding the previously laid-out criteria
- **Frequency of reporting:** Singular – potentially to be used for later analysis of deviations
- **Target audience and authorized viewers:** Granting each contributor only access to their own planning tool – meta-analysis can be made available to a controller or kept anonymous

5 Implementation

5.1 Compensation Model

The compensation model **consists of one main input sheet as well as one primary output sheet** used for controlling the Inner Source project. Additionally, the model includes **one combined input and overview sheet for each project manager** of a participating unit.

The different data tables are connected using Excel's PowerQuery function, which, other than avoiding clutter, makes data manipulation more efficient and thus increases the performance of the calculation model.

The project management overview comes pre-filled with data about the project and the participating teams, as well as the previously planned work effort and its distribution between the teams. The **project manager is only asked to add the team members' names**. For the duration of the project, this sheet is updated with currently added information displaying the distribution of work between the teams and potential deviations from the original plan.

Inner-Source Project Management

Business Unit	Team Econ
Project Name:	Testproject Contribution Model
Team Lead:	John Cains
In Collaboration With:	Team CompSci

Team Members
Forbes Nash
Dave Ricardo
Milhouse Friedman
Adam Schmidt
Vilfredo Loreto

Project Data	Planned	Actual	Deviation
Overall Effort (hours)	1600	2000	25,00%
Cost per personnel hour	85,00 €	85,00 €	0,00%
Total Expected Cost	136 000,00 €	170 000,00 €	25,00%
Usage	45%	45%	0,00%
Contribution (in %)	35%	33,38%	-4,63%
Contribution Value	47 600,00 €	56 743,97 €	19,21%
Suggested Compensation	13 600,00 €	19 756,03 €	45,26%

Figure 10: Inner-Source Project Management Overview

The **main information the compensation requires is entered via the developer input sheet**. As mentioned, in a production scenario, this information would have to be gathered automatically, from an ERP or version-control system. However, as per the prototype, the dataset has to be entered manually.

It receives as inputs the time of the commit as a Unix timestamp, the name of the contributor, the type of change as well as which phase of development it falls under. It also takes the size and quality of the commit through Lines of Code and the number of errors and warnings that were introduced or resolved.

Inner-Source Contribution Progress

Business Units: Team Econ & Team CompSci
Project Name: Testproject Contribution Model

Timestamp	Date	Contributor	Type	Phase	LoC	Errors (Negatives denote Resolution)
1650931720	26.04.2022 00:08	Ada Loveley	Addition	Planning	24	2
1650936184	26.04.2022 01:23	Gottfried Bahlsen	Revision	Planning	40	-1
1650957548	26.04.2022 07:19	Charles Cabbage	Addition	Analysis	39	1
1650964521	26.04.2022 09:15	Noam Plomsky	Addition	Analysis	10	0
1650979357	26.04.2022 13:22	Egon Dijkstra	Addition	Analysis	22	0
1650983046	26.04.2022 14:24	Forbes Nash	Revision	Analysis	45	-1
1650984552	26.04.2022 14:49	Dave Ricardo	Revision	Design	48	-1
1650990028	26.04.2022 16:20	Milhouse Friedman	Addition	Design	32	0
1650992831	26.04.2022 17:07	Adam Schmidt	Revision	Analysis	9	0
1651004959	26.04.2022 20:29	Vilfredo Loreto	Addition	Design	51	0

Figure 11: Input Table for Inner Source Contributions

This combined data is connected to PowerQuery for the **main analysis that is carried out in the controlling sheet**, an example of which is displayed below.

Inner-Source Contribution Controlling

Project Name:	Participants:		Error Multiplier:
Testproject Contribution Model	Team Econ	Team CompSci	10

Project Data	Planned	Actual
Overall Effort (hours)	1600	2000
Avg. Personnel cost per hour	85,00 €	85,00 €
Overall Cost	136 000,00 €	170 000,00 €
Usage Team Econ	45%	45%
Usage Team CompSci	55%	55%
Contribution Team Econ	35%	33,38%
Contribution Team CompSci	65%	66,62%

	Team Econ		Team CompSci		
Compensation-Data	Planned	Actual	Planned2	Actual2	Checksum
Usage	45%	45%	55%	55%	OK
Contribution (in %)	35%	33,38%	65%	66,62%	OK
Contribution (value)	47 600,00 €	56 743,97 €	88 400,00 €	113 256,03 €	OK
Compensation	13 600,00 €	19 756,03 €	- 13 600,00 €	- 19 756,03 €	OK

Row Labels	Errors Produced	Errors Resolved	Total LoC Contributed
Team CompSci	94	155	5739
Ada Loveley	48	3	1919
Charles Cabbage	25	7	1379
Egon Dijkstra	14	65	1327
Gottfried Bahlsen	5	52	582
Noam Plomsky	2	28	532
Team Econ	86	25	3791
Adam Schmidt	14	2	564
Dave Ricardo	27	14	1267
Forbes Nash	2	1	93
Milhouse Friedman	20	2	745
Vilfredo Loreto	23	6	1122
Grand Total	180	180	9530

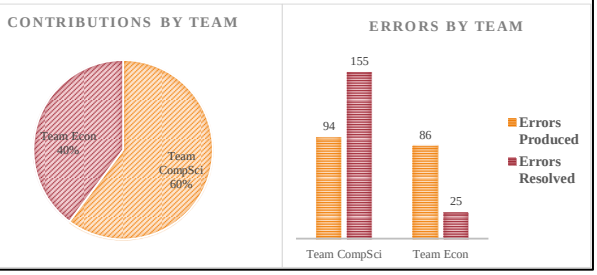


Figure 12: Controlling Overview for the Compensation Model

Here, the controller adds the general project information from prior planning rounds, including estimated hours to completion, an assumed hourly rate for the development team and information about expected usage and contribution (orange table). **The model also accepts subsequent updates on all of these inputs and changes the compensation calculation accordingly**. In addition, the template requires an input for the aforementioned **error multiplier, to rebalance some of the workload applied towards resolving errors** in the codebase.

With this information, the model evaluates both teams' contributions with regards to the originally planned values. Using the formula introduced in Chapter 4.2.1**, the model calculates the relative contribution of all contributors** and – taking into account the expected usage of the Inner Sourced component – **determines whether one team disproportionately benefits** from

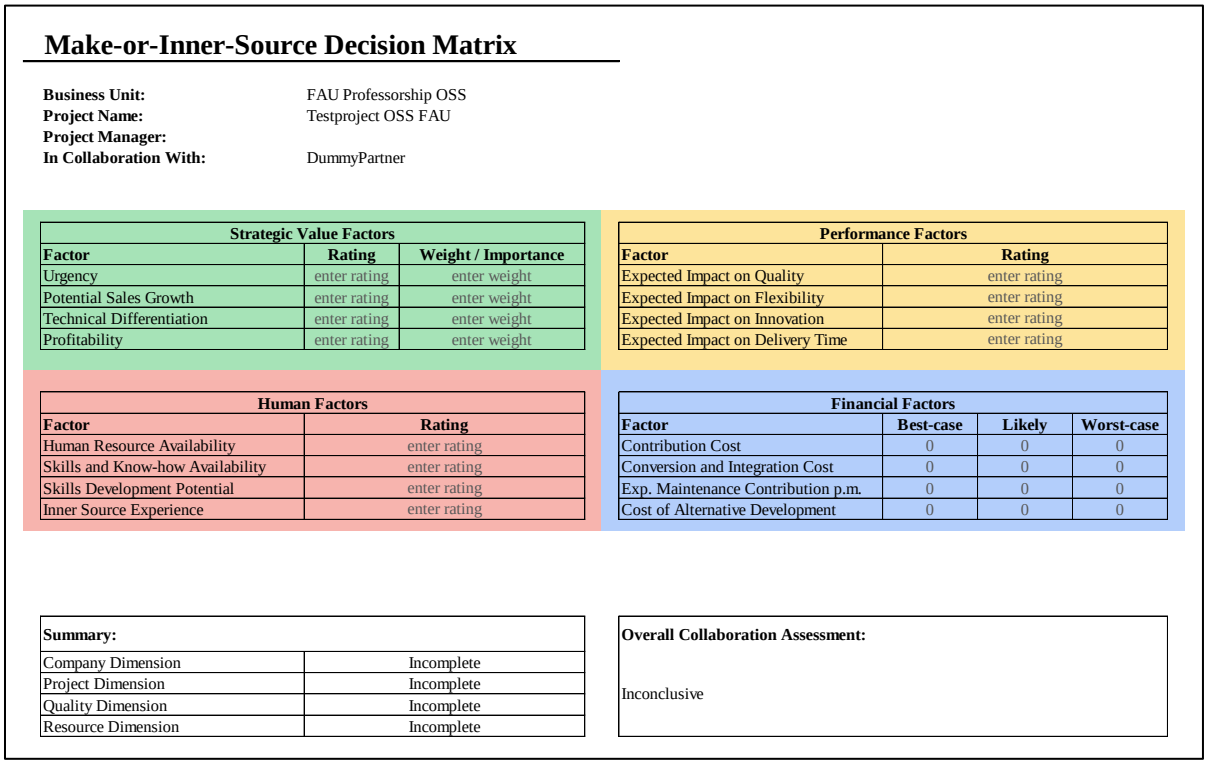
the work of the other(s). The resulting compensation "payment" represents a suggestion of value that is owed.

In addition to the compensation calculation, the dashboard also aggregates information on individual developer contributions, as well as overall team contributions and error distribution. This information can be used to corroborate the model's judgment if questions about the process arise.

5.2 Viability Model

When the **decision to potentially Inner Source a project** is to be evaluated, the controller sets up the evaluation sheets for each of the participating parties. For each potential collaboration, this includes at least three spreadsheets – **one for each of the collaborators** as well as **one overarching controlling and calculation sheet per project**. In addition, each project draws from a sheet containing variables and a look-up table for the wording of the result. This can be shared between projects, but is technically still part of the overarching model.

To make the model accessible and its results easily understandable the implementation of the collaborator sheets **mirrors the previously introduced Make-or-Inner-Source decision matrix**.



Make-or-Inner-Source Decision Matrix

Business Unit: FAU Professorship OSS
Project Name: Testproject OSS FAU
Project Manager:
In Collaboration With: DummyPartner

Strategic Value Factors		
Factor	**Rating**	**Weight / Importance**
Urgency	enter rating	enter weight
Potential Sales Growth	enter rating	enter weight
Technical Differentiation	enter rating	enter weight
Profitability	enter rating	enter weight

Performance Factors	
Factor	**Rating**
Expected Impact on Quality	enter rating
Expected Impact on Flexibility	enter rating
Expected Impact on Innovation	enter rating
Expected Impact on Delivery Time	enter rating

Human Factors	
Factor	**Rating**
Human Resource Availability	enter rating
Skills and Know-how Availability	enter rating
Skills Development Potential	enter rating
Inner Source Experience	enter rating

Financial Factors			
Factor	**Best-case**	**Likely**	**Worst-case**
Contribution Cost	0	0	0
Conversion and Integration Cost	0	0	0
Exp. Maintenance Contribution p.m.	0	0	0
Cost of Alternative Development	0	0	0

Summary:	
Company Dimension	Incomplete
Project Dimension	Incomplete
Quality Dimension	Incomplete
Resource Dimension	Incomplete

Overall Collaboration Assessment:

Inconclusive

Figure 13: Empty input view for a potential Inner Source contributor

In this matrix, the **contributor is asked to estimate each of the decision factors**, as well as weighing the factors by their relative importance, if applicable. Ratings and weights are presented as a drop-down menu with specific descriptions, to avoid confusion and sanitize user input. The Likert-scale allows for easier subsequent data analysis and comparison between contributors.

Strategic Value Factors		
Factor	**Rating**	**Weight / Importance**
Urgency	4	4
Potential Sales Growth	enter ra	
Technical Differentiation	enter ra	
Profitability	enter ra	

1 - not at all important
2
3
4 - fairly important
5
6
7 - very important

Figure 14: Drop-down menus with textual description

To assess the financial viability, users are further **asked to estimate the development cost associated with both Inner Sourcing as well as the alternative cost**, most commonly traditional in-house development. To make this estimate more robust, the model uses the three-point-estimation technique, requiring a realistic, a pessimistic and an optimistic estimate of each of the cost factors.

Financial Factors			
Factor	**Best-case**	**Likely**	**Worst-case**
Contribution Cost	30 000	50 000	80 000
Conversion and Integration Cost	5 000	7 500	20 000
Exp. Maintenance Contribution p.m.	500	700	1 200
Cost of Alternative Development	60 000	80 000	110 000

Figure 15: Financial assessment using three-point-estimation

Using this information, the model yields an assessment of each of the four previously introduced viability dimensions as well as an overall assessment of viability for this collaborator.

Summary:	
Company Dimension	Weak
Project Dimension	Medium
Quality Dimension	Strong
Resource Dimension	Medium

Overall Collaboration Assessment:

Weak Disapproval - The currently entered financial and non-financial information suggests the project may be viable for Inner Sourcing. However, it could lead to an increase in development time.

Figure 16: Exemplary result of viability assessment for single contributor

This assessment is drawn from the central controlling part of the model. Here, the information entered by all participating parties is aggregated in a template, where the actual calculation of viability takes place. This is also where an impartial entity, like a controller or project manager, can revise inputs and, if deemed necessary, re-adjust some of the weights assigned to each of the value factors.

As described above, this calculation yields a judgment for all of the established Inner Source dimensions, as well as giving special consideration to financial and scheduling viability. **Depending on the result of the calculation, the model draws on texts from a look-up-table that contains custom summaries for any possible combination of verdicts**.

For completeness, the actual controlling sheet contains one such template for each assessing partner, that are displayed side by side for better comparison. To the end of better visualization, only one is shown here.

Make-or-InnerSource Controlling Sheet

Project Name
Testproject OSS FAU

Assessing Departments
FAU Professorship OSS
DummyPartner

FAU Professorship OSS

Company Dimension Factors	Rating	Weight	f-weight	Score
Urgency	enter rating	enter weight	0	waiting for input
Potential Sales Growth	enter rating	enter weight	7	waiting for input
Technical Differentiation	enter rating	enter weight	7	waiting for input
Profitability	enter rating	enter weight	5	waiting for input
Human Resource Availability	enter rating	7	3	waiting for input
Skills and Know-how Availability	enter rating	7	3	waiting for input
Skills Development Potential	enter rating	7	7	waiting for input
Inner Source Experience	enter rating	7	2	waiting for input
Long-term Strategic Factors				**0**
Human Development Factors				**0**
Company Position				**Incomplete**

Project Dimension Factors	Rating	Weight	f-weight	Score
Expected Impact on Quality	enter rating	7	5	waiting for input
Expected Impact on Flexibility	enter rating	7	4	waiting for input
Expected Impact on Innovation	enter rating	7	5	waiting for input
Expected Impact on Delivery Time	enter rating	7	3	waiting for input
Contribution Cost				€ -
Conversion Cost				€ -
Exp Maintenance Cost				€ -
IS Cost for one year				€ -
Alternative Cost				€ -
Performance Factors				**0**
Financial Impact				**incomplete**
Project Position				**Incomplete**

Quality Dimension Factors	Rating	Weight	f-weight	Score
Urgency	enter rating	enter weight	-7	waiting for input
Potential Sales Growth	enter rating	enter weight	2	waiting for input
Technical Differentiation	enter rating	enter weight	7	waiting for input
Profitability	enter rating	enter weight	1	waiting for input
Expected Impact on Quality	enter rating	4	7	waiting for input
Expected Impact on Flexibility	enter rating	4	2	waiting for input
Expected Impact on Innovation	enter rating	4	5	waiting for input
Expected Impact on Delivery Time	enter rating	7	7	waiting for input
Time Consideration				**0**
Strategic Factors				**0**
Performance Factors				**0**
Quality Position				**Incomplete**

Resource Dimension Factors	Rating	Weight	f-weight	Score
Human Resource Availability	enter rating	7	7	waiting for input
Skills and Know-how Availability	enter rating	7	7	waiting for input
Skills Development Potential	enter rating	7	4	waiting for input
Inner Source Experience	enter rating	7	6	waiting for input
Contribution Cost				€ -
Conversion Cost				€ -
Exp Maintenance Cost				€ -
IS Cost for one year				€ -
Alternative Cost				€ -
Human Resource Factors				**0**
Financial Impact of IS				**incomplete**
Project Position				**Incomplete**

Overall Metrics	Neutral
Financial Assessment	Neutral
Time Consideration	Neutral

Inconclusive

Figure 17: (Empty) Controlling view of one contributor's inputs

6 Demonstration

6.1 Compensation Model

To demonstrate the functionality of the compensation model under different circumstances, a 300 lines sample data set is used, representing a small development endeavor. Assume a project with the previously mentioned teams (Team Econ and Team CompSci) has the following input parameters. Coincidentally, **usage of and contribution to the Inner Source component are in equal relation**, and there is no deviation from the planned values. If this plan is met during the development process, there will be no need for compensation between the contributors.

Project Data	Planned	Actual
Overall Effort (hours)	1000	1000
Avg. Personnel cost per hour	85,00 €	85,00 €
Overall Cost	85 000,00 €	85 000,00 €
Usage Team Econ	35%	35%
Usage Team CompSci	65%	65%
Contribution Team Econ	35%	
Contribution Team CompSci	65%	

Figure 18: Project Data for a Compensation Model

However, using the sample data set with an error modifier of 0 (zero) **the contribution of Team Econ is calculated to be 39,78%, i.e. in excess of the planned 35%**. To compensate for this, the model suggests a payment of 4062,70€ to Team Econ in exchange for the extra work.

	Team Econ		Team CompSci	
Compensation-Data	**Planned**	**Actual**	**Planned2**	**Actual2**
Usage	35%	35%	65%	65%
Contribution (in %)	35%	39,78%	65%	60,22%
Contribution (value)	29 750,00 €	33 812,70 €	55 250,00 €	51 187,30 €
Compensation	- €	- 4 062,70 €	- €	4 062,70 €

Figure 19: Compensation Calculation for changes in the distribution of contributions

Critically, **this does not consider which party produced and resolved more errors** during the process. Applying an error modifier of 7,5 to the previous example causes the estimated workload to (almost) balance out, leading to a suggested payment of only 17,84€. **If the modifier is raised further to 12, for instance, the balance shifts in favor of Team CompSci**, who are now owed 2466,16€, as they disproportionately worked on errors in the codebase.

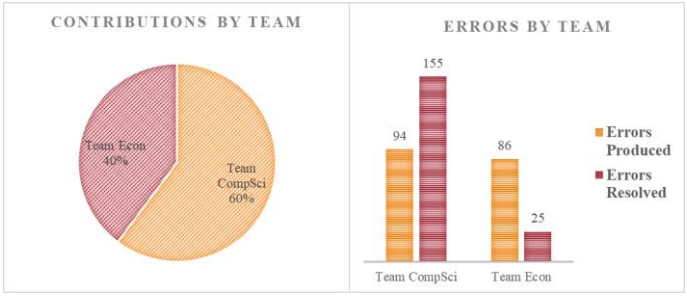


Figure 20: Compensation Model Contribution and Error Statistics

Further, **the model can also compensate for changes in the usage statistics**. Continuing with the relatively close modifier of 7,5, when Team Econ uses the component 55% of times instead of the planned 35%, but their contribution stays the same, they can compensate for this with the equivalent of 17017,84€. This seems logical, as the team is a majority user of the component but has allocated fewer resources to its development than the other contributors.

	Team Econ		Team CompSci	
Compensation-Data	**Planned**	**Actual**	**Planned2**	**Actual2**
Usage	35%	55%	65%	45%
Contribution (in %)	35%	34,98%	65%	65,02%
Contribution (value)	29 750,00 €	29 732,16 €	55 250,00 €	55 267,84 €
Compensation	- €	17 017,84 €	- €	- 17 017,84 €

Figure 21: Compensation Calculation for changes in the distribution of usage

6.2 Viability Model

To make use of the viability model, **the user simply enters their project estimations** into the matrix and **receives a recommendation in terms of non-financial and financial factors**. If applicable, the model also issues a specific warning if Inner Sourcing is likely to introduce a delay incompatible with the project's time frame, as can be seen in the example below.

Strategic Value Factors		
Factor	**Rating**	**Weight / Importance**
Urgency	5	7
Potential Sales Growth	1	5
Technical Differentiation	4	7
Profitability	3	6

Performance Factors	
Factor	**Rating**
Expected Impact on Quality	5
Expected Impact on Flexibility	5
Expected Impact on Innovation	3
Expected Impact on Delivery Time	2

Human Factors	
Factor	**Rating**
Human Resource Availability	3
Skills and Know-how Availability	4
Skills Development Potential	2
Inner Source Experience	2

Financial Factors			
Factor	**Best-case**	**Likely**	**Worst-case**
Contribution Cost	30 000	50 000	80 000
Conversion and Integration Cost	5 000	7 500	20 000
Exp. Maintenance Contribution p.m.	500	700	1 200
Cost of Alternative Development	60 000	80 000	110 000

Summary:	
Company Dimension	Weak
Project Dimension	Medium
Quality Dimension	Strong
Resource Dimension	Medium

Overall Collaboration Assessment:

Weak Disapproval - The currently entered financial and non-financial information suggests the project may be viable for Inner Sourcing. However, it could lead to an increase in development time.

Figure 22: Example Viability Model Input and Result

When this disadvantage is removed from the input matrix, the outcome changes to the following statement.

Overall Collaboration Assessment:

Weak Recommendation - The currently entered financial and non-financial information suggests the project may be viable for Inner Sourcing. This approach is further expected to have a positive impact on development time.

Figure 23: Example Viability Judgment - Weak Recommendation

When the non-financial parameters are overwhelmingly positive and a quicker turnaround time is expected, one of the following statements could be issued, depending on the financial viability of Inner Sourcing.

Overall Collaboration Assessment:

Full Recommendation - The currently entered non-financial information suggests the project would benefit from Inner Sourcing and enable faster development at comparable costs.

Overall Collaboration Assessment:

Full Recommendation - Inner Sourcing this project has the potential to increase quality, improve delivery time and reduce production cost.

Figure 24: Example Viability Judgments - Full Recommendation

When a project is judged non-viable, one of the following negative statements may be displayed.

Overall Collaboration Assessment:

Weak Disapproval - The currently entered non-financial information suggests the project is unviable - a redefinition or in-depth analysis is advised.

Overall Collaboration Assessment:

Hard Disapproval - The currently entered financial and non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after re-defining the project parameters significantly.

Figure 25: Example Viability Judgments - Disapproval

All further potential combinations between the decision factors are listed exhaustively within appendix C.

7 Limitations and Outlook

It is important to note that the goal of this research was to establish how the practices of management accounting could be tailored to the realm of Inner Source software engineering. To this end, this work tries to design a prototype artifact that can serve as the basis for further, more in-depth developments. Evidently, as this research is done **within the scope of a master's thesis, certain limitations had to be introduced**.

As mentioned, any kind extensive management oversight risks hampering the benefits of Inner Sourcing in the first place, which remains to be evaluated by industry practitioners, and could lead to major changes to better reflect real-world application of the artifacts.

As it stands, **the proposed compensation model assumes that all code committed is of equal value** to determine the overall contribution. The introduction of the custom error modifier represents an approach to remedy this and recognize code quality as a factor in value, but it's very rudimentary in its execution. In theory, **partners could still benefit from artificially inflating their LoC numbers** without having contributed to the project at the same scale. Potential methods to increase precision include a weighted approach to LoC metrics by Mas y Parada (2007) or a work time calculation from time between commits as proposed by Buchner and Riehle (2022).

Understandably, suggesting compensation payment of any kind may be seen as controversial. It is important to underline that **collaborative software development is not a zero-sum-game and likening it to one should be done carefully**. In fact, the viability model considers this by comparing traditional development and the potential cost through Inner Sourcing. This criticism of the compensation model is, however, only applicable in cases of one-off collaboration. Long-term collective work could use the suggested compensation as a running total to ensure equal effort on all sides – without ever exchanging money.

As for **the viability model**, the calculation **could be extended with further negative dimensions associated with Inner Sourcing**, such as loss of IP, which is a major concern in some collaborative development situations. Clearly there will also be situations where the model delivers unsatisfactory results, due to the limitation and standardization of inputs. This lack of customization could be resolved in later iterations.

Further, **both of the models do not employ mechanisms that corroborate input against other participants or prior projects**, albeit such functionality should be comparably easy to add.

There is also potential in implementing the discussed Black-box approach as a form to measure the long-term impact and feasibility of certain project parameters and approaches. Both interviewees showed interest in the potential of evaluating their Inner Source projects this way. However, as demonstrated statistically, for this analysis to yield resilient results a significant number of real-life input projects are required.

8 Conclusions

Although the results of this thesis have to contend with several limitations and simplifications, the work nonetheless **demonstrates the possibilities of connecting sound management accounting principles with functional Inner Source processes**. For industry, it presents easily understandable approaches to the inclusion of collaborative software development within the core management functions of planning and monitoring, that are also lightweight enough as to not prohibit free combination of efforts across organizational boundaries. As such, **it may provide a framework to further the adoption** of Open Source principles within companies and **could aid in lowering barriers to entry** into this fascinating and promising realm.

The broad sweep approach that was chosen during the theoretical chapters aimed to not only establish a synergy between management accounting and software engineering, but also to contextualize the challenges and opportunities this connection can provide. The extensive discussion of related research material has also stressed the topic's interdisciplinary position in between the fields of economics, statistics, management as well as computer science. **Located at this intersection between subjects, the topic can draw from a number of potential techniques** to resolve the apparent dichotomy between Open Source principles and management oversight. Combined with the introduction of expert opinions to verify the theoretical findings, this body of work **creates ample opportunity for more focused research to tie into and expand upon**.

Appendix A Interview Expert No. 1

Interview Transcription
Interviewer(s): Julian Hirsch (JH)
Interviewee(s): Expert No. 1

Appendix B Interview Expert No. 2

Interview Transcription
Interviewer(s): Julian Hirsch (JH), Dirk Riehle (DR)
Interviewee(s): Expert No. 2

Appendix C Viability Model Judgments Lookup Table

Non-financial	Financial	Time Consideration	Verdict	Explanation
Negative	Negative	Negative	Hard Disapproval	The currently entered financial and non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after re-defining the project parameters significantly.
Negative	Negative	Neutral	Hard Disapproval	The currently entered financial and non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after re-defining the project parameters significantly.
Negative	Negative	Positive	Weak Disapproval	The currently entered financial and non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after re-defining the project parameters significantly.
Negative	Neutral	Negative	Hard Disapproval	The currently entered financial and non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after re-defining the project parameters significantly.
Negative	Neutral	Neutral	Weak Disapproval	The currently entered non-financial information suggests the project is unviable - a redefinition or in-depth analysis is advised.
Negative	Neutral	Positive	Weak Disapproval	The currently entered non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after re-defining the project parameters significantly.
Negative	Positive	Negative	Weak Disapproval	The currently entered non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. Additionally, it could lead to an increase in development time. However, the lower overall cost could be considered a valuable trade-off.
Negative	Positive	Neutral	Weak Disapproval	The currently entered non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. However, the lower overall cost could be considered a valuable trade-off.
Negative	Positive	Positive	Weak Recommendation	The currently entered non-financial information suggests the project is unviable for Inner Sourcing or the contribution of this department is non-viable. However, lower cost and quicker delivery time could be considered a valuable trade-off.
Neutral	Negative	Negative	Weak Disapproval	The currently entered non-financial information suggests the project may be viable for Inner Sourcing. However, it would have a considerable negative impact on delivery time and production cost.
Neutral	Negative	Neutral	Weak Disapproval	The currently entered non-financial information suggests the project may be viable for Inner Sourcing. However, it would have a considerable negative impact on development cost.
Neutral	Negative	Positive	Weak Recommendation	The currently entered non-financial information suggests the project may be viable for Inner Sourcing. The higher cost can be considered acceptable if development time is prioritized.
Neutral	Neutral	Negative	Weak Disapproval	The currently entered financial and non-financial information suggests the project may be viable for Inner Sourcing. However, it could lead to an increase in development time.

Neutral	Neutral	Neutral	Weak Disapproval	The current analysis parameters are inconclusive and do not yield a clear recommendation for or against Inner Sourcing. A re-balancing of inputs may yield better results.
Neutral	Neutral	Positive	Weak Recommendation	The currently entered financial and non-financial information suggests the project may be viable for Inner Sourcing. This approach is further expected to have a positive impact on development time.
Neutral	Positive	Negative	Weak Disapproval	The currently entered non-financial information suggests the project may be viable for Inner Sourcing. While this is expected to have a positive impact on development cost, it can lead to delays.
Neutral	Positive	Neutral	Weak Recommendation	The currently entered non-financial information suggests the project may be viable for Inner Sourcing at a lower cost than the alternative development.
Neutral	Positive	Positive	Full Recommendation	The currently entered non-financial information suggests the project may be viable for Inner Sourcing. Additionally, it can be expected to have a positive impact on both development time and cost.
Positive	Negative	Negative	Hard Disapproval	The currently entered non-financial information suggests the project may be viable for Inner Sourcing. However, due to expected disadvantages in development cost and time, alternative development is preferable.
Positive	Negative	Neutral	Weak Disapproval	The currently entered non-financial information suggests the project would benefit from Inner Sourcing, but could lead to higher overall costs. Further analysis recommended.
Positive	Negative	Positive	Weak Disapproval	The currently entered non-financial information suggests the project would benefit from Inner Sourcing. However, as it is likely to lead to higher production cost, Inner Sourcing is only recommended if development time is paramount.
Positive	Neutral	Negative	Weak Recommendation	The currently entered non-financial information suggests the project would benefit from Inner Sourcing. However, as it is likely to lead to longer development time, Inner Sourcing is only recommended if development time is not crucial.
Positive	Neutral	Neutral	Full Recommendation	The currently entered non-financial information suggests the project would benefit from Inner Sourcing. Delivery time and production cost are not expected to be affected.
Positive	Neutral	Positive	Full Recommendation	The currently entered non-financial information suggests the project would benefit from Inner Sourcing and enable faster development at comparable costs.
Positive	Positive	Negative	Weak Disapproval	The currently entered financial and non-financial information suggests the project would benefit significantly from Inner Sourcing. However, this option is only viable if an increase in development time is acceptable.
Positive	Positive	Neutral	Full Recommendation	The currently entered financial and non-financial information suggests the project would benefit significantly from Inner Sourcing. Development time is not expected to be affected significantly
Positive	Positive	Positive	Full Recommendation	Inner Sourcing this project has the potential to increase quality, improve delivery time and reduce production cost.

References

Ahlawat, P., Boyne, J., Herz, D., Schmieg, F., & Stephan, M. (2021). Why You Need an Open Source Software Strategy. Retrieved from https://mkt-bcg-com-public-pdfs.s3.amazonaws.com/prod/open-source-software-strategy-benefits.pdf

Anderson, D. J. (2003). *Agile management for software engineering: Applying the theory of constraints for business results*. *The Coad series*. Upper Saddle River, NJ, London: Prentice Hall Professional Technical Reference.

Andreo, S., Calà, A., & Bosch, J. (2021). OpEx Driven Software Architecture a case study. In S. Biffl, E. Navarro, W. Löwe, M. Sirjani, R. Mirandola, & D. Weyns (Eds.), *ECSA: European Conference on Software Architecture: Vol. 12857. Software Architecture: 15th European Conference, ECSA 2021, Virtual Event, Sweden, September 13-17, 2021, Proceedings*. Cham: Springer International Publishing.

Anthony, R. N., & Govindarajan, V. (2014). *Management control systems* (First European Edition). New York: McGraw-Hill Education.

Arndt, T. (2018). Big Data and software engineering: prospects for mutual enrichment. *Iran Journal of Computer Science*, *1*(1), 3–10. https://doi.org/10.1007/s42044-017-0003-0

Aurum, A., & Wohlin, C. (Eds.) (2005). *SpringerLink Bücher*. *Engineering and Managing Software Requirements*. Berlin/Heidelberg: Springer-Verlag. Retrieved from http://swbplus.bsz-bw.de/bsz120991276err.htm https://doi.org/10.1007/3-540-28244-0

BaniMustafa, A. (2018). Predicting Software Effort Estimation Using Machine Learning Techniques. In *2018 8th International Conference on Computer Science and Information Technology (CSIT)* (pp. 249–256). IEEE. https://doi.org/10.1109/CSIT.2018.8486222

Bibi, N., Anwar, Z., & Rana, T. (2021). Expertise based skills management system to support resource allocation. *PloS One*, *16*(8), e0255928. https://doi.org/10.1371/journal.pone.0255928

Bloch, M., Blumberg, S., & Laartz, J. (2012). Delivering large-scale IT projects on time, on budget, and on value. Retrieved from https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value

Boehm, B. W., & Papaccio, P. N. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, *14*(10), 1462–1477. https://doi.org/10.1109/32.6191

Bosio, D., Bev Littlewood, L. Strigini, & M. J. Newby (2002). Bosio, D., Littlewood, B., Strigini, L., & Newby, M. J. (2002, February). Advantages of open source processes for reliability: clarifying the issues. In C. Gacek & B. Arief (Chairs), *Proceedings of the Open Source Software Development Workshop,* Newcastle upon Tyne, UK.

Buchner, S., & Riehle, D. (2022). Calculating the Costs of Inner Source Collaboration by Computing the Time Worked. In T. Bui (Ed.), *Proceedings of the Annual Hawaii International Conference on System Sciences, Proceedings of the 55th Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences. https://doi.org/10.24251/HICSS.2022.896

Budzier, A., & Flyvbjerg, B. (2012). Overspend? Late? Failure? What the Data Say About IT Project Risk in the Public Sector. In Commonwealth Secretariat (Ed.), *Commonwealth Governance Handbook 2012/13: Democracy, development and public administration* (pp. 145–157). London: Commonwealth Secretariat. Retrieved from https://arxiv.org/pdf/1304.4525

Bueno, S., & Salmeron, J. L. (2008). TAM-based success modeling in ERP. *Interacting with Computers*, *20*(6), 515–523. https://doi.org/10.1016/j.intcom.2008.08.003

Cánez, L. E., Platts, K. W., & Probert, D. R. (2000). Developing a framework for make-or-buy decisions. *International Journal of Operations & Production Management*, *20*(11), 1313–1330. https://doi.org/10.1108/01443570010348271

Cappelli, M. (2016). Overcoming the challenges of a complex ERP environment. Retrieved from https://searcherp.techtarget.com/tip/Overcoming-the-challenges-of-a-complex-ERP-environment

Capraro, M., & Riehle, D. (2017). Inner Source Definition, Benefits, and Challenges. *ACM Computing Surveys*, *49*(4), 1–36. https://doi.org/10.1145/2856821

Charifzadeh, M., & Taschner, A. (2017). *Management accounting and control: Tools and concepts in a central European context / Michel Charifzadeh and Andreas Taschner*. Weinheim, Germany: Wiley-VCH.

DeCotiis, T. A., & Dyer, L. (2016). Defining and Measuring Project Performance. *Research Management*,

22(1), 17–22. https://doi.org/10.1080/00345334.1979.11756516

Deephouse, C., Mukhopadhyay, T., Goldenson, D. R., & Kellner, M. I. (1995). Software Processes and Project Performance. *Journal of Management Information Systems*, *12*(3), 187–205. https://doi.org/10.1080/07421222.1995.11518097

Deshpande, A., & Riehle, D. (2008). The Total Growth of Open Source. In B. Russo, E. Damiani, S. Hissam, B. Lundell, & G. Succi (Eds.), *IFIP – The International Federation for Information Processing. Open Source Development, Communities and Quality* (Vol. 275, pp. 197–209). Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-09684-1_16

Dingsoyr, T., Dyba, T., Gjertsen, M., Jacobsen, A. O., Mathisen, T.-E., Nordfjord, J. O., . . . Strand, K. (2019). Key Lessons From Tailoring Agile Methods for Large-Scale Software Development. *IT Professional*, *21*(1), 34–41. https://doi.org/10.1109/MITP.2018.2876984

Driscoll, M., Webb, H., & Schmidt, J. (2015). ERP Complexity vs. business growth: At odds or in alignment depends on your approach. Retrieved from https://www.cognizant.com/whitepapers/2015-research-on-the-ERP-landscape_Publisher.pdf

Dueñas, S., Cosentino, V., Gonzalez-Barahona, J. M., Del Castillo San Felix, A., Izquierdo-Cortazar, D., Cañas-Díaz, L., & Pérez García-Plaza, A. (2021). Grimoirelab: A toolset for software development analytics. *PeerJ. Computer Science*, *7*, e601. https://doi.org/10.7717/peerj-cs.601

Dul, J., & Hak, T. (2007). To quantify or to qualify: That's not the question. *Journal of Purchasing and Supply Management*, *13*(3), 207–209. https://doi.org/10.1016/j.pursup.2007.09.010

Filieri, A., Maggio, M., Angelopoulos, K., D'Ippolito, N., Gerostathopoulos, I., Hempel, A. B., . . . Vogel, T. (2015). Software Engineering Meets Control Theory. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 71–82). IEEE. https://doi.org/10.1109/SEAMS.2015.12

Gietzmann, M. B. (1996). Incomplete contracts and the make or buy decision: Governance design and attainable flexibility. *Accounting, Organizations and Society*, *21*(6), 611–626. https://doi.org/10.1016/0361-3682(96)00002-5

Gillies, S. (2016, February 11). Open source demonstrates the future of work: Traditional work paradigms are collapsing. Open source models offer a more humane future of and for "work.". Retrieved from https://opensource.com/open-organization/16/2/open-source-demonstrates-future-work

Granlund, M., & Lukka, K. (1998). It is a Small World of Management Accounting Practices. *Journal of Management Accounting Research*, 153–179.

Hihn, J., & Menzies, T. (2015). Data Mining Methods and Cost Estimation Models: Why is it So Hard to Infuse New Ideas? In *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)* (pp. 5–9). IEEE. https://doi.org/10.1109/ASEW.2015.27

Johansson, M., Hallberg, N., Hinterhuber, A., Zbaracki, M., & Liozu, S. (2012). Pricing strategies and pricing capabilities. *Journal of Revenue and Pricing Management*, *11*(1), 4–11. https://doi.org/10.1057/rpm.2011.42

Joshi, A., Kale, S., Chandel, S., & Pal, D. (2015). Likert Scale: Explored and Explained. *British Journal of Applied Science & Technology*, *7*(4), 396–403. https://doi.org/10.9734/BJAST/2015/14975

Jung, W., & Han, S. H. (2017). Which Risk Management Is Most Crucial for Controlling Project Cost? *Journal of Management in Engineering*, *33*(5), 4017029. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000547

Kaner, C., & Bond, W. P. (2004). Software Engineering Metrics: What Do They Measure and How Do We Know? *In METRICS 2004. IEEE CS.*

Keefer, D. L., & Bodily, S. E. (1983). Three-Point Approximations for Continuous Random Variables. *Management Science*, *29*(5), 595–609. Retrieved from http://www.jstor.org/stable/2631360

Kuster, J., Huber, E., Lippmann, R., Schmid, A., Schneider, E., Witschi, U., & Wüst, R. (2015). Project Controlling. In J. Kuster, E. Huber, R. Lippmann, A. Schmid, E. Schneider, U. Witschi, & R. Wüst (Eds.), *Management for Professionals. Project Management Handbook* (pp. 165–186). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-45373-5_17

Lancioni, R. (2005). Pricing issues in industrial marketing. *Industrial Marketing Management*, *34*(2), 111–114. https://doi.org/10.1016/j.indmarman.2004.07.009

Larson, E. W., & Gobeli, D. H. (1989). Significance of project management structure on development success. *IEEE Transactions on Engineering Management*, *36*(2), 119–125. https://doi.org/10.1109/17.18828

Lawrie, T., & Gacek, C. (2002). Issues of dependability in open source software development. *ACM SIGSOFT Software Engineering Notes*, *27*(3), 34–37. https://doi.org/10.1145/638574.638584

Likert, R. (1932). A Technique for the Measurement of Attitudes. *Archives of Psychology*. (140), 1–55.

Link, A. N., Teece, D. J., & Finan, W. F. (1996). Estimating the benefits from collaboration: The case of SEMATECH. *Review of Industrial Organization*, *11*(5), 737–751. https://doi.org/10.1007/BF00214832

Mas y Parareda, B., & Pizka, M. (2007). Measuring Productivity Using the Infamous Lines of Code Metric. In J. Keung (Chair), *Proceedings of The First International Workshop on Software Productivity Analysis and Cost Estimation.* Symposium conducted at the meeting of Information Processing Society of Japan; Special Interest Group on Software Engineering, Nagoya, Japan. Retrieved from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.4666&rep=rep1&type=pdf#page=13

Medina-Serrano, R., González-Ramírez, R., Gasco-Gasco, J., & Llopis-Taverner, J. (2020). Strategic sourcing: Developing a progressive framework for make-or-buy decisions. *Journal of Industrial Engineering and Management*, *13*(1), 133. https://doi.org/10.3926/jiem.2858

Morgan, L., Gleasure, R., Baiyere, A., & Dang, H. P. (2021). Share and Share Alike: How Inner Source Can Help Create New Digital Platforms. *California Management Review*, *64*(1), 90–112. https://doi.org/10.1177/00081256211044830

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering. In A. Finkelstein (Ed.), *Proceedings of the conference on The future of Software engineering - ICSE '00* (pp. 35–46). New York, New York, USA: ACM Press. https://doi.org/10.1145/336512.336523

Open Source Initiative (2007, March 22). The Open Source Definition. Retrieved from https://opensource.org/osd

Oun, T. A., Blackburn, T. D., Olson, B. A., & Blessner, P. (2016). An Enterprise-Wide Knowledge Management Approach to Project Management. *Engineering Management Journal*, *28*(3), 179–192. https://doi.org/10.1080/10429247.2016.1203715

Paulk, M. C. (2009). A History of the Capability Maturity Model for Software. *The Software Quality Profile*, *1*(1).

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). Capability maturity model, version 1.1. *IEEE Software*, *10*(4), 18–27. https://doi.org/10.1109/52.219617

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2014). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/mis0742-1222240302

Perens, B. (1998). The Open Source Definition. Retrieved from https://www.researchgate.net/profile/Bruce-Perens/publication/200027107_Perens_Open_Source_Definition_LG_26/links/568bdeb408ae16c414a9c549/Perens-Open-Source-Definition-LG-26.pdf

Pollack, J., Helm, J., & Adler, D. (2018). What is the Iron Triangle, and how has it changed? *International Journal of Managing Projects in Business*, *11*(2), 527–547. https://doi.org/10.1108/IJMPB-09-2017-0107

Raith, F., Richter, I., & Lindermeier, R. (2017). How Project-management-tools are used in Agile Practice. In B. C. Desai, J. Hong, & R. McClatchey (Eds.), *Proceedings of the 21st International Database Engineering & Applications Symposium on - IDEAS 2017* (pp. 30–39). New York, New York, USA: ACM Press. https://doi.org/10.1145/3105831.3105865

Reynolds, C. J., & Wyatt, J. C. (2011). Open source, open standards, and health care information systems. *Journal of Medical Internet Research*, *13*(1), e24. https://doi.org/10.2196/jmir.1521

Riehle, D. (2009). The Commercial Open Source Business Model. In M. L. Nelson, M. J. Shaw, & T. J. Strader (Eds.), *Lecture notes in business information processing, 1865-1348: Vol. 36. Value creation in e-business management: 15th Americas Conference on Information Systems, AMCIS 2009, Sigebiz Track, San Francisco, CA, USA, August 6-9, 2009, selected papers / [edited by] Matthew L. Nelson, Michael J. Shaw, Troy J. Strader* (1st ed.). New York: Springer.

Riehle, D. (2011). Controlling and Steering Open Source Projects. *Computer*, *44*(7), 93–96. https://doi.org/10.1109/MC.2011.206

Riehle, D., Capraro, M., Kips, D., & Horn, L. (2016). Inner Source in Platform-Based Product Engineering. *IEEE Transactions on Software Engineering*, *42*(12), 1162–1177. https://doi.org/10.1109/TSE.2016.2554553

Riehle, D., 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. https://doi.org/10.1109/MS.2009.44

Rook, P. (1986). Controlling software projects. *Software Engineering Journal*, *1*(1), 7. https://doi.org/10.1049/sej.1986.0003

Stol, K.-J., Avgeriou, P., Babar, M. A., Lucas, Y., & Fitzgerald, B. (2014). Key factors for adopting inner source. *ACM Transactions on Software Engineering and Methodology*, *23*(2), 1–35. https://doi.org/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. https://doi.org/10.1016/j.infsof.2011.06.007

Volpi, M. (2019). How open-source software took over the world. Retrieved from https://techcrunch.com/2019/01/12/how-open-source-software-took-over-the-world/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAALOQM0-mVOkO3itQzp9N1gGN2xpBF4tyfr-4W4VFrJJtDzzCtdF4RPqH0HuT9kEkSgz1Qb-YMkV-NpYCGc2wZZ992Ica_c6-AhSZBWBsqi6ugrP_MdcM7twIjwGh6al7RMHbpIQh-Nggo5qz5DQTDZmH9JxS8fPyM5jlGgflKnNj

Expert 1 (2021, April 1). Interview by J. Hirsch.

Watson, R. T., Boudreau, M.-C., York, P. T., Greiner, M. E., & Wynn, D. (2008). The business of open source. *Communications of the ACM*, *51*(4), 41–46. https://doi.org/10.1145/1330311.1330321

Expert 2. (2021, April 7). Interview by J. Hirsch, & D. Riehle.