Management Accounting Concepts for Inner Source Software Engineering

Julian Hirsch(✉) and Dirk Riehle

Computer Science Department, Friedrich-Alexander University Erlangen-Nürnberg, Martensstraße 3, 91058 Erlangen, Germany
julian.hirsch@fau.de, dirk@riehle.org
https://oss.cs.fau.de/

Abstract. Inner source software development is the use of open source development's best practices inside a company. In inner source, developers collaborate on reusable software components across company-internal organizational silo boundaries for mutual benefit. As such, inner source goes against the grain of traditional management techniques. In this article, we present two conceptual models of management accounting for inner source. We derived these prototypes by performing a literature review and triangulating the results with interviews of industry practitioners. We demonstrate how the conceptual models can be used for monitoring and controlling inner source projects and to determine their future viability.

Keywords: Software engineering · Inner source · Management accounting · Business intelligence

1 Introduction

The collective and non-hierarchical idea of open source has helped propel the software engineering paradigm from a niche phenomenon to a dominant force across markets, with double or triple digit growth rates to underline its increased prevalence [1–3].

Peer-review, unobstructed communication and transparency of development allow for frequent releases while mitigating flaws and revisions. In fact, it has 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 [4,5]. Further, the egalitarian and meritocratic nature of work, championed by user rights in open source licenses, makes it an attractive outlet for developers to collaborate on projects without direct financial incentive [6].

Unsurprisingly, with these advantages to open source software development, private companies are interested in branching into the space. Their goals, however, appear to be diametrically opposed, as a company's goal is to extract economic value out of their development and protect intellectual property [7].

N. Carroll et al. (Eds.): ICSOB 2022, LNBIP 463, pp. 101–116, 2022.
https://doi.org/10.1007/978-3-031-20706-8_7

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 it derivative works [8]. This mismatch between open source working principles, proprietary licensing and community-driven development will probably remain unresolvable, forcing companies to choose two of the three options, while opting to mitigate losing the third. One approach (of several) to capture some of the benefits of open source is the limitation of open source practices and principles to a closed system with organizations - appropriately named inner source [9]. With these rather disruptive changes to company processes, a number of (potential) problems when moving towards inner source have been identified by the likes of Riehle et al. [10] and Stol et al. [6,11], respectively. To combat these - and to ensure adoption within more traditional management structures - steering and controlling subsystems are most likely required to make inner source a useful integration into most companies.

Within a design science framework, we present potential approaches to integrate inner source software engineering with management accounting. To that end, in Sect. 2 we present related work regarding inner source, accounting, and project controlling, followed by a description of our research method in Sect. 3, which leads into a discussion of results in Sect. 4. Subsequently, we dedicate Sects. 5 and 6 to limitations and concluding thoughts, respectively.

2 Related Work

Prior research in this exact area is limited, with one adjacent topic being presented by Buchner and Riehle [12], which approaches the topic of financially evaluating inner source collaboration from a financial accounting perspective, employing transfer pricing principles. Capraro et al. [13] further developed a framework for measuring collaboration across organizational boundaries. The area of inner source management accounting, however, lacks an established approach entirely, possibly due to the legal ambiguity on the topic of management accounting in general. Since we seek to establish a connection between inner source and management accounting, related work covers both of these topics as well as the adjacent area of software project controlling, which deals with many of the same areas of interest.

2.1 Inner Source

Inner source is the use of open source best practices to within an enterprise - thus eliminating the need for full public availability and permissive licensing [10]. At first glance, this limitation to within the scope of an enterprise seems to be a stark contrast to the ideas of open collaboration. 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. This is achieved by sacrificing 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 can expect to improve compliance with schedules, achieve higher reliability and more efficient development practices due to less parallel development, and to generally further develop and improve their software product platforms [10,14].

2.2 Management Accounting

Management accounting represents the internal reporting and controlling tool, accumulating business information used to supplement management decision making. Unlike financial accounting, it is not bound by country-specific legislation and not limited to purely financial information, and as such, it is freely extensible to cater to any company-critical needs. It often includes projections and forecasts on critical KPIs and business metrics [15].

It is also of interest during strategic planning and sourcing. Combining both non-financial and financial aspects, it aids the decision between designing and developing products in-house or buying them for a third party - also known as a make-or-buy decision. More modern approaches, enabled through in-depth process insights, have extended this to create hybrid solutions like "make-and-buy", opening the door to partnerships and collaboration [17,18].

Thus, a management accounting system provides the necessary information both for analyzing the raw data to measure performance of current business operations as well as to serve as the basis for further planning. As such, its task are interwoven with all levels of operations and carried out across several organizational units. The main distinguishing factor being occasion - with some tasks taking place in the planning phase before a project is started and others used to analyze and steer currently running projects.

Summing up, the main tasks of a management accounting and controlling system, grouped by when in the project life cycle they take place, are:

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

2.3 Software Project Controlling

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. For this reason, organizations engaged in software development are always looking for ways to improve their processes and iteratively improve upon their development projects [19–22].

In essence, every project has to be controlled in terms of its cost as well as its processes, including scheduling as well as project outcome and process quality which holds true in software development until today [23–25]. Approaching cost

controlling from a bottom-up perspective, the cost-distribution (or glass-box) approach, originally described by Boehm and Papaccio [24] becomes useful. This aggregative costing method relies on detailed documentation of all cost factors, as well as when they were incurred, which is usually already found in modern day project management, ERP, or management accounting systems. Specifically, the glass-box emphasizes the distribution of these types of cost:

1. **Value-added or corrective activity** - i.e., time spent on development, implementation and improvement vs. documentation and maintenance effort
2. **Capital or labor cost** - Capital Expenditure vs. Operating Expenditure
3. **Project phase** - planning, implementation, testing, maintenance, etc.

In software engineering, it can be difficult to gauge efficiency and technical performance, as such aspects are often too complex to be measured directly [26]. Simple Lines of Code (LoC) metrics are used when comparing the size of programs and code 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 [27] have adapted this LoC approach, proposing an extension to "Redundancy-free LoC per Effort" as well as an inclusion of "Defects per LoC". This combination of metrics was found to be a fair and comprehensive measure of productivity in software engineering.

3 Research Method

This paper uses the design science research method laid out by Peffers et al. [28] which serves as a framework for information systems researchers in the development of artifacts. This approach was chosen due to its ability to convey new developments in an easily-understandable way, building a proof-of-concept solution artifact in the process. Consequently, our work follows these steps:

1. **Problem Identification** - Accurately defining the problem and showing the topic's importance. Combining literature analysis with industry interviews, we established a dependable problem definition and align it with requirements from practice.
2. **Objective Definition** - Identifying what solution a design artifact should and could provide. Using the requirements established with industry partners, we determine our concept design should implement the two main task groups traditionally handled by management accounting and apply them to the inner source context.
3. **Solution Design** - Developing an innovative design science artifact in accordance with the specified requirements. Our solution aims to provide analysis and decision making aid with minimal intrusion into the software process.
4. **Implementation** - Building the design model according to specification and documenting the process. We implement the approach using a basic spreadsheet system, that makes the process traceable and the model easy to be understood and modified.

5. **Demonstration** - Presenting the design artifact within its intended context. For this, we simulate a planning and evaluation process with retroactive analysis of software projects.

4 Research Results

4.1 Problem Identification

The problem identification leaned heavily on literature research to identify typical challenges associated with the introduction of inner source as well as the inclusion of management accounting processes to guide these new introductions. In addition, interviews conducted with industry partners complemented and corroborated the literature findings to make them more dependable and move closer towards real-world industry requirements.

Literature Review. The rights bestowed on any contributor and user in open source are technically at odds with the goal of quantifying contributions to determine appropriate compensation, usually monetary. As one of the key principles of open source economics, no royalties or license fees are collected.

Additionally, inner source does not follow the clear division between companies and organizational units that most management systems are set up with, complicating the creation of cooperative environments. Accounting subsystems are seen as stable and non-adaptive, forcing processes to change around them, clashing with the flexible approach displayed in inner source. Research suggests that companies are hesitant to change management accounting to accommodate changes, rather using them to promote economic and competitive stability [29].

Another point of contention are sources, quality and consistency of input data. Typically, software engineering professionals prefer undisturbed work environments, with as little management overhead as possible. Adding additional requirements for recording contributions could risk losing support from practitioners and might be harmful to efficiency and operations long-term. To guarantee user acceptance, ease of use and simplicity are paramount, and high levels of abstraction are expected [30–32]. To align this with the unobstructed working principles of open and inner source, while satisfying the information and steering requirements of management, programmers and technical leads can be at most tangentially involved - whereas managers and key decision makers get access to detailed and accurate metrics about the inner source process.

Further, many project monitoring and planning systems only record financial data qualitatively, while recording non-financial metrics qualitatively (if at all), making meaningful analysis a challenge. In many cases, this leads to sub-optimal understanding and even disregard of qualitative data - simply because it is more difficult to handle [33].

Industry Interviews. We conducted structured interviews with two industry experts in software development management positions at a multi-industry company. These partners have experience in the management of collaborative development efforts and have been involved with and shaped the company's inner

source processes. As such, they are considered key stakeholders and provide valuable insights into industry practices.

Many of the points raised above were mirrored during the industry interviews, with practitioners lamenting inconsistency in the granularity and quality of data used to run inner source analyses, mainly due to increased overhead caused by catering to yet another system. Crucially, implementing an inner source process does not take shape as a harmonization between organizational units, but rather adds an additional layer of complexity to the existing structure. This leads to a rather loose organization and low-priority designation of all inner source processes.

To improve upon this, the industry experts emphasize the need for consistent metrics that allow for objective measurements to achieve any meaningful valuation of work and to acquire dependable data for subsequent planning rounds. Importantly, pertaining to code contribution, they also mention that in their experience, a single code commit rarely has value of its own. Rather, it is tied to its correctness (verified through tests) and its context, i.e. the value of the respective user story to the overall component. The interviewees also corroborate the usefulness of code classification as either corrective action, commodity functionality, or the creation of highly innovative features. Any system should attempt to capture this distinction and allow for easy valuation and comparison between different types of contribution.

Pertaining to the specific application of inner source within biomedical engineering, legal compliance was mentioned as one further hurdle in implementing inner sourced processes. Within the interviewees' field of work, processes require significant oversight to assure legal compliance, which can be difficult to combine with collaborative development, as it can be harder to trace.

Finally, the interviewees expressed concern with valuating inner source contribution with the goal of comparing efforts among participants. They fear a tracking system risks unsettling contributors and might decrease continued participation in inner source efforts.

4.2 Objective Definition

The research process described in the previous section provides insights into the perceived incongruity between management's need for information and the inherent principles of open collaboration ascribed to inner source. Further, it highlights the challenges practitioners are faced with when implementing a seemingly non-conforming new process. Consequently, this paper's research goal is defined as: The establishment of management accounting tools

- that can control and quantify inner source that is unobstructive to the collaboration process itself
- with a high degree of flexibility to apply to a wide range of pre-existing processes and management systems

with the goal of providing support to decision makers within software engineering companies.

4.3 Conceptual Solution Design

It was deemed unfeasible and against the caution for simplicity to implement all of the established requirements into one design model. Therefore, we made the decision 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.

Compensation Model. The first artifact - the model for monitoring and direct compensation - assumes 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. It applies a glass-box approach to analytics, and aims to document each low-level cost item, as well as its associated cost type, project phase and whether the action provided direct or indirect value to the endeavor. As such, the 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 as well as 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. Using:

$$LoC = Lines\ of\ Code \tag{1}$$

$$E = Errors\ produced\ within\ Code \tag{2}$$

$$R = Error\ resolutions\ within\ Code \tag{3}$$

$$\phi = Custom\ error\ modifier \tag{4}$$

the overall contribution in percentage points for contributor A is thus calculated as follows:

$$Contribution_A = \frac{\Sigma LoC - (E_A - R_A) \times \phi}{LoC_{Total}} \tag{5}$$

Using the error modifier phi, the participants or the controller can decide what emphasis to assign code quality. Using a higher value rewards contributions without errors and shifts the net-contribution percentage towards contributors that resolve more issues than they create. For instance, a modifier 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 value of phi to zero cancels this rebalancing-effect out. Building on the contribution figure, adding:

$$W = Total\ Work\ hours\ (expected\ or\ actual) \tag{6}$$

$$C = Cost\ per\ personnel\ hour \tag{7}$$

$$U = Share\ of\ Usage\ (expected\ or\ actual) \tag{8}$$

the deviating contribution, and thus, suggested compensation, is calculated using this formula:

$$Compensation_{A\ to\ B} = (W \times C) \times (U - Contribution_A) \tag{9}$$

As per this implementation, the model is aimed to be a net-zero mechanism. As such, its goal is to measure and coordinate collaboration between two or more developing parties and to assure mutual understanding of effort and suggest equal contribution. When collaborators within a company want to avoid actual payment, the attributed value can simply be used as a running tally of collaboration to ensure equal cooperation between partners long term.

Viability Model. The second artifact seeks 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.

This artifact leans on the planning and sourcing angle of management accounting - mirroring the idea of a make-or-buy decision. 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 of an inner source component can be considered the cost of one's contribution as well as the cost of adoption and integration in one's own environment.

The main difference between 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. Advantageously, this mean that the solution is not only custom made, but every contributor can directly influence development. 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 or because they lack know-how, even though it causes higher upfront cost than with own development.

The following is an attempt to transform the established make-or-buy template by Medina-Serrano et al. [18] to a template displaying factors for inner source considerations. Importantly, since this model is not supposed to evaluate whether development should be pursued or not - simply how it should be carried out - the criteria were adapted and rearranged to reflect this change.

This template can help decision-makers by establishing the difference in considerations between own development and inner sourcing - such as the impact

Table 1. Make-or-inner-source decision factors

Triggers for make-or-IS assessment	Strategic value factors	Human factors	Performance factors	Financial factors	Possible outcomes
New product introduction	Urgency	Human resource availability	Exp. impact on quality	Contribution cost	Hard rejection
Need for improvement	Potential sales growth	Skills and know-how availability	Exp. impact on flexibility	Conversion and integration cost	Weak rejection
Need for higher quality	Technical differentation	Potential skills development	Exp. impact on innovation	Exp. maintenance contribution	Weak recommendation
Need for competitive advantage	Profitability	Inner source experience	Exp. impact on delivery time	Cost of alternative development	Full recommendation

on performance, measured in delivery time, cost or quality. It also helps evaluate potential synergy effects regarding resources and strategic value and provides an analysis of the financial implications of an inner source decision. To the end of specific analysis, we arranged the decision factors into a matrix that visualizes different approach dimensions.

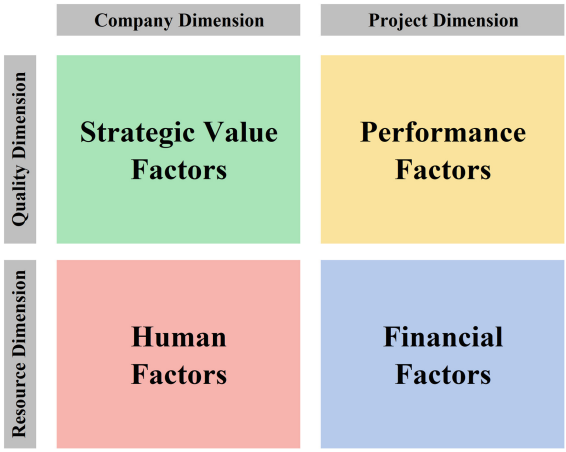


Fig. 1. Make-or-inner-source decision matrix

With 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 dimensions, as well as a recommendation for each of the participating business units considering their delivery time and financial impact.

4.4 Implementation

Both design artifacts were implemented as spreadsheets solutions, to allow for quick iteration and adaptation, while using Excel's PowerQuery to guarantee model simplicity and performance.

Compensation Model. The model for direct compensation takes the following manual inputs, both as planned prior to execution and as incurred during the project. They should be provided through a project controller or overarching project manager, to ensure objectivity.

- Overall project effort (in hours)
- Cost per personnel hour
- Usage percentage of the final artifact

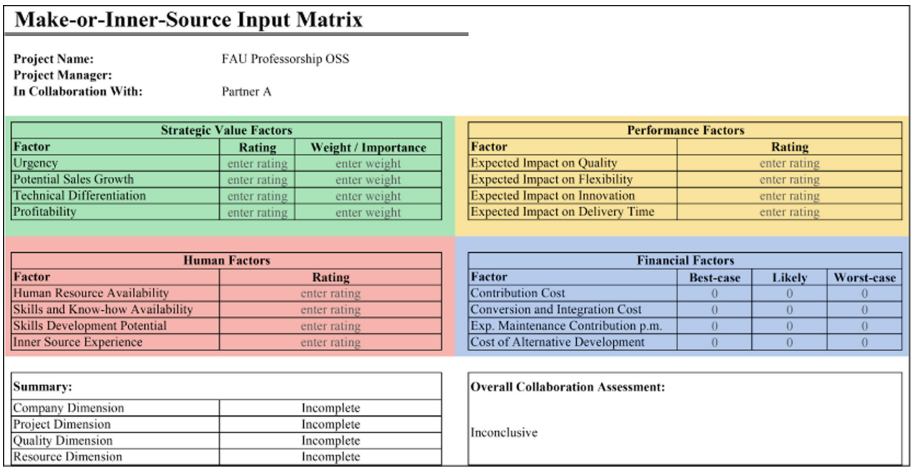
as well as the code commits from the repository, containing the following information. As per the prototype, this is done using a csv-file import, containing a unix timestamp of the contribution, the contributor name, type of change made, phase of development, number of lines of code as well as the number of errors, warnings and resolutions.

From this, the compensation model computes the contributions of each participant, as described under 4.3 and gives an overview of the inner source project for controlling. Taking into account the expected usage of the component under development, it determines whether one team disproportionately benefits from the work of the other(s). The resulting compensation "payment" suggestion is relayed to the individual teams after revision by the controller.

Naturally, for each of the inputs, the model accepts subsequent updates and changes the valuation accordingly. 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 to stakeholders if questions about the process arise.

Viability Model. The viability model extends the previously introduced decision matrix, with a template that guides a potential inner source collaborator through the project evaluation. The contributor is tasked with estimating each of the decision factors, as well as assigning a weight by their relative importance. An example of a contributor's assessment mask can be seen below, including both the color-coded input fields, as well as the real-time model result.

While qualitative data is quantified using a seven-point Likert-scale, financial estimations are made more robust using the three-point-estimation techniques. To improve usability and sanitize user input, ratings and weights are presented as drop-down menus with text descriptors. This information is aggregated and, if deemed necessary, revised by an impartial entity, that can re-adjust factor weights after the fact. The model yields a judgment for the four established inner source dimensions and gives special consideration to financial and scheduling viability, and displays warnings if either are in conflict with important project



Make-or-Inner-Source Input Matrix

Project Name: FAU Professorship OSS
Project Manager:
In Collaboration With: Partner A

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

Fig. 2. Empty assessment view for a potential contributor

parameters, even if the overall parameters may be favorable. To return an overall assessment of collaboration viability, the model draws on custom texts for any possible combination of verdicts, that can be shown to a user.

4.5 Demonstration

We demonstrate the models by illustrating the most important cases, and the models' behavior in them.

Compensation Model. The compensation suggestion is delivered to the user along with usage proportions and the calculated valuation of their respective contributions. One potential outcome can be seen below.

	Team 1		Team 2	
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 €

Fig. 3. Exemplary compensation model result

This model's behavior can be described by four basic cases:

Case one (Usage = Contribution; Modifier = 0; Actual = Planned): In this case, planned values are met and all calculation cancels out. We determine that no party received a disproportionate advantage from the cooperation. No payment suggestion is made.

Case two (Usage_A > Contribution; Modifier = 0; Actual = Planned): Party A contributed less than their usage, but in accordance with prior planning. Compensation to B suggested.

Case three (Usage = Contribution; Modifier > 0; Actual = Planned): All else being equal, a higher error modifier shifts the balance towards the party which resolved more errors. A formerly balanced cooperation may now require compensation from the more error-prone contributor to adjust for the higher workload generated.

Case four (Actual <> Planned): Any deviation from planned values can cause the same shifts in contribution balance as described above. The model automatically accounts for changes in usage, contribution, and any other input value provided.

Viability Model. This artifact judges the viability of an endeavor along the four dimensions, as well as returning independent verdicts of viability considering scheduling and financial aspects. Combining this information, the model returns a general judgement in the following form, already seen in Fig. 2.

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

Overall Collaboration Assessment:

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

Fig. 4. Exemplary viability model result

In total, the model's output can be described by 108 combinations of the verdicts above. To demonstrate its functionality, the following illustrative cases were selected.

Case one (Full Recommendation): In these cases, the non-financial parameters within the main inner source dimensions are overwhelmingly positive, and both financial and scheduling aspects are not prohibitive. Depending on whether they are neutral or positive, the message issued to the user changes.

> **Full Recommendation** - Inner sourcing this project has the potential to increase quality, improve delivery time and reduce production cost.
> **Full Recommendation** - The currently entered non-financial information suggests the project would benefit from inner sourcing and enable faster development at comparable costs.

Case two (Disapproval by non-financial factors): Similarly, if a majority of the model's non-financial dimensions yield a negative result, the model judges the project as a whole to be non-viable. In this case, calculations on affected scheduling and finances are made and provided to the controller, but not specifically displayed. The user may be presented with a message similar to the following examples.

Weak Disapproval - The currently entered non-financial information suggests the project is unviable. A redefinition and in-depth analysis are advised.
Hard Disapproval - The currently entered financial and non-financial information strongly suggests the project is unviable for inner sourcing or the contribution of this department is non-viable. Re-evaluation is advisable after potentially re-defining the project parameters significantly.

Case three (Weak Disapproval on scheduling or financial grounds): In cases where the non-financial factors are positive, but the model determines a higher risk of delays or significantly increased cost, this information is relayed to the user in finer detail. It is ultimately at the project decision-makers' discretion to determine which of these aspects are critical to the success of the endeavor. The verdict may be presented akin to the messages below.

Weak Disapproval - The currently entered financial and non-financial information suggests the project would benefit significantly from inner sourcing. However, as it could lead to a potential increase in development time, inner sourcing is only recommended if development time is not a crucial success factor.
Weak Disapproval - The currently entered non-financial information suggests the project may be viable for inner sourcing. Crucially, inner sourcing is expected to increase overall cost, while improving on development time. At the stakeholders' discretion, the higher cost can be considered acceptable if time to market is prioritized.
Weak Disapproval - The currently entered non-financial information suggests the project may benefit from inner sourcing. However, due to expected disadvantages in development cost and time, alternative development is generally preferable.

5 Limitations

As mentioned, any kind extensive management oversight risks hampering the benefits of inner sourcing in the first place. This 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 value factor, but it's intentionally simplistic in execution. In theory, partners could 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 [27] or a work time calculation from time between commits as proposed by Buchner and Riehle [12].

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. Even so, there are cases where compensation payments are explicitly recommended, such as for taxation purposes [16].

As for the viability model, the calculation could be extended with further negative dimensions associated with inner sourcing, depending on the individual 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.

6 Conclusions

In this paper we demonstrate 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 light-weight 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 and reduce the risk of uncontrolled processes and mismanagement.

The broad sweep approach that was chosen during the theoretical foundation 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 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.

References

1. Deshpande, A., Riehle, D.: The total growth of open source. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (eds.) OSS 2008. ITIFIP, vol. 275, pp. 197–209. Springer, Boston, MA (2008). https://doi.org/10.1007/978-0-387-09684-1_16

2. Volpi, M.: How open-source software took over the world (2019). https://techcrunch.com/2019/01/12/how-open-source-software-took-over-the-world
3. Ahlawat, P., Boyne, J., Herz, D., Schmieg, F., Stephan, M.: Why you need an open source software strategy (2021). https://mkt-bcg-com-public-pdfs.s3.amazonaws.com/prod/open-source-software-strategy-benefits.pdf
4. Bosio, D., Littlewood, B., Strigini, L., Newby, M.J.: Advantages of open source processes for reliability: clarifying the issues. In: Gacek, C., Arief, B. (eds.) Proceedings of the Open Source Software Development Workshop, pp. 30–46 (2002)
5. Lawrie, T., Gacek, C.: Issues of dependability in open source software development. SIGSOFT Softw. Eng. Notes **27**, 34–37 (2002)
6. Stol, K.-J., Avgeriou, P., Babar, M.A., Lucas, Y., Fitzgerald, B.: Key factors for adopting inner source. ACM Trans. Softw. Eng. Methodol. **23**, 1–35 (2014)
7. Riehle, D.: The commercial open source business model. In: Nelson, M.L., Shaw, M.J., Strader, T.J. (eds.) AMCIS 2009. LNBIP, vol. 36, pp. 18–30. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03132-8_2
8. Open Source Initiative: The Open Source Definition (2007). https://opensource.org/osd
9. Capraro, M., Riehle, D.: Inner source definition, benefits, and challenges. ACM Comput. Surv. **49**, 1–36 (2017)
10. Riehle, D., Capraro, M., Kips, D., Horn, L.: Inner source in platform-based product engineering. IIEEE Trans. Softw. Eng. **42**, 1162–1177 (2016)
11. Stol, K.-J., Babar, M.A., Avgeriou, P., Fitzgerald, B.: A comparative study of challenges in integrating Open Source Software and Inner Source Software. Inf. Softw. Technol. **53**, 1319–1336 (2011)
12. Buchner, S., Riehle, D.: Calculating the costs of inner source collaboration by computing the time worked. In: Bui, T. (ed.) Proceedings of the 55th Hawaii International Conference on System Sciences (2022)
13. Capraro, M., Dorner, M., Riehle, D.: The patch-flow method for measuring inner source collaboration. In: Zaidman, A., Kamei, Y., Hill, E. (eds.) Proceedings of the 15th International Conference on Mining Software Repositories, pp. 515–525. ACM, NY (2018)
14. Morgan, L., Gleasure, R., Baiyere, A., Dang, H.P.: Share and share alike: how inner source can help create new digital platforms. Calif. Manage. Rev. **64**, 90–112 (2021)
15. Charifzadeh, M., Taschner, A.: Management Accounting and Control: Tools and Concepts in a Central European Context. Wiley-VCH, Weinheim (2017)
16. The International Federation of Accountants (IFAC): Evaluating and Improving Costing in Organizations. New York, NY, USA (2009)
17. Cánez, L.E., Platts, K.W., Probert, D.R.: Developing a framework for make-or-buy decisions. Int. J. Oper. Prod. Manage. **20**, 1313–1330 (2000)
18. Medina-Serrano, R., González-Ramírez, R., Gasco-Gasco, J., Llopis-Taverner, J.: Strategic sourcing: developing a progressive framework for make-or-buy decisions. J. Ind. Eng. Manage. **13**(1), 133–154 (2020)
19. Deephouse, C., Mukhopadhyay, T., Goldenson, D.R., Kellner, M.I.: Software processes and project performance. J. Manage. Inf. Syst. **12**, 187–205 (1995)
20. Bloch, M., Blumberg, S., Laartz, J.: Delivering large-scale IT projects on time, on budget, and on value (2012). https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value

21. Budzier, A., Flyvbjerg, B.: 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. Commonwealth Secretariat, London (2012)
22. Dingsoyr, T., et al.: Key lessons from tailoring agile methods for large-scale software development. IT Prof. **21**, 34–41 (2019)
23. Larson, E.W., Gobeli, D.H.: Significance of project management structure on development success. IEEE Trans. Eng. Manage. **36**, 119–125 (1989)
24. Boehm, B.W., Papaccio, P.N.: Understanding and controlling software costs. IIEEE Trans. Softw. Eng. **14**, 1462–1477 (1988)
25. Filieri, A., et al.: 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 (2015)
26. Kaner, C., Bond, W.P.: Software engineering metrics: what do they measure and how do we know? In: METRICS 2004. IEEE CS (2004)
27. Mas y Parareda, B., Pizka, M.: Measuring productivity using the infamous lines of code metric. In: Keung, J. (ed.) Proceedings of the First International Workshop on Software Productivity Analysis and Cost Estimation, pp. 4–9 (2007)
28. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manage. Inf. Syst. **24**, 45–77 (2014)
29. Granlund, M., Lukka, K.: It is a small world of management accounting practices. J. Manage. Account. Res. **10**, 153–179 (1998)
30. Bueno, S., Salmeron, J.L.: TAM-based success modeling in ERP. Interact. Comput. **20**, 515–523 (2008)
31. Cappelli, M.: Overcoming the challenges of a complex ERP environment (2016). https://searcherp.techtarget.com/tip/Overcoming-the-challenges-of-a-complex-ERP-environment
32. Driscoll, M., Webb, H., Schmidt, J.: ERP complexity vs. business growth - at odds or in alignment depends on your approach (2015). https://www.cognizant.com/whitepapers/2015-research-on-the-ERP-landscape-Publisher.pdf
33. Dul, J., Hak, T.: To quantify or to qualify: that's not the question. J. Purch. Supply Manage. **13**, 207–209 (2007)