Sustaining Arm's Length Cost Allocations for Highly Integrated Development Functions—An Explorative Case Study of Transfer Pricing for InnerSource Communities[*]

Oliver Treidler Tom-Eric Kunz Maximilian Capraro Michael Dorner

Abstract

Most contemporary developments in transfer pricing relate to intangibles. Appropriately coping with increasingly highly integrated value chains constitutes another driving force. This article examines the application of the arm's length principle in the context of a related-party transaction characterized by integrated collaboration among decentralized business units and the joint utilization and development of intangibles. While the underlying theoretical transfer pricing concepts will be touched upon, we aim to present a case-based application of the arm's length principle. The pragmatic approach presented in this article aims to support practitioners navigating the tradeoff between finding arm's length solutions for increasingly complex, digitized organizations and effectively utilizing available internal data for transfer pricing purposes.

While the OECD recently started tinkering with formulary apportionment approaches for marketing intangibles in the context of the Pillar 1 reforms, day-to-day transfer pricing remains focused on applying transfer prices that are commensurate with the arm's length principles. As such, Chapter VI of the OECD Transfer Pricing Guidelines of 2017 remains perhaps the most important source of reference for practitioners when it comes to intangibles.[1] The same applies to Chapter VIII when it comes to cost contribution arrangements (CCAs). For this article, it is deemed sufficient to limit the references to Chapter VI and Chapter VIII of the OECD guidelines.

1 Integrated Software Development—A Working Definition of Inner Source Communities

Identifying arm's length transfer prices is a highly context-specific exercise and one-size-fits-all approaches are seldomly appropriate, especially when operating outside of the TNMM (CPM) context that is narrowly focused on determining arm's length net margins for routine entities (tested parties). In introducing our fictional case study, it is thus imperative to emphasize that it reflects a highly simplified and generic InnerSource community [1]. In a nutshell, an InnerSource community can be understood as an open-source ecosystem within an MNE, which is based on the Leitmotiv of granting free access to technical software components to all members of the community, which, in turn, contributes to the further development and refinement of the components by making novel contributions and improvements. This Leitmotiv also illustrates that the contributions made within the community are driven by an inherent self-interest of the respective development. While the transfer pricing implications (arm's length allocation of economic benefits) will be addressed below, it must be clear from the start that—at its core—InnerSource has deep cultural and philosophical implications for adopting organizations. The

[*]This article first appeared in MNE Tax on May 25, 2022.

[1]The OECD guidelines were updated in January 2022 (as this article was prepared), but the concepts discussed herein remain valid, and thus, the references were sustained.

concept needs to be embraced by management and the entire organization to sustainably contribute to commercial success sustainably [2].

Organizations that adopt InnerSource often differ from one another. InnerSource is used in organizations of various sizes (from SMEs with under 500 employees to conglomerates with tens of thousands of employees) operating in a variety of sectors such as information technology, financial services, healthcare and pharmaceuticals, industrial and manufacturing, retail and e-commerce and others [3].

Organizations observed a variety of benefits from adopting InnerSource practices:

- **More cost-effective**—Because InnerSource enables more collaboration among an organization's teams, it reduces duplicate work and local workarounds for company-global problems—leading to more efficient and, overall, less costly software development.

  When using an InnerSource component, teams do not have to request features from other—potentially busy—teams but can contribute needed changes themselves without wasting time. This can lead to a significant increase in time-to-market for teams using InnerSource components.

- **Shared effort and costs**—By contributing to an InnerSource component jointly with other teams or providing its own InnerSource components, a team can share development effort and risk with other parties in the InnerSource community.

- **Knowledge sharing and improved quality**—The intricate interplay between users, contributors, and maintainers of an InnerSource component (see section Y) leads to an increased flow of know-ledge and information through an organization, resulting in improved quality of the software components and ultimately products.

While many InnerSource communities will exhibit broadly similar properties, a careful analysis must be conducted in each individual case. In this context, it is particularly important to be aware that the scope of applying a CCA-type of solution will hinge on ascertaining the tested transaction sharing the same core properties of the fictional InnerSource community introduced in the following.

The main characteristic features of software development within an InnerSource community can be summarized as follows:

- **R&D-specific**—Inner source is limited to (software) development functions. Downstream business processes (such as distribution of the software or products featuring software components) are independent of InnerSource practices. While the development function is thus highly integrated, with regard to other functions, the level of collaboration (integration) between legal entities within a group will (often) not reflect a highly integrated value chain.

- **Fine-grained**—The contributions to an individual InnerSource component (project) are often small (incremental) improvements. Individual developers may merely contribute a small sub-functionality or fix a bug of an existing issue relevant to their work.

- **Role-based**—Developers, each of whom can be associated with different legal entities and/or business units within the MNE organization, may simultaneously take on multiple roles in an InnerSource community. As such, they are often active in communities for different software components, reflecting the complex, product-oriented organizational set-up often encountered in decentralized software development. Therefore, a role-based view needs to be adopted to appropriately capture the activity (as well as the related costs and value contributions) of individual developers to a specific software project.

Considering that the role-based perspective is a unique feature of InnerSource, a brief introduction to archetypical roles is essential. The informal groups that form around individual software components within the InnerSource community can be divided into four non-exclusive roles: providers, users, contributors, and maintainers.

Providers initially make a software component available to the InnerSource community. Subsequently, every member of the community can select these software components and use them as tools for their own work or as a component in products (in the role of the user) and contribute improvements to them (in the role of the contributor). These contributions are checked for quality by particularly experienced developers (in the role of maintainer). Often, but not always, the providers will also act as maintainers for the component they had initially provided to the InnerSource community. In some cases, the maintainer will integrate the new contributions into the existing software component. In principle, employees from all entities of an MNE adopting InnerSource can simultaneously hold several roles regarding one or more InnerSource components at the same time.

Another important component-independent role is the so-called Inner Source Program Office (ISPO). An InnerSource community is a coordinated initiative to promote inner-source practices within an organization. As such, the ISPO performs functions critical to the successful adoption of InnerSource. Often, the role of the ISPO is organizationally located as a standing department close to the CTO. It is to be noted that the ISPO does not act as the owner (entrepreneur) of the community and that different legal entities can jointly perform the role (activities) of an ISPO both in a formalized or informal manner.

2 Initial Transfer Pricing Considerations

From a transfer pricing perspective, the most intriguing (challenging) feature of InnerSource is that employees from multiple related entities closely collaborate on one specific stage of the value chain. The complexities encountered when dealing, for example, with matrix organizations are exacerbated by the fact that individual employees may simultaneously assume multiple component-specific roles.

Given these properties, appropriately delineating the tested transaction is conceptually not exactly a straightforward task. Pursuant to paragraph 3.9 of the OECD guidelines, the arm's length principle should be applied on a transaction-by-transaction basis. For the case at hand, strict adherence to a transaction-by-transaction approach would, however, imply the necessity to conduct a separate valuation of the contributions to, and benefits derived from, each software component (arguably including an analysis of legal ownership and DEMPE functions pursuant to Chapter VI Section B of the OECD guidelines). Looking closely at the Leitmotiv of InnerSource and the economically relevant characteristics (in the sense of paragraph 1.36 of the OECD guidelines), however, we would suggest that adopting an aggregated perspective constitutes the more reliable and sensible application of the arm's length principle in the context of InnerSource. Specifically, the tested transaction should be conceived as the framework governing the rules of the communities agreed by the participants. In other words, independent parties would base their decision for participating in an InnerSource community on whether they reasonably anticipate that the costs (contributions) individual participants need to bear will be in a commercially sensible (advantageous) proportion to the benefits derived from the participation. As such, the decision to agree to an InnerSource community's governing framework closely mirrors the preconditions for applying a CCA pursuant to Chapter VIII of the OECD guidelines.

Applying the provisions of Chapter VIII to InnerSource thus seems sensible. Because of the nature of the collaboration, (component-specific) one-sided transfer pricing methods are not deemed conceptually suitable (no clear-cut tested party) and would be impractical to implement on a transaction-by-transaction basis. A two-sided approach (applying the profit split method) would also be conceptionally questionable, as only one distinct function of the value chain (development) is subject to the InnerSource community. Consequently, the correlations of contributions to the InnerSource community to the ultimate profitability of the separate legal entities (business units) will generally be too weak, especially when considering that the decisions (risks) relating to the utilization of individual components as well as the (more complex) downstream activities are allocated to the separate legal entities. Also, valuation methods, i.e., valuing individual software components (and deriving license fees), will generally not be a good fit for InnerSource, as it is rather an exception that patents protect the relevant InnerSource components themselves.

The highly integrated development function within the community is characterized by the collaboration among the participating entities being focused on creating mutual benefits. As such, one of the most important prerequisites of applying a CCA, the alignment of commercial interest, is fulfilled. This can best be visualized by thinking of a club among equals in which the member entities have the same rights and obligations in using the software components without being subjected to a hierarchy, in contrast to traditional service transactions, such as contract development, which are characterized by a principal assigning specific tasks to the service provider. While establishing the applicable cost base for a service transaction is rather straightforward, the delineation of cost components in the context of a CCA requires a more nuanced analysis, i.e., it must be defined which costs (activities) of a member are qualified as contributions for which the other members are willing to pay. In addition to identifying the relevant costs, it must be analyzed whether the contributions of individual members of the community can reasonably be valued at costs or whether (some) contributions contain more commercial value (often due to utilizing unique intangibles) that would require a higher compensation at arm's length as the benefits derived from having access to the contributions exceed costs.

In setting up a transfer pricing structure for a CCA or InnerSource, it is thus essential to address the question of whether the costs borne by participants making contributions correspond (in the sense of being proportionate) to the benefits derived by other participants. If the relationship between costs and benefits is not (at least approximately) proportional, the imbalance must be corrected (adjusted) through balancing payments. Applied to InnerSource, we will thus primarily have to address the challenge that not all users are necessarily also contributors to a specific component. Since corresponding *users only* do not actively contribute to components or provide any other service in return for utilizing the contributions of the other members, they must be given special consideration in the transfer pricing context to ensure that these "users only" participants (of each InnerSource component) appropriately share in the costs of the community.

On the other hand, it must also be ensured that participants cannot push costs into the cost pool, which are related to contributions that do not convey a commercial benefit to other participants (i.e., no or very few users of a specific InnerSource component other than its provider), as otherwise, the total cost pool would potentially be ballooning and render joining the community unattractive and contradict the community ethos (akin to a prohibitive entrance fee to a club).

Notably, the identified challenges are not only relevant for transfer pricing practitioners but also for management advocating the adoption of an InnerSource, as respective rules will contribute to facilitating the perception of enforcing fairness (prevent free-riding). As the mutual benefits are commercially dependent on members collaborating and exchanging ideas, the effectiveness of InnerSource depends on defining rules of the game that facilitate an inclusive collaboration.

3 An Illustrative Case Study

The following fictitious case study will prototypically illustrate that the internal data available within an InnerSource community (specifically, so-called patch-flow data [1]) will generally be sufficient to address the challenges from a transfer pricing perspective appropriately.

3.1 Properties of the Inner Source Community

The functional and risk analysis can generally be conceived as the centerpiece for any transfer pricing analysis. While each InnerSource initiative will inevitably have idiosyncratic properties, it can reasonably be assumed that the relative importance of the archetypical roles introduced above will be similar. In this article, we confine the analysis to an abbreviated RACI analysis, focusing on delineating which processes (functions) are attributable to the individual roles—introduced above[2]. It must be kept in mind that, as

[2]In day-to-day practice, the scope of the analysis should be enhanced and substantiated by interviews with members of the respective InnerSource community.

emphasized above, individual employees will often simultaneously perform multiple roles in the context of multiple InnerSource components.

The relevant processes (functions) to be analyzed, as well as the working assumptions (characteristics) of our representative InnerSource community, can be summarized in Table 1. The five main processes are labeled A to E, the sub-processes, if any, are labeled by a consecutive number. Again, it would be conceptually feasible to extent the analysis by identifying further sub-processes, but for a high-level transfer pricing analysis the analysis presented below is deemed sufficient.

Table 1 and Table 2 summarize the result of our case study's indicative functional and risk analysis. To assess, at least indicatively, the value-added contributions of individual roles by applying the RACI methodology, it should be noted that for

$\boxed{R}$	(responsible)	follows with somewhat higher value-added as it involves operational—often routine—functions;
$\boxed{A}$	(accountable)	reflects (strategic) ownership and the respective management and supervisory function;
$\boxed{C}$	(consulted)	reflects special, valuable input in the context of InnerSource, such as entrepreneurial decisions of unique know-how.
$\boxed{I}$	(informed)	is deemed characterized as the lowest value-added as it involves minimal and (often) passive participation;

The RACI analysis illustrates the integrated nature of cooperation within the InnerSource community. Specifically, it illustrates that all participants are expected to assume the roles of provider, contributor, and user in different constellations depending on the specific component. It further illustrates that while the user is completely autonomous with respect to the utilization of a component (D1), it does, when viewed as a stand-alone role, not contribute to the main value-added process of the community (B1 – B3). As such, the analysis reemphasizes the need for imbalances within the community (user only) to be addressed by balancing payments. The commercialization (based on components derived from InnerSource) is considered as downstream function—independent of InnerSource—with all members of the community acting as entrepreneurs for their respective business (unit).

Another key aspect highlighted in the RACI analysis is that the component-independent role of the ISPO requires special consideration. In this context, it is noteworthy that the role of ISPO is focused on facilitating efficient operations. However, neither the respective management functions (C1 – C3) nor the protection functions (E1 and E2) are deemed core functions of the ISPO in the case at hand. Notably, to adhere to the Leitmotiv of InnerSource, all members need to voluntarily embrace the strategic framework and the respective governing rules (indicated by $\boxed{C}$ in C1). As such, it is deemed appropriate to compensate the ISPO for the component independent support functions at cost (i.e., qualifying the costs incurred for allocation through the cost pool)[3]. In respect to the provider, it is noteworthy that the contribution for A1 and A2 can be measured on a stand-alone basis, adhering to the working assumption formulated above a valuation at cost appears sensible[4].

3.2 Validating Application of a CCA Framework

A key feature of CCAs is the sharing of contributions (paragraph 8.5 OECD guidelines). In the case at hand, the contributions can be determined using the patch-flow method for measuring InnerSource collaboration: The patch flow is the flow of code contributions across organizational boundaries [1].

[3]Specifically, the costs for performing a code review (B2) attributable to the maintainer function can be reliable assessed based on actual (estimated) time spend. A mark-up on costs is not deemed to be required in the case at hand. One reason is that in most cases (as assumed for our case study), the employees performing the role of maintainer will be employed by a legal entity that also employs (a substantial number of) individuals using InnerSource components from the InnerSource community.

[4]The costs of developing the initial seed (A1) and providing a component to the InnerSource community (A2) are assumed to be available based on internal cost accounting data (What is the cost of an engineer per hour?) and component-specific planning and tracking information (How many engineers worked on the component for how long?).

	InnerSource Process	Description
A	**Initial provision of an InnerSource component**	
A1	Developing seed component	Development of the initial software component ("seed") [4]. The value (as opposed to the costs incurred) of initial software development can often (not) be reliably estimated—as for any software.A specific feature of a software component might be highly relevant to the success of one business unit's mission and insignificant to this of another. We are not aware of a method to (semi-)automatically quantify the benefit derived from a specific software component or feature.
A2	Provision for the InnerSource community	Making the software accessible to the InnerSource community, it is necessary to provide the component as InnerSource software (e.g., minimizing technical dependencies, designing guidelines for contributions, writing guidelines and tutorials for usage, setting up the project page or wiki, ...).
B	**Code Contributions (Patch Flow)**	
B1	Performing code contribution	Writing and contributing a code contribution (e.g., fixing a bug or adding functionality to a software). For this case study, we assume that for a period of time (i.e., the first year after provision), the provider will still make a large majority of contributions. The share of contributions from other contributors gradually increases over time.
B2	Review of code contribution	Performing quality control (so-called "code review") of contribution and share feedback, which allows the contributors to learn and improve.
B3	Integration or rejection of code contribution	Deciding whether to integrate the contribution and, if necessary, exercising the respective integration. One core assumption is that most rejected code contributions have no value to an InnerSource component – even if costs were incurred (this assumption is deemed consistent with the safeguard against ballooning the cost pool).
C	**Management of the InnerSource program**	
C1	Management of the Inner-Source program	Overall strategic responsibility (budgeting, goal setting, etc.) for adopting InnerSource in the MNE.
C2	Operating infrastructure	Operating the necessary infrastructure for hosting InnerSource components and collaborating on them (e.g., code hosting platforms like Github Enterprise or Gitlab).
C3	Internal marketing & coaching	Promoting the InnerSource program (and thus indirectly the InnerSource components) and explaining InnerSource practices within the MNE.
D	**Utilization**	
D1	Utilization of an InnerSource component	As emphasized above, the decision to use InnerSource components and how to utilize them is—without limitations—at the discretion of the individual members of the InnerSource community.
E	**Application for patents, protective measures**	
E1	Filing for patents	Patents can be neglected for this case study: We assume that InnerSource components themselves are not protected by patents, while products built on top of them might be protected directly by legal entities in the role of user (see D above).
E2	Technical protective measures	Technical protective matters can be neglected for this case study since these are taken by default as part of the operation of the MNE's software development infrastructure.

Table 1: Typical processes in InnerSource projects and programs.

	Inner source process	Component-independent		Component-specific			
		ISPO	Member	Provider	User	Contri-butor	Main-tainer
A1	Developing initial seed component			R A			
A2	Provision for the InnerSource community		I	R			
B1	Performing code contribution		I	R		R	A
B2	Review of code contribution		I		I		C R
B3	Integration/rejection of code contribution		I			I	R A
C1	Management of InnerSource program	R A	C				
C2	Operating development infrastructure	R	C I				
C3	Internal marketing & coaching	R A	R				
D1	Utilization of an IS component				R A		
E1	Filing for patents	A	I				
E2	Technical protective measures	A	R				

Table 2: Roles and their functions and risks in typical InnerSource processes.

Measuring the patch flow allows for both a detailed and an aggregated breakdown of the collaboration within the InnerSource community.

For the development of intangible goods, each participant is contractually granted a property right in the intangible resulting from the CCA's activities (or a right to use or exploit depending on legal circumstances). As each member of the InnerSource community has the right to use or exploit the developed software, there is no need to pay a license fee or any other consideration for the use of the developed software (paragraph 8.11 OECD guidelines).

According to OECD guidelines paragraph 8.12, a CCA differs from any other intragroup transfer of property or services in that part or all of the consideration intended by the participants is the expected mutual and proportionate benefit from the pooling of resources and capabilities, i.e., exactly what constitutes the Leitmotiv of InnerSource (see Section 2).

Based on the above, the following guiding principle seems most appropriate for the arm's length design of the InnerSource transfer pricing system, "those who use but do not contribute must pay."

Conversely, however, this also means that costs individual members incur for contributions that are exclusively in the member's own interest—or relate to clearly identified InnerSource components—may not be transferred to a general cost pool; i.e., the costs qualified for allocation among InnerSource participants are thus deliberately limited. Therefore, it is necessary to ensure a sufficiently differentiated collection of (value) contributions for an appropriate delineation of costs or benefits. In the interests of proportionality, it should be noted that the level of detail of the analysis should be based on business considerations—i.e., an evaluation should be carried out based on data that is collected for business reasons.

The guiding principle emphasizes the community idea as well as the inherent self-interest of the entities to participate in such a community. It is thus consistent that the determination of compensation payments is not based on a stand-alone market value for individual contributions (e.g., for the providers on process A1 or individual contributions B1), which neither by applying a CUP nor valuations can be reliably determined. Instead, the arm's length compensation payments can appropriately be derived based on the recorded costs as well as from the distribution of roles within the InnerSource community assumed by employees associated with specific legal entities. In our view, three interrelated implications are applicable to operationalize the guiding principle:

- Legal entities are *users* of, and contributors to (or providers of), an InnerSource component at the same time. In such a case, no compensation for contributions is due to the imminent self-interest of the contributing entities. Possibly, reimbursement of a part of the costs might be due in case legal entities that exclusively act as users can be identified (see next point).
- Legal entities identified as *users only* of an InnerSource component. In such a case, the users must bear an appropriate part of the costs incurred by the provider(s) and contributor(s). The respective balancing payments should ideally be calculated to appropriately capture the proportionate benefit (e.g., share or intensity of utilization).
- A legal entity that is only a *provider/contributor and not a user*. In this case, the provider/contributor would have to be reimbursed separately as an external service provider (contract developer or service provider outside of the CCA). The costs would have to be distributed within the community (see also paragraphs 8.14 and 8.17 of the OECD guidelines)[5].

The maintainer as a component-specific role needs to receive special consideration since the maintainer has to bear additional (review) costs because of its central function for the benefit of all users (review of the contributions, quality assurance by a decision on integration of the contribution the users need to pay an arm's length compensation for these functions).

In addition to determining arm's length balancing payments on component-specific data, an arm's length remuneration for the component-independent functions of the ISPO must be determined.[6]

3.3 Utilizing Patch Flow Data to Calculate Adjustment Payments

The relevant patch flow data is available and can be extracted for transfer pricing purposes. By analyzing the data, three distinct tiers can be identified to determine balancing payments commensurate with the underlying assumption that those who use but do not contribute must pay (see above). Figure 1 illustrates this assumption by the intersection.

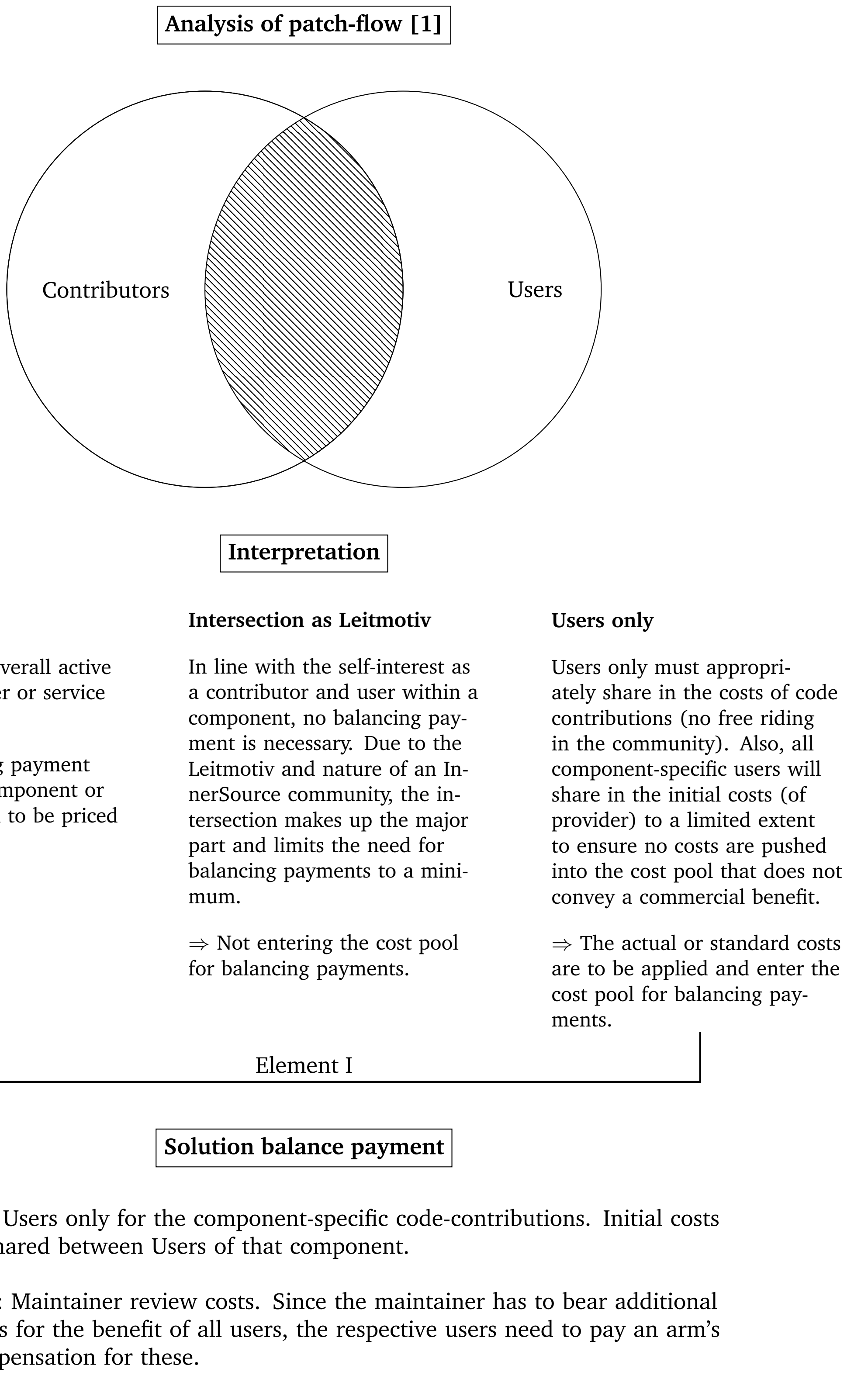
Thus, only the respective non-intersections (non-hatched area) are relevant for balancing payments (compensation) and must be determined. In the context of the right set of contributors, the data analysis must focus on whether the entity contributes within other projects/components or, in general, only provides contributions without any interest in utilizing components and actively participates in the community. The latter case reflects a distinct transaction characterized as a provision of services (contract developer). Therefore, it would have to be reimbursed separately and costs would be borne by the whole community (see above).

The balancing payment consists of two elements, whereas Element I, in connection with the right set of users, is about the compensation for only utilizing an InnerSource component, i.e., no contribution of value (code), thus balancing code contributions of the other members. In the case of code contributions, assuming that the provider/contributor is a user within the overall community, no cost markup is required here; rather, the ACTUAL or standard costs are to be applied.

Element II of the balancing payment is for the compensation of the maintainer activity (value-added). Since the maintainer has to bear additional (review) costs due to its central function for the benefit

[5]For simplicity, we do not elaborate on this issue. However, analysis of the patch flow would easily allow for identifying respective entities. For transfer pricing purposes, it is deemed critical to keep service providers (contract developers) outside of the scope of the CCA. As the Leitmotiv does not apply to the service providers (they do not share in the mutual benefits), they do not qualify as CCA members.

[6]For the purpose of the case study, we assume that the costs for operating the development infrastructure (C2) are fixed and set by the ISPO. As these costs have pass-through character and do not reflect valuable contributions of the community members, it seems appropriate to isolate these costs from the calculation of balancing payments.



Contributors only

Analysis whether overall active community member or service provide

⇒ Either balancing payment by users of that component or distinct transaction to be priced separately.

Intersection as Leitmotiv

In line with the self-interest as a contributor and user within a component, no balancing payment is necessary. Due to the Leitmotiv and nature of an InnerSource community, the intersection makes up the major part and limits the need for balancing payments to a minimum.

⇒ Not entering the cost pool for balancing payments.

Users only

Users only must appropriately share in the costs of code contributions (no free riding in the community). Also, all component-specific users will share in the initial costs (of provider) to a limited extent to ensure no costs are pushed into the cost pool that does not convey a commercial benefit.

⇒ The actual or standard costs are to be applied and enter the cost pool for balancing payments.

Element I: Users only for the component-specific code-contributions. Initial costs are to be shared between Users of that component.

Element II: Maintainer review costs. Since the maintainer has to bear additional review costs for the benefit of all users, the respective users need to pay an arm's length compensation for these.

Figure 1: Overview from analysis of patch-flow data [1] and its interpretation to balance payments as solution.

of all users (review of the contributions, quality assurance by the decision on the integration of the contribution the users need to pay an arm's length compensation for these. The review costs are to be shared between the users of the component. For that, there are two options: Options A is a lump-sum as a percentage charging of labor costs for maintainers (e.g., 10% of labor time), bandwidth, or tiering if applicable. Option B is the approximation of costs by number of reviews and their average duration. Due to the asynchronous nature, measurement can be difficult and must be assessed on a case-by-case basis. It is assumed that if there is more than one legal entity with a maintainer role, they all contribute the same share to the code review.

4 Conclusion

While the case study presented here constitutes an admittedly simplified and idealized situation, the core insights and conclusions should provide sensible guidance for further discussions and refinement, which most importantly needs to facilitate an intense discussion between the companies using InnerSource and transfer pricing experts. On the one hand, companies using InnerSource need to grasp the fact that the tax implications of the collaboration (sharing and joint development of intangibles) are inescapable, largely because InnerSource—from a tax perspective—is not equivalent to open sourcing vis-a-vis third parties (thus free access is not commensurate with arm's length conditions). On the other hand, tax practitioners and tax authorities need to understand and embrace the Leitmotiv of InnerSource. Based on such an understanding, it should follow that—for tax purposes—the intersection illustrated in the above section must be accepted as justifiably being out of scope for balancing payments. While defining the intersection precisely will remain a challenge, it should first and foremost be understood that InnerSource is not about shifting profits and eroding tax bases, as no participating entity will surrender highly valuable intangibles to other entities (especially not to entities lacking economic substance that might be in tax-friendly jurisdictions). Thus, tax authorities need to understand that the stakes are not as high as compared to other IP transactions in the realm of transfer pricing. For companies using InnerSource, such a mindset and framework would constitute an enormous relief, as the uncertainty as well as risks and complexities currently being associated with transfer pricing inhibit the adoption of InnerSource to the detriment of efficiencies. The underlying proposition of harnessing the benefits of InnerSource is that all participating companies will eventually exhibit an improved bottom line, which, in turn, should also appease the taxman.

Oliver Treidler is the managing director at TP&C GmbH. Contact him at `ot@tp-and-c.de`. **Tom-Eric Kunz** is a consultant with TP&C GmbH. Contact him at `tk@tp-and-c.de`. **Maximilian Capraro** is a member at InnerSource Commons, a non-profit foundation connecting practitioners and those who want to learn about InnerSource. Contact him at `max@capraro.net`. **Michael Dorner** is a software engineering researcher with Blekinge Institute of Technology, Sweden. Contact him at `michael.dorner@bth.se`.

References

[1] Maximilian Capraro, Michael Dorner, and Dirk Riehle. "The patch-flow method for measuring inner source collaboration". In: *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*. ACM Press, 2018, pp. 515–525. DOI: 10.1145/3196398.3196417.

[2] Maximilian Capraro and Dirk Riehle. "Inner Source Definition, Benefits, and Challenges". In: *ACM Computing Surveys* 49 (4 Dec. 2016), pp. 1–36. DOI: 10.1145/2856821.

[3] InnerSource Commons. *State of InnerSource 2021*. https://innersourcecommons.org/learn/research/state-of-innersource-survey-2021/. 2021.

[4] Klaas-Jan Stol et al. "Key Factors for Adopting Inner Source". In: *ACM Transactions on Software Engineering and Methodology* 23 (2 2014), pp. 1–35. DOI: 10.1145/2533685.