Inner Source—Adopting Open Source Development Practices within Organizations: A Tutorial

Klaas-Jan Stol and Brian Fitzgerald

Lero—the Irish Software Engineering Research Centre, University of Limerick, Ireland

Abstract. Inner source, the adoption and tailoring of Open Source development practices inside organizations, is a topic of increasing interest. While Inner Source offers a number of benefits, in our experience many practitioners are unclear as to what Inner Source is, and what steps to take towards adoption. In this article we present a tutorial in which we outline nine key factors, pertaining to product, process and organization, which we have found to be important in working with organizations who are interested in Inner Source. This paper illustrates these nine factors with three inner source initiatives that we have studied.

Keywords: open source development practices, inner source, adoption, key factors, tutorial

Introduction

Open Source has had an enormous impact on the software industry.[1] Open Source Software (OSS) is widely adopted by software development organizations in a variety of ways. Besides adopting OSS *products*, either as productivity tools or as off-the-shelf components, a number of organizations have adopted open source *practices* to develop their software. This is known as 'inner source' as the software is 'sourced' internally, although different terms have been used such as "progressive open source" and "corporate open source."[2] Unlike traditional approaches, developers of an inner source project do not belong to a single team or department. Instead, anybody within the confines of the organization can become a contributing member of this internal community, either as a user or contributor. Raymond compared traditional software development approaches to building cathedrals, while referring to open source style development as a "Bazaar."[3] Hence, inner source may be viewed as a bazaar within a corporate cathedral.[4]

Several large organizations have adopted inner source over the last decade. An early study described the experiences of Hewlett-Packard,[5] followed by other company experience reports including Alcatel-Lucent,[6] Philips Healthcare,[4] IBM[7] and SAP.[8] Each of these companies has taken its own approach to adopting inner source. While all development methods must be tailored to an organization's specific context, inner source is not a *defined* methodology, such as for instance Scrum, which defines a number of roles, ceremonies and artifacts.[9] Instead, inner source is best captured as a philosophy, using those practices from open source communities that can greatly add value to an organization's development approach. Some common open source practices are:[2]

- Universal access to all development artifacts such as code and documentation, and allowing anyone to inspect and submit contributions;
- Independent peer-review of contributions by others in the developer "community";
- Informal communication channels, such as mailing lists and Internet Relay Chat (IRC) channels to allow for ad-hoc communication that can be archived for later reference;
- Self-selection of tasks to allow developers to identify parts of the software that they feel they can improve, or defects to fix;
- Early feedback and frequent releases to keep a project alive and quickly improving.

This is by no means an exhaustive list of practices, but the above are a very common subset in successful OSS projects such as the Linux kernel and the Apache webserver.

There is an increasing interest in adopting inner source as it can lead to potential benefits such as the following:

- Increased level of software reuse through software products and components becoming available as inner source projects for anyone within an organization to use.[5,7] In many organizations, teams cannot access other teams' source code.
- Improved quality through leveraging of Linus's Law[3]—*Given enough eyeballs, all bugs are shallow*.[5, 8]

Digital Object Indentifier 10.1109/MS.2014.77 0740-7459/$26.00 2014 IEEE

- Open Innovation through taking advantage of the broad expertise and creativity of the whole developer pool within an organization, rather than just one department or team.[8, 10, 11]
- Increased speed of development with a community potentially as large as the organization's entire development staff which can result in a faster time-to-market.[4, 12]
- Improved mobility of personnel as developers become more familiar with various software projects as well as the tools used in a central platform repository.[5]

While these benefits are very appealing, organizations may face challenges in understanding where to start or what to expect, and as a result practitioners and managers typically have many questions, such as:

- What software product would be suitable to inner-source?
- How is development of an inner source project different from a 'traditional' software project?
- Why would developers want to start or get involved in an inner source project?

In this article, we aim to shed some light on these questions. Based on a comprehensive review of existing research as well as our actual hands-on experience in a number of organizations, we identified a set of nine key factors that are important for organizations to consider when adopting inner source.[2] The nine factors are clustered into three higher-level categories: 1) those pertaining to a software product's suitability; 2) practices and tools, and 3) people and management factors. Together, the factors comprise a 'framework' to understand inner source initiatives—in this article we present three such analyses of existing inner source initiatives. These three cases differ greatly in size, domain and level of activity, and provide insightful examples of inner source. We collected the data for these case studies through interviews with key people involved as well as supporting documentation on these initiatives. We recommend that organizations interested in adopting inner source use these nine factors as a lens to gauge where they stand, and to guide implementation of their inner source initiatives. An assessment of these factors can help identify a product suitable for inner-sourcing, as well as identifying potential barriers that would have to be overcome to achieve inner-sourcing success. Based on our observations in the three cases presented below, we offer a number of next steps (see Table 1).

Software Product

Seed Product

The first step towards an inner source initiative is that of selecting an appropriate 'seed product,' an existing initial implementation of a software product or component. Similar to open source communities, an inner source project is very difficult to start from scratch—without an initial vision of a project, it's hard to attract developers from across an organization to invest time and resources. Instead, it's much more useful to have an initial seed product that can attract a developer community and grow to a successful inner source project. This seed project must offer sufficient *value* to an organization: starting an inner source project around a new operating system or database management system is unlikely to attract many contributors because building such commodity software is rather wasteful.

At Philips Healthcare, the inner source initiative started with a component suite built around the DICOM (Digital Imaging and Communication in Medicine) standard that is widely used in systems for medical imaging. Philips Healthcare develops various products in this domain, such as X-ray and MRI scanners. This component suite, which evolved to a platform for Philips' software product line, had significant business value, and developing it in-house was therefore a logical decision.

A second example is the Session Initiation Protocol (SIP) stack implementation developed at Lucent (who later merged with Alcatel). The initial implementation was written by a single developer in the late nineties. At that time, SIP was becoming an important standard in telecommunications, and there were no suitable implementations available that were complete, affordable, and of high quality. Therefore, developing it internally proved to be a good decision, as it represented significant business value to the company. Over time, the source code was made available to others within the company, attracting an internal community of users and contributors.[6]

The third example is the implementation of a file format, Philips File Standard for Pictorial Data (PFSPD), which originated within Philips Research (a different division from Philips Healthcare). This format had been developed internally for storing video sequences, a technology that was not as commonplace when it emerged in the nineties, at which time the processing capabilities of PCs were still limited and required special hardware. The file format was used for research on video processing algorithms that subsequently could be implemented in hardware. Over time, hundreds of different algorithms were implemented, all using the PFSPD format.

Stakeholders

A second factor to consider is that the seed product must have a variety of stakeholders so that it can benefit from contributions throughout the organization. A variety of stakeholders can result in contributions and requirements from different product groups or business units, introducing more diversity in the contributions. If only a single team or project has a stake in a project, then there is no real benefit from

Digital Object Indentifier 10.1109/MS.2014.77 0740-7459/$26.00 2014 IEEE

inner-sourcing the project, as the 'wisdom of the organization/crowd' will not be leveraged.

The SIP stack implementation at Lucent benefited greatly from its inner-sourcing strategy. SIP is a text-based protocol, and thus requires a parser to parse SIP messages. The SIP protocol is rather complex, and while the initial implementation worked well, there was still room for improvement. Developers from elsewhere in the organization offered such improvements; one developer, for instance, had extensive parsing expertise and contributed an improved parser implementation. Other contributors were also able to make improvements to parts of the code.

Philips Healthcare's medical software product line platform became an important part of a wide variety of products developed within the company. As many different stakeholders had an interest in this platform, it made sense to pool resources to further develop it. Since the products vary widely in their requirements, the platform also benefited from a range of improvements from the various stakeholders.

At Philips Research, the initial version of PFSPD was written by a few developers who identified a need to collaborate with teams based in the USA. These teams did not have the specialized hardware infrastructure. A new version of PFSPD for (at that time) newer platforms running Linux therefore became a necessity to collaborate closely with other teams for research and implementation of video algorithms.

Modularity

The third product-related factor to consider is a project's modularity. While modularity is a generally desired feature of software, it is of particular importance for community-based development. Firstly, it allows developers to focus on learning a subset of the overall code to which they can then contribute in a meaningful way. Secondly, it facilitates parallel development: different developers can work on different parts of the project at the same time, in parallel, without any merge conflicts when contributions are checked in at the same time.

While an inner source project should have sufficient functionality to be interesting enough to attract contributors, additions that compromise the module's cohesiveness should be prevented. If not, the module may become too heavyweight and harder to reuse.

Within Philips Healthcare, for instance, the initial component suite offered too many combinations—Philips architects therefore decided to evolve it into a more integrated and pre-tested platform that was highly configurable, thus making it easier to use correctly.

PFSPD benefited from a well-designed API early on in the project. Additional utility tools (39 in total) were developed around the main library, such as converters, comparison tools and viewers. Given that these were developed as separate tools, the tool chain exhibited a great degree of modularity. At some point, maintaining and releasing these different tools was found to be a burden, and it was decided to integrate them into a single command-line application, while maintaining the original modular architecture.

Practices and Tools

Development Practices

The development approach for an inner source project differs in significant ways from conventional approaches, including contemporary ones such as agile methods. It's important that developers are comfortable with a more flexible approach, since new ideas (e.g., features or improved algorithms) may be turned into code quickly, which is then peer-reviewed by the community of developers throughout the organization. This is different from conventional approaches where requirements are identified before their implementation—in open source development, requirements are sometimes characterized as being *asserted after the fact*—features are added by developers who perceive certain missing functionality from their perspective.[13]

The degree to which organizations can adopt such flexible approaches will differ widely. Philips Healthcare, whose medical devices are subject to strict regulations (such as from the US Food and Drug Administration), have adopted a hybrid approach that allows them to comply with necessary regulations while gaining the flexibility offered by open source practices. In their *co-development* projects, business units (the 'customers') and the core team (who manage the shared platform) work together on newly identified features and requirements. Such 'co-development' ensures sufficient domain expertise in the implementation, as well as compliance with the platform's architecture. Should a business unit urgently require a certain feature for a product release, then they are free to make local changes to the platform. While an open source license would require any changes be contributed back to the project, in inner source this is not required since licensing is generally not an issue for internal development—the software is still proprietary, after all. However, it would be highly desirable to give back any changes to the core team so that other business units can also benefit from such additions, and the business unit doesn't need to carry the burden of maintaining their private features.

PFSPD was never an 'official' project but rather an initiative by motivated developers to tend to their needs. Since development was extra-curricular, work on PFSPD should be characterized as a series of development 'bursts'; whenever a new feature of improvement was necessary, and time was available, developers would work intensively on the project. Before a developer would work on a feature, this

Digital Object Indentifier 10.1109/MS.2014.77 0740-7459/$26.00 2014 IEEE

intention was usually announced on the mailing list, which sometimes resulted in feedback on the proposed approach.

Quality Assurance

Open source communities have developed a number of unique quality assurance practices that sets them apart from traditional software development approaches. For instance, eer-review of contributions by the developer community is a highly effective way to solicit feedback on contributions. Peer-review is not unique to open source development, but the large scale on which this can happen is—as reflected by the aforementioned Linus's Law. '*Release early. Release often*' is another motto commonly found in open source communities which helps to solicit early feedback on new code.[3] Time-based and frequent releases is also useful as it facilitates a stable rhythm of feedback that eliminates the problem of developers rushing to have their new features included shortly before a new release as they are unsure when the next release will happen—if a feature doesn't make it for the upcoming release, future releases will happen in a predictable manner.[14]

Within Philips Healthcare, making new releases of the platform of several millions lines of code is not trivial, and typically new versions are released twice a year. However, the core team regularly makes snapshots of the latest development version and some teams use this opportunity for frequent integration to keep the feedback loop as short as possible. These teams found this proactive attitude worked much better than waiting for the regular, less frequent releases, at which point a lot of integration testing would have to be done.

A key element of ensuring quality in the PFSPD project was that any problem reports from users would immediately be investigated. This way, issues were quickly resolved, which also built users' confidence in the project's quality.

Tools

A seemingly trivial issue is the availability of a common set of development tools throughout an organization. However, many organizations have grown organically over an extensive time period, where acquired businesses have become business units or departments, each with their own set of tools. Different departments may also have a high degree of autonomy in what tools are used. These heterogeneous and incompatible tools may prove to be a barrier to sharing contributions or even being able to build and run the software on certain platforms.[5, 8]

Philips Healthcare is such a 'federated' organization that has grown over time; they have overcome the issue of disparate tools by using a standardized toolset offered by an external provider, that offers a common development environment through a Software-as-a-Service approach. A local support team within the company assists business units in deploying and using these tools, for instance by providing training.

The PFSPD project benefited greatly from the common set of tools that were used early on in the project. This proved particularly useful when one of the core team members moved to a different division based in the USA. Having the necessary infrastructure already available for distributed development, development continued seamlessly after this move, and with the newly introduced time difference, development could 'follow the Sun' during bursts of development.

People and Management

Coordination and Meritocratic Leadership

Inner source projects need a more flexible approach to coordination and leadership compared to conventional approaches that are based on organizational hierarchies and roles. One of the key tenets of a 'bazaar-style' approach is meritocracy,[8] in which developers who contribute significantly to certain parts of the code and thus have deep expertise, can become coordinators, or 'trusted lieutenants,' assisting the 'benevolent dictator' in managing and coordinating the project. Coordination within 'bazaars' is based on self-organization,[8] where developers self-select the tasks to work on, whether these are code contributions, defect fixes or bug reports and documentation. Whereas many open source projects typically don't have a commercial aim (though companies are increasingly involved in development of some of the successful open source projects), it may be important to formalize a number of roles within the core team that has the responsibility for managing an inner source project.[15] Which roles are needed will depend on the context of the organization, and they may emerge as needed—at Lucent, for instance, besides a benevolent dictator a 'community liaison' interacted with the various business units that were using the inner source project.

The PFSPD 'core team' consisted of four key members, but other developers contributed as well. The developer community of the project was very limited in size, with a slightly bigger community of several tens of users.

Transparency

Inner source is derived from the open source phenomenon, where product development is community-based and the process is transparent and generally open to anyone who wishes to be involved. Transparency is therefore another key factor to consider, but this may not always be straightforward. Developers and managers may not be comfortable with sharing code and development responsibility.[5] However, providing full access to all development artifacts through supporting infrastructure such as a source code repository and a wiki for sharing knowledge is important, as are a mailing list and an

Digital Object Indentifier 10.1109/MS.2014.77 0740-7459/$26.00 2014 IEEE

IRC server for asynchronous and synchronous communication, respectively. Without this transparency, developers wouldn't be able to 'lurk' and have no means to contribute to the project, which would greatly inhibit the inner source initiative.

At Philips Healthcare, the source code for the SPL platform is accessible to all developers (stored in a central version control system as described above). Furthermore, the development process has become much more transparent with a wiki server to facilitate organization-wide knowledge sharing, and a mailing list through which developers can communicate and ask questions, much like how developers in open source communities interact. In fact, the initial helpdesk that Philips had set up internally was found to be obsolete, because the mailing list was so successful in providing a Q&A forum.

Transparency also played a role in the PFSPD project. Initially, access to the source code repository was limited to the core developers, but it became an inner source project once the source code was migrated to an internal SourceForge installation. Two separate mailing lists provided the main channels for communication: a developer list to discuss daily development and progress, and a user list to support the user community. New releases were downloaded several tens of times.

Management Support and Motivated Individuals

Perhaps one of the most important factors to starting successful inner source initiatives is that of top-level management support. Inner source projects often start as 'grassroots' initiatives by key individuals within the organization who strongly believe that their organization can benefit from adopting open source practices. However, work on inner source projects can be hard to justify because it doesn't fit neatly within a strategic planning roadmap. Consequently, managers may consider any resources spent on such seemingly informal development programs to be wasteful and inefficient. However, once they realize the benefits that inner source offers, management can become supportive.[8] It is important that this support is complemented with a budget and resources.

Philips Healthcare's management have made a strong commitment to their inner source model. While it started out as a grassroots initiative, benefits were soon demonstrated and there is now support and advocacy from top-level management. While this support is important, it's not enough to merely impose inner source.[4] Developers must be motivated to voluntarily participate in this model to make it successful. At the individual developer level, having fun is an important intrinsic motivation widely found in open source communities. Gurbani and colleagues described how this led some developers to work on the SIP stack at Lucent in their spare time.[6]

At a higher organizational level, business units within Philips are encouraged to use, and actively engage in, the inner source project. Wesselius described how Philips Healthcare experimented with different business models to stimulate and encourage business units to contribute and engage in the development model.[4]

The PFSPD project was, as briefly mentioned, never an officially funded project assigned to a specific team. The collaboration between contributors emerged from their need to work on the software as that helped them in collaborating. However, due to the novelty of this collaboration model, for some contributors it was hard to justify their time spent on the project as management didn't fully understand or appreciate their efforts, or the nature of the inner source initiative. For others, time spent on this project could be justified as a 'technology transfer' activity. The general appreciation from the project's users further kindled developers' motivation to sustain work on the project.

Whether or not developers will get involved in an inner source project also depends strongly on the corporate culture and norms within an organization. Some organizations have a highly traditional culture with much emphasis on conformance with rules and guidelines, whereas others have a more liberal culture where more proactive taking initiative is encouraged.

Conclusion

Inner source is an emerging topic with increasing interest from practitioners who look to replicate the success of well-known open source projects. In this article we've illustrated nine key factors that we derived from our inner source research, which provides a useful 'framework' to analyze, compare and implement inner source initiatives. Based on the results of these assessments, we compiled a list of steps to get started with inner source (see Table 1). The table also outlines some risks associated with each factor. While we believe these are useful steps to follow, it's important to realize that merely following these steps doesn't necessarily result in a successful inner source initiative. Other factors may be at play that can affect the success such a program can achieve, and more research is needed to better understand what "makes or breaks" an inner source initiative. Furthermore, defining what makes an inner source initiative 'successful' depends on the context. In all three cases presented, inner source offers mechanisms that empower developers. For instance, business units within Philips Healthcare are empowered by their ability to make local changes to the shared platform shortly before a release; Lucent developers were empowered to improve the code, and researchers at Philips Research were able to collaborate on their video algorithm research.

Based on our recent interactions with a number of companies in several countries, we believe inner source

Digital Object Indentifier 10.1109/MS.2014.77 0740-7459/$26.00 2014 IEEE

is gaining significant traction in industry. We join Wesselius[4] in encouraging readers to share their experiences, challenges and lessons learned in setting up inner source initiatives.

References

1. Ayala, C., Cruzes, D. S., Hauge, Ø. and Conradi, R., Five Facts on the Adoption of Open Source Software. *IEEE Software* 2011**,** *28* (2), 95-99. DOI: 10.1109/MS.2011.32
2. Stol, K., Avgeriou, P., Babar, M. A., Lucas, Y. and Fitzgerald, B., Key Factors for Adopting Inner Source. *ACM Transactions on Software Engineering and Methodology* 2014**,** 23 (2). DOI: 10.1145/2533685
3. Raymond, E. S., *The Cathedral and the Bazaar*. O'Reilly: 2001.
4. Wesselius, J., The Bazaar inside the Cathedral: Business Models for Internal Markets. *IEEE Software* 2008**,** *25* (3), 60-66. DOI: 10.1109/MS.2008.79
5. Dinkelacker, J., Garg, P. K., Miller, R. and Nelson, D., Progressive Open Source. In *24th International Conference on Software Engineering*, 2002; pp 177-184. DOI: 10.1145/581339.581363
6. Gurbani, V. K., Garvert, A. and Herbsleb, J. D., A Case Study of a Corporate Open Source Development Model. In *International Conference on Software Engineering*, 2006. DOI: 10.1145/1134285.1134352
7. Vitharana, P., King, J. and Chapman, H. S., Impact of Internal Open Source Development on Reuse: Participatory Reuse in Action. *Journal of Management Information Systems* 2010**,** *27* (2), 277-304. DOI: 10.2753/MIS0742-1222270209
8. Riehle, D., Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi, Y., Naveh, B. and Odenwald, T., Open Collaboration within Corporations Using Software Forges. *IEEE Software* 2009**,** *26* (2), 52-58. DOI: 10.1109/MS.2009.44
9. Fitzgerald, B., Stol, K., O'Sullivan, R. and O'Brien, D., Scaling Agile Methods to Regulated Environments: An Industry Case Study. In *International Conference on Software Engineering*, 2013, pp. 863-872. DOI: 10.1109/ICSE.2013.6606635
10. Copeland, P. and Savoia, A., Entrepreneurial Innovation at Google. *IEEE Computer* 2011**,** *44* (4), 56-61. DOI: 10.1109/MC.2011.62
11. Morgan, L., Feller, J. and Finnegan, P., Exploring Inner Source as a Form of Intra-organisational Open Innovation. In *European Conference on Information Systems*, 2011. http://aisel.aisnet.org/ecis2011/151
12. van der Linden, F., Applying Open Source Software Principles in Product Lines. *Upgrade* 2009**,** *X* (2), 32-41.
13. Scacchi, W., Free and Open Source Development Practices in the Game Community. *IEEE Software* 2004**,** *21* (1), 59-66. DOI: 10.1109/MS.2004.1259221
14. Michlmayr, M. and Fitzgerald, B., Time-Based Release Management in Free and Open Source (FOSS) Projects. *International Journal of Open Source Software & Processes* 2012**,** *4* (1), 1-19. DOI: 10.4018/jossp.2012010101
15. Gurbani, V. K., Garvert, A. and Herbsleb, J. D., Managing a Corporate Open Source Software Asset. *Communications of the ACM* 2010**,** *53* (2), 155-159. DOI: 10.1145/1646353.1646392

Acknowledgements. We are grateful to all informants of our studies for their time and enthusiasm. This work was conducted as part of the ITEA2 SCALARE project, supported by Enterprise Ireland and by Science Foundation Ireland grant 10/CE/I1855 to Lero—the Irish Software Engineering Research Centre (www.lero.ie).

About the Authors

Dr. Klaas-Jan Stol is a research fellow at Lero—the Irish Software Engineering Research Centre, based at the University of Limerick. His research interests include contemporary software development methods such as open source software, inner source, crowdsourcing, and agile and lean methods. Contact him at klaas-jan.stol@lero.ie.

Prof. Brian Fitzgerald is Chief Scientist at Lero—the Irish Software Engineering Research Centre. He holds an endowed chair, the Frederick Krehbiel Chair in Innovation in Business and Technology at the University of Limerick. His research interests include open source software, crowdsourcing, and lean and agile methods. Contact him at bf@lero.ie.

Table 1. Summary of the three examples of inner source and recommendations to start an inner source initiative.

	Factor	Observations from the three cases	Recommendations	Potential Risks
Product	**Seed product**	SIP stack, DICOM and PFSPD file format all standards; internally developed implementations offering common functionality.	Identify seed projects that implement common functionality or an internally implemented standard.	Starting an inner source project with too many requirements up-front may not attract sufficient developers to deliver the project soon enough.
	Stakeholders	A variety of stakeholders helped in all three cases, as different expertise benefited the project. Additional features and improved algorithm implementations were contributed.	Identify a seed project that has different stakeholders to create a sufficiently large community of users and developers who all have an interest in the project.	Similar to conflicts that can arise among developers in an OSS project, differentces of opinion may negatively affect a project. A large number of stakeholders may lead to conflicting requirements, possibly leading to conflicts at the 'personal' level.
	Modularity	Modularity played an important role in all cases; the SIP stack was refactored, the medical platform was configurable and the functionality offered in PFSPD was divided into different tools.	Pay extra attention to the seed project's modularity and interface so as to facilitate reuse, expert contributions and long-term compatibility.	Modularity and "packaging" of a component should be carefully considered; product should not be based on too many constraining assumptions about its runtime environment that would decrease its independence or modularity.
Practices & Tools	**Development practices**	Open source or hybrid approaches were adopted. All cases exhibited approaches that facilitated flexibility to contributors.	Facilitate a development process that encourages contributions and also considers the constraints of the project and organization.	There is a risk that developers find it hard or uninteresting to contribute due to complexity of the project.
	Quality assurance	Peer review of contributions and responding promptly to user-reported problems.	Ensure quick turn-around of peer-reviewing contributions, inspection of problem reports, and supporting early-adopting users, so as to resolve problems quickly.	Contributions should be thoroughly reviewed so as to ensure that the product quality is maintained.
	Tools	In all cases the use of a common or compatible set of tools proved to be very important.	Ensure that appropriate and common tool support is available, without people having to learn new tools.	Converting to a new toolset may affect productivity if developers are not familiar with them.
People & Management	**Coordination & Leadership**	Coordination varied from traditional governance forms to more self-organizing forms in the smaller projects.	Recognize project creator's ownership. As project's importance and community grows, aim for coordination and management structures that maintain contributors' interest.	Conflicts may arise about who should lead a project.
	Transparency	In all cases the inner source project's source code was available, as was one or more mailing lists (for users and developers). Sometimes a wiki was used for knowledge sharing.	Provide full access to source code to encourage code reviews and suggestions for improvement. Provide a wiki installation for knowledge sharing, and a mailing list for internal communication that can be archived.	Developers and managers may not be comfortable with sharing code, either due to a feeling of losing control, or out of fear of personal assessment based on contributions.
	Management support & motivation	High-level management support and advocacy proved essential for inner source success. Inner source offered empowerment to contributors as it allows them to fix local problems. In some cases some developers contributed in their spare time.	Advocate and lobby with management by showing benefits of project for organization. It is important to sustain the 'flexible' nature of project to ensure contributors retain interest. Foster empowerment to users by letting them to solve their own problems.	If management make strategic changes, for instance, withdrawing support to developers to work on projects they believe are important, developer motivation is likely to drop, jeopardizing an inner source initiative's sustainability.

Digital Object Indentifier 10.1109/MS.2014.77 0740-7459/$26.00 2014 IEEE

Author Contact Information

Dr Klaas-Jan Stol
Lero—the Irish Software Engineering Research Centre
Tierney Building
University of Limerick
Limerick, IRELAND
Tel. +353 87 666 7729

Prof. Brian Fitzgerald
Lero—the Irish Software Engineering Research Centre
Tierney Building
University of Limerick
Limerick, IRELAND
Tel. +353 87 969 1492

Digital Object Indentifier 10.1109/MS.2014.77 0740-7459/$26.00 2014 IEEE