# 1Sample Pattern 1

## Title

Manifesto for Change

## Patlet

A culture of local optimization and silo-thinking can impede the adoption of Innerource. This can be overcome by the creation of a formal document such as a manifesto, a community charter, or an Open-Source Contribution Policy.

## Problem

Organizational policies (KPI's and incentive structures) and processes create a culture of local optimization where mid-level managers and teams are disincentivized from information sharing and collaboration.

## Context

- Large organizations with multiple autonomous Business Units (BU's) lead to the formation of "\_baronies\_" in middle and lower management.

- The incentive and KPI measures in place do not consider InnerSource contributions by teams and this leads to silo-thinking, which in turn prevents widespread adoption of IS across organization.

## Forces

- The existence of multiple BU's complicates the creation of collaborative networks across an organization. The existence of multiple BU's led to decentralized software development and teams working towards their local goals.

- A key cause that hinders collaboration and sharing of code is a sense of "\_Not invented here\_" amongst teams.

- Lack of trust between teams over the maintenance of a shared software component which can be critical to one team while non-critical to another (P07).

- Middle managers (Technical owners and Project managers) face pressure to keep their teams happy while simultaneously meet deadlines (P04).

- Lack of awareness of costs associated with a bespoke solution such as licensing, onboarding, training, and maintaining can cause teams to favor local optimization (P10).

- Hours spent on development need to be booked with existing projects under different BU's. This leads to ambiguity around booking hours spent on IS projects (P11).

## Solution

The creation of a formal document such as: i) a manifesto (P14); ii) InnerSource principles based on the InnerSource manifesto from IS commons (P10); iii) a community charter (P07); iv) An Open Source Contribution Policy (P16); or v) InnerSource Maturity model (P10), will provide a sound theoretical basis to overcome silo-thinking.

This document should: a) Outline the goals; and b) Mandate strategies and incentives to promote IS. The Manifesto is the "\_License to Contribute\_" (P14). Prior to the manifesto creation groundwork is done to update rules, processes, finance requirements, time allocation, guidelines, and trainings (P11). This means that once the manifesto is adopted it is not simply a document but becomes a foundation for InnerSource. Another benefit is that the manifesto acts as an operational safety net for contributors working on projects outside their assigned area (P01).

## Resulting Context

- Making it a company-wide strategy will help to communicate and advocate InnerSource to developers and managers (P08).

- A manifesto can help to empower developers to use OS (P14).

\*\*Limitation/Blockers\*\*

- In organizations with multiple product lines, success can vary depending on type of application across products (P20).

- The creation and adoption of a manifesto needs many rounds of communication with enterprise level management (P08).

- This pattern cannot work in a standalone fashion and has to be implemented in conjunction with other patterns for efficacy (P01,P04,P07, P08,P11, P17,P23, and P25).

## Known Instances (optional)

This pattern has been identified by four (P10, P11, P20, P08) panelists to have been implemented in their organizations and has seen limited use in the case of three panelists (P01, P23, P07).

## Alias

Change manifesto (S1)

# 2Sample Pattern 2

## Title

Create a collaborative environment (S2)

## Patlet

Organizational policies and politics create a culture of local optimization which inhibits collaboration. This is resolved by the creation of stakeholder boards, guilds, developer exchanges, cross-team pairing, collaborative decision making via RFCs, and knowledge sharing forums.

## Problem

Organizational policies (KPI's and incentive structures) and politics create a culture of local optimization where mid-level managers and teams focus primarily on achieving their own objectives; this means that information sharing and collaboration are not a priority for these organizational units, or this could even be perceived as being at conflict with the team's objectives.

## Context

- Large organizations with multiple autonomous Business Units (BU's) lead to the formation of "\_baronies\_" in middle and lower management.

- The incentive and KPI measures in place do not consider InnerSource contributions by teams and this leads to silo-thinking which in turn prevents widespread adoption of IS across organization.

- A highly politicized and top-down environment could also discourage people from taking the initiative to explore InnerSource.

## Forces

- Another key cause that hinders collaboration and sharing of code is a sense of "\_didn't build it here\_" amongst teams.

- Lack of trust between teams over the maintenance of a shared solution which can be critical to one team while non-critical to another (P07).

- Middle managers (Technical owners and Project managers) face pressure to keep their teams happy while simultaneously meet deadlines (P04).

- Ignorance of costs associated with a bespoke solution such as licensing, onboarding, training, and maintaining can cause teams to favor local optimization (P10).

- Hours spent on development needed to be booked with existing projects under different BU's. This led to ambiguity around booking hours spent on IS projects (P11).

## Solutions

- The creation of an ISPO/OSPO/Technical Committee (P01, P04, P08, P11, P23) which acts as a support mechanism and provides representation to stakeholders such as developers, agile coaches, engineers, and senior management. The ISPO/OSPO can work in a decentralized fashion by providing company-wide guidance but allowing teams to lead the InnerSource implementation in their area (P04). The ISPO/OSPO has specialized InnerSource roles such as advocates, community managers to enhance collaboration (P11).

- Modifying and consolidating code management tools (e.g., GitHub) to facilitate collaboration between teams (P08, P23).

- Creating dedicated communication channels such as in MS Teams channels, digital events, webpages in intranet where people can get support and exchange knowledge (P14).

- Face-to-face conference of stakeholder representatives to plan InnerSource implementation and sustenance (P10).

## Resulting Context

- The combination of the above solutions will result in increased collaboration and a shared understanding of InnerSource across the organization.

- Praise and encouragement by the ISPO/OSPO/Committee will motivate contributors.

- Collaborative code management tools will enable developers to InnerSource their code and contribute with ease.

\*\*Limitation/Blockers\*\*

- Extensive communication must be done with senior management to bring different parts of the organization, such as the developers and technical committee, on board (P08, P11).

- Lack of a specialized role who can train developers in InnerSource (P11).

## Known Instances (optional)

This pattern has been identified by seven (P14, P23,P01,P20,P08,P10,P04) panelists to have been implemented in their organizations.

# 3Sample Pattern 3

## Title

Involve teams in IS implementation (S3)

## Patlet

Organizational policies (KPI's and incentive structures) disincentivize people from contributing to InnerSource. Involving key teams such as security, engineering, and architecture in the early phase of implementation can prevent blockers appearing at a later stage and mitigate the problem of local optimization.

## Problem

Organizational policies (KPI's and incentive structures) and politics create a culture of local optimization where mid-level managers and teams are disincentivized from information sharing and collaboration.

## Context

- Large organization's with multiple autonomous Business Units (BU's) lead to the formation of "\_baronies\_" in middle and lower management.

- The incentive and KPI measures in place do not consider InnerSource contributions by teams and this leads to a silo-thinking which in turn prevents widespread adoption of IS across organization.

## Forces

- The existence of multiple BU's complicates creation of collaborative networks across organization.

- Another key cause that hinders collaboration and sharing of code is a sense of "\_didn't build it here\_" amongst teams.

- Lack of trust between teams over the maintenance of a shared solution which can be critical to one team while non-critical to another (P07).

- Middle managers (Technical owners and Project managers) face pressure to keep their teams happy while simultaneously meet deadlines (P04).

- Ignorance of costs associated with a bespoke solution such as licensing, onboarding, training, and maintaining can cause teams to favor local optimization (P10).

- Hours spent on development needed to be booked with existing projects under different BU's. This led to ambiguity around booking hours spent on IS projects (P11).

## Solutions

- Involvement of diverse stakeholders can be ensured by the InnerSource team working closely with security, HR, compliance, engineering, and internal customers (such as R&D) to influence the general approach and create an additional avenue for InnerSource (P01, P04, P08, P14, P18, and P23).

## Resulting Context

- InnerSource becomes a natural part of the development process and empowers teams to use InnerSource as a tool where it is beneficial.

- In P04's organization this led the central engineering system team to extensively leverage InnerSource for all tooling and made them consider InnerSource as a scenario in the processes as well (P04).

- Helps to find common solutions to common problems from the teams instead of having fragmented ad-hoc solutions (P23).

## Limitation/Blockers

- Steep learning curve leading to build-up of frustration in the early phases and a mismatch between expectations and reality (P04).

- Strategy will work for a small organization not a large one with multiple business units (P11).

- Extra communication efforts needed to organize people from different teams (P08).

- If adoption is bottom-up IS will not scale beyond the first one or two layers of the organisational structure (P20).

\*\*Known Instances (optional) \*\*

This pattern has been identified by eleven (P01, P03,P04,P08,P09,P10,P14,P18,P20,P23, and P25) panellists to have been implemented in their organisations.

# 4Sample Pattern 4

## Title

Elicit support from senior management (S4)

## Patlet

Organizational policies (KPI's and incentive structures) disincentivize people from contributing to InnerSource. Top management support can help overcome established organizational culture which promotes local optimization.

## Problem

Organizational policies (KPI's and incentive structures) and politics create a culture of local optimization where mid-level managers and teams are disincentivized from information sharing and collaboration.

\*\*Context\*\*

- Large organization's with multiple autonomous Business Units (BU's) lead to the formation of "\_baronies\_" in middle and lower management.

- The incentive and KPI measures in place do not consider InnerSource contributions by teams and this leads to a silo-thinking which in turn prevents widespread adoption of IS across organization.

\*\*Forces\*\*

- The existence of multiple BU's complicates creation of collaborative networks across organization.

- Another key cause that hinders collaboration and sharing of code is a sense of "\_didn't build it here\_" amongst teams.

- Lack of trust between teams over the maintenance of a shared solution which can be critical to one team while non-critical to another (P07).

- Middle managers (Technical owners and Project managers) face pressure to keep their teams happy while simultaneously meet deadlines (P04).

- Ignorance of costs associated with a bespoke solution such as licensing, onboarding, training, and maintaining can cause teams to favor local optimization (P10).

- Hours spent on development needed to be booked with existing projects under different BU's. This led to ambiguity around booking hours spent on IS projects (P11).

\*\*Solutions\*\*

- Senior management declares IS to be the norm not the exception (P23).

- Senior management from the Vice-President level include IS in their presentations on company strategy (P17, P19).

- Strong messaging by COO & CTO's via emails and companywide events on the importance of InnerSource (P08).

- Allocating funds for a mutual collaborative environment (e.g., GitHub).

- Sponsorship by a change management committee such as a Digital Transformation Program (P07).

- Setting up a dedicated program which enables 10% of engineering time allocated to InnerSource (P15).

- Get senior management to greenlight the manifesto before publishing it. This is also a solution to Sample Pattern 1.

\*\*Resulting Context\*\*

- Senior management support will convince fence-sitters to take the leap (P03).

\*\*Limitation/Blockers\*\*

- Difficult to get senior managers at the COO and CTO level who recognize the value of InnerSource (P08).

- Without grassroots participation the top-down implementation will fail (P10).

- The size and type of organization can mean many software development initiatives competing for senior management attention (P23).

- In a hierarchical organization with multiple layers, InnerSource must be promoted by a committed and motivated person for senior management to take notice (P11).

- Absence of metrics to measure utilization of time allocated for InnerSource (P15).

- Difficulty in searching for code and aversion to sharing code can nullify senior management support (P01).

\*\*Known Instances (optional)\*\*

This pattern has been identified by six panelists (P01, P07, P08, P11,P15,P17) to have been implemented in their organizations.

# 5Sample Pattern 5

\*\*Title\*\*

Create Unique InnerSource License (S5)

\*\*Patlet\*\*

The legal ambiguity around Intellectual Property (IP) can prevent many business units from participating in InnerSource. The creation of customized internal InnerSource license with validation from the legal department and which accounts for diversity in intra-group trading rules can address issues related to distribution, use, and modification rights of shared source code.

\*\*Problem\*\*

With organizations that are globally distributed, the sharing of source code across national boundaries raises questions of distribution, use, and modification rights. The legal ambiguity around IP ownership, code use rights, and patents can prevent many business units from participating in InnerSource.

\*\*Context\*\*

- For organization's with multiple BU's, in the scenario of a BU being sold, valuable Intellectual Property (IP) can be lost in the absence of a licensing arrangement (P18).

- On the other hand, a very restrictive IS license which prohibits publishing of IS software as open source can discourage projects from adopting IS.

\*\*Forces\*\*

- Without a license in place, sharing between BU's is inhibited by legal requirements.

- If projects find the license too restrictive, they will often stay with proprietary software and not opt for InnerSource (P07).

\*\*Solutions\*\*

- A gatekeeping arrangement is adopted where the project registers on a company platform and enters an "\_incubation phase\_" where it is first checked for the IS license and compliance with export control regulations (P11).

- Involving legal teams or export control to create tailor made licenses for the individual cases (P03, P11, P18).

- The ISPO establishes an allow and deny list for the open-source licenses to be used in IS projects, as IS projects may cross companies' boundaries and country boundaries (P17).

\*\*Resulting Context\*\*

- Senior management support will convince fence-sitters to take the leap (P03).

- License done with and publicly approved by the legal department is publicized on internal social network (e.g., Yammer). This results in greater participation by BU's as the IP rules are clearly defined.

\*\*Limitation/Blockers\*\*

- There should be a clear distinction between IS and open-source SMEs, leaders, policies and processes and goals and objectives (P25).

- A proprietary IS license done by legal can be confusing to the engineers who are used to simpler Open Source licenses (P11).

\*\*Known Instances (optional)\*\*

This pattern has been identified by four panelists (P11, P14, P17, P23) to have been implemented in their organizations.

# 6Sample Pattern 6

\*\*Title\*\*

Collaborate with Subject Matter Experts (SME's) (S34)

\*\*Patlet\*\*

The legal ambiguity around the sharing of source code across national boundaries can prevent business units from participating in InnerSource. Collaborating in InnerSource policy development with SMEs from departments such as tax, intellectual property, export control and legal, will help creation of appropriate Intellectual Property (IP) policies and meet compliance requirements.

\*\*Problem\*\*

With organizations that are globally distributed, the sharing of source code across national boundaries raises questions of distribution, use, and modification rights. The legal ambiguity around IP ownership, regulatory compliance, code use rights, and patents can prevent many business units from participating in InnerSource.

\*\*Context\*\*

- Software engineers do not have the expertise to draft appropriate Intellectual Property (IP) policies and understand compliance requirements.

- Collaboration in InnerSource projects is, to a large degree, cross-divisional and international (P09).

- In sectors with strong regulatory requirements, it is very important to integrate compliance in processes around InnerSource and Open Source (P14).

\*\*Forces\*\*

- The InnerSource program can face objections from SME's if it does not meet legal and compliance requirements (P11).

- If business units find it difficult to ensure compliance across cross-divisional and multi-region projects, they can be tempted to limit collaboration between teams.

- If the maintenance team is working on software which is subject to stringent requirements around quality management and traceability, this might prevent contributions from other teams in the organization (P04, P23).

\*\*Solutions\*\*

- Develop a compliance training for software developers where several of these challenges are explained in practical terms (P23).

- Collaborate with SME's to develop policies and reach consensus on compliance issues (P11, P17, P23).

\*\*Resulting Context\*\*

- Collaboration leads to the creation of appropriate governance, legal and regulatory processes (P09, P11, P17).

\*\*Limitation/Blockers\*\*

- It is difficult to find the right people from the legal and regulatory departments who are knowledgeable about InnerSource (P10, P18).

- Availability of SMEs can be limited due to other priorities (P23).

- Due to involvement of people from different backgrounds it can be difficult to reach a consensus on issues (P14).

\*\*Known Instances (optional)\*\*

This pattern has been identified by seven panelists (P04, P09, P10, P11, P14, P17, P23) to have been implemented in their organizations.

# 7Sample Pattern 7

\*\*Title\*\*

Compliance attribution to specific product (S6)

\*\*Patlet\*\*

Software which needs to comply with regulatory requirements needs domain knowledge and attribution of compliance requirements to the product which might be outside the expertise of the contributing teams. Specific regulatory requirements can be complied with by attributing compliance to a specific product and associating with the repository used to build the product.

\*\*Problem\*\*

Regulatory or taxation requirements specific to the industry sector (e.g., finance, healthcare) require a refined understanding and application of the compliance requirements to products. If the maintenance team is working on software which is subject to stringent requirements around quality management and traceability, this might prevent contributions from other teams in the organization.

\*\*Context\*\*

- Collaboration in InnerSource projects is, to a large degree, cross-divisional and international (P09).

- In sectors with strong regulatory requirements, it is very important to integrate compliance in processes around InnerSource and Open Source (P14).

\*\*Forces\*\*

- If business units find it difficult to ensure compliance across cross-divisional and multi-region projects, they can be tempted to limit collaboration between teams.

- If the maintenance team is working on software which is subject to stringent requirements around quality management and traceability, this might prevent contributions from other teams in the organization (P04, P23).

\*\*Solutions\*\*

- The responsibility of complying with regulatory requirements is with the projects or components which use InnerSource-d software (P09, P11). In cases where there are different components, each component is responsible for complying with requirements.

\*\*Resulting Context\*\*

- Distributing compliance responsibilities at the project and component level ensures compliance and takes the burden off from contributors (P09).

\*\*Limitation/Blockers\*\*

- This pattern is not applicable when compliance is built into the organization's engineering processes which means all projects whether InnerSource-d or not need to comply with requirements (P01, P04).

\*\*Known Instances (optional)\*\*

This pattern has been identified by three panelists (P09, P11, P25) to have been implemented in their organizations.

# 8Sample Pattern 8

\*\*Title\*\*

Clear demarcation of responsibility to comply with requirements (S7)

\*\*Patlet\*\*

Software which needs to comply with regulatory requirements needs domain knowledge and can discourage contributions from other teams. In globally distributed organizations, compliance responsibilities can be distributed between the team who maintain the source code and the collaborators who could be using the code for specific product in their region.

\*\*Problem\*\*

Regulatory or taxation requirements specific to industry sector (e.g., finance, healthcare) require a refined understanding and application of the compliance requirements to products. If the maintenance team is working on software which is subject to stringent requirements around quality management and traceability, this might prevent contributions from other teams in the organization.

\*\*Context\*\*

- Collaboration in InnerSource projects is, to a large degree, cross-divisional and international (P09).

- In sectors with strong regulatory requirements, it is very important to integrate compliance in processes around InnerSource and Open Source (P14).

\*\*Forces\*\*

- If the maintenance team is working on software which is subject to stringent requirements around quality management and traceability, this might prevent contributions from other teams in the organization (P04, P23).

\*\*Solutions\*\*

- The contributing guidelines make it clear that responsibility for compliance is with the maintaining team and not with contributors from other parts of the organization (P01, P11, P23). This avoids reluctance of people to contribute to projects.

\*\*Resulting Context\*\*

- Distributing compliance responsibilities at the project and component level ensures compliance and takes the burden off from contributors (P09).

\*\*Limitation/Blockers\*\*

- This pattern is not applicable when compliance is built into the organization's engineering processes which means all projects whether InnerSource-d or do not need to comply with requirements (P01, P04).

\*\*Known Instances (optional)\*\*

This pattern has been identified by three panelists (P09, P11, P25) to have been implemented in their organizations.

# 9Sample Pattern 9

\*\*Title\*\*

Appoint dedicated IS champions (S8)

\*\*Patlet\*\*

InnerSource programs do not scale beyond grassroots initiatives due to a lack of motivated advocates. InnerSource can be promoted and sustained by dedicated champions or community managers who are responsible for promoting IS through the organization by different means such as blogging, talks, and conversations with teams.

\*\*Problem\*\*

For an InnerSource program to be successful and to scale beyond small grassroots initiatives there is a need for a dedicated manager who has a collaborative mindset, good communication skills, skillful marketing, and can take a strategic overview.

\*\*Context\*\*

- Organizational structures favor local optimization which leads to focus on team level priorities over multi-team collaboration (P01).

- Communication and promotion of InnerSource is a full-time role which cannot be performed by developers and engineers due to time constraints (P11). Part-time commitment results in reduced effectiveness due to conflicting priorities.

- Multiple existing grassroots initiatives need to be scaled-up for strategic benefits (P20).

- A broad communication strategy is needed which focuses on promoting a culture receptive to IS (P18).

- The lack of a cohesive communication plan in organizations can limit the visibility of InnerSource.

\*\*Forces\*\*

- Local optimization can manifest in large organizations with multiple BU's due to diversity of IS initiatives and lack of coordination between them.

- Lack of IS experts who understand the need for inviting, facilitating, and mentoring contributions (P10, P20, P23).

- Lack of IS experts who have a sound understanding of OS and IS concepts (P10).

- Organizations suffer from an overload of information and simply adding more information on IS will not help (P04).

\*\*Solutions\*\*

- Appointment of dedicated champions or community managers responsible for promoting IS through the organization by different means such as blogging, talks, conversations with teams etc. (P14, P17). The champions can be engineers who are part of the ISPO or a technical oversight board, who help with outreach and project reviews (P19).

- For large organizations a core team of champions representing the BU's needs to be constituted who passionate about IS and motivated by a desire to help it grow (P03, P07, P20, P25). The team can be organized as a steering committee with regular meetings and a clear mandate.

- The Champions have great social skills, good knowledge of compliance & regulatory issues, and are excellent communicators with the ability to motivate people (P11, P17).

- The Champions act as establishers of IS within the organization by mentoring teams, facilitating learning, encouraging retrospectives, bringing external knowledge into the organization (P10).

\*\*Resulting Context\*\*

- If the champions are recruited from within the organization, they can influence policy and effectively advocate for IS (P03).

- Increased transparency around process and a support system is created for developers looking to make the best use of IS (P14).

- The champion acts as a knowledge exchange interchange between the BU's by spreading benefits and best practices of IS (P14).

- As the champion is known and well-networked, they will become a single point of contact for InnerSource questions (P09).

\*\*Limitation/Blockers\*\*

- Appointment of a dedicated champion with enough seniority requires support from senior management (P03, P09, P18, P23).

- In a large organization, a team of champions will be needed to scale the initiative (P04).

- For the role to be effective it needs to be full-time, structured, distinct from developer roles, and the champion adequately trained in InnerSource (P03, P07, P09, P11).

- Senior management can lack understanding of the non-technical aspects of the champion role and there is no budgetary support for hiring and retention of appropriate personnel (P11).

- It is difficult to find a person with good IS and communication skills to fill the role and a replacement if the post falls vacant (P08, P13).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 15 panelists (P01, P03, P04, P07, P08, P09, P10, P11, P13, P14, P17, P18, P20, P23, and P25) to have been implemented in their organizations.

# 10Sample Pattern 10

\*\*Title\*\*

Create mechanism for secure access to shared code (S11)

\*\*Patlet\*\*

There is a conflict between security requirements of restricted access to code and InnerSource principles of open access. This conflict can be resolved by: a) Creation of a defined set of rules governing code confidentiality; b) Clear definitions of different sharing levels of repositories; and c) Use of automated tools to scan shared repositories for confidential information.

\*\*Problem\*\*

InnerSource implementation needs free sharing of code for collaboration whereas security concerns and policies require restricted access to the code.

\*\*Context\*\*

- The core benefit of InnerSource i.e., openness is lost if code is not readily accessible by developers (P07).

- Security policies must be considered to prevent leakage and unauthorised access to code (P08).

- In organizations where InnerSource is not the norm, contributions can be limited to projects which are perceived to be publicly accessible (P11).

\*\*Forces\*\*

- Existing tooling and repositories can have security constraints which inhibit code openness (P07).

- Strong security concerns around leakage of customer data, injection of malicious code, and industrial sabotage can prevent open code.

- Concerns that open code will expose flaws in the security framework (P01).

- Project tools such as JIRA are closed by default (P10).

\*\*Solutions\*\*

- Access can be secured by clearly defining the different sharing levels of repositories. Repositories can be classified as (P17, P23): a) Public - open source: accessible for all software developers in the world; b) Shared - inner source: accessible for all software developers in the organization; c) Internal – Accessible only to a legal entity within the organization such as a BU; and d) Closed – closed source: only accessible to named individuals in the organization.

- To prevent unauthorized access further measures can be taken such as: Single Sign-On (SSO), 2-Factor Authentication, and rotation of access tokens (P23). Sensitive repositories can be closed by default but with metadata available via an IS portal.

- Time limited read-only access can be granted to collaborators through an automated approval process (P04). This prevents the maintaining team from becoming bogged down in administration and simultaneously keeps the code secure.

- Delegation of access level decisions to projects (P17).

- A security agreement can be created governing shared repositories which prohibits storage of passwords, tokens, client data, usernames, business critical algorithms, and distribution of code from shared repositories (P23).

- Automated tools can be used to scan the repository for any stored confidential data (such as passwords, tokens etc), known vulnerabilities in dependencies and prevent commits with such data (P04, P17).

\*\*Resulting Context\*\*

- This minimizes the risk of maintaining teams accidently receiving confidential data in shared repositories (P04).

- Security requirements are met without reverting to closed code and the access levels prevent unintentional modifications.

\*\*Limitation/Blockers\*\*

- The restrictive security policies can be a symptom of deficiencies in the security framework in which scenario this solution will be inapplicable (P01).

- Restricting access to code is against InnerSource principles as all repositories should be open and it is better to build long term trust (P01, P25).

- It is difficult to scale the process of curating and classifying data based on security levels (P04).

- Access mechanism can lead to developer dissatisfaction and impact discoverability (P04).

- The development and maintenance of automated tools presents an additional overhead (P08).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 6 panelists (P03, P04,P08, P11, P17, and P23) to have been implemented in their organizations.

# 11Sample Pattern 11

\*\*Title\*\*

Involve security team in IS implementation (S14)

\*\*Patlet\*\*

The concerns of the security team regarding security of open repositories and lack of clarity regarding security policies can restrict access to code. This can be mitigated by creating awareness of the secure access guidelines by means of training sessions for contributors. Additionally, the lack of clarity in security policies around access and sharing levels can be resolved by involving the security team in the InnerSource implementation phase (C8).

\*\*Problem\*\*

InnerSource implementation needs free sharing of code for collaboration whereas security concerns around code leakage and confidentiality require restricted access to the code. Also, a lack of clarity in security policies around access and sharing levels can be a security challenge.

\*\*Context\*\*

- The core benefit of InnerSource i.e., openness is lost if code is not readily accessible by developers (P07). Additionally, a Single Sign On (SSO) type system needs to be in place for ease of access.

- Operational safety can be compromised if deficiencies exist in the security framework (P01).

- InnerSource projects need to be complaint with security to prevent leakage and conform with the engineering best practices of the organization (P04, P08).

- Due to lack of clear rules around repository access, visibility, and creation, there exist a multitude of visibility and access settings for repositories (P09).

\*\*Forces\*\*

- Strong security concerns around leakage of customer data, injection of malicious code, and industrial sabotage can prevent open code.

- Inherited security constraints from existing projects make it difficult to reuse existing tools repositories (P07).

- Lack of awareness around security issues can lead to contributors inadvertently leaving confidential data in shared repositories.

\*\*Solutions\*\*

- Awareness of the secure access guidelines can be created by means of dedicated training sessions (P11, P23).

- Another technique is to require developers to specify access (either shared or closed) every time they create a new repository.

- An appropriate security policy can be created by involving the security team in the IS implementation phase. This can be done via the creation of an IS task force with representatives from Security, Legal, Engineering, and the ISPO (P04, P23).

\*\*Resulting Context\*\*

- This minimizes the risk of maintaining teams accidently receiving confidential data in shared repositories (P04).

- Developers become comfortable with creating InnerSource repositories which comply with security requirements (P08).

- This will reduce security risks as there is clarity around security policies and enable creation of secure repositories (P04, P08).

\*\*Limitation/Blockers\*\*

- Development of a clear and consistent security policy needs communication with the security team and security experts with open-source experience (P08).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 5 panelists (P04, P08, P11, P23, and P25) to have been implemented in their organizations.

# 12Sample Pattern 12

\*\*Title\*\*

Create a single source of information (S9)

\*\*Patlet\*\*

A key challenge is the lack of discoverability of InnerSource projects in organizations. The discoverability of IS projects in an organisation can be enhanced by creating a single source of information which can include: a) A searchable catalogue of inner-source assets; b) An IS portal using software such as SAP; c) Adapting an existing portal and adding InnerSource labelled projects; and d) Embedding the portal into the IS on-boarding process.

\*\*Problem\*\*

Lack of discoverability can hinder efforts by advocates to elicit greater contribution in InnerSource projects. Awareness of IS within the company is limited due to the absence of a single portal for InnerSource project information.

\*\*Context\*\*

- Despite best efforts awareness about IS projects is limited within the company.

- Implementation of stop-gap solutions such as an IS catalogue does not resolve the problem of discoverability.

\*\*Forces\*\*

- A false dichotomy exists between IS and assigning teams to work on code.

- Searching for IS projects on GitHub is not a feasible solution.

\*\*Solutions\*\*

- Create a single source information such as a Wiki or an IS portal (P14), a searchable catalog of inner-source assets (P07, P19), or adapt an existing portal and add InnerSource labelled projects to it with the ability to filter with a dedicated InnerSource label (P17).

- Adopt a seeding strategy by implementing a software forge such as GitHub on an experimental basis (P10, P25). Once developers experience the utility of this, they will drive transition to the central software forge.

\*\*Resulting Context\*\*

- Information sharing becomes convenient due to all documentation being in one place (P11).

- Enhances visibility and enables easy discovery of IS projects by contributors (P08, P17).

\*\*Limitation/Blockers\*\*

- For desired effect this needs to be combined with a 12/18-month communications plan targeting leadership, developers, and all functional stakeholders (P25).

- The portals, Wikis should be visually attractive to engage prospective contributors (P11).

- Engineers need awareness that their responsibility does not end with putting code in a central repository and that they are serving other engineers (P01).

- The content of the portal needs to be curated and should have value added bits such as newsletters and key statistics (P17).

- Developers need to ingrain the habit of searching for code on the portal prior to writing code (P13).

- A portal can also suffer from a discoverability problem in larger organizations with many developer resources and tools (P04).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 9 panellists (P08, P11, P13, P14, P15, P17, P20, P23,and P25) to have been implemented in their organizations.

# 13Sample Pattern 13

\*\*Title\*\*

Implement community building program (S21)

\*\*Patlet\*\*

Awareness of InnerSource is limited to few individuals within the organization and there is a lack of shared understanding of InnerSource principles in teams adopting InnerSource. Awareness and shared understanding of IS practices can be developed via communities of practice where developers across different silos collaborate.

\*\*Problem\*\*

Awareness of the benefits and principles of InnerSource with limited to a few committed people in the engineering team. The intangible nature of InnerSource leads to varying interpretations and difficulty in ensuring consistency when scaling beyond one or two teams who are familiar with IS principles.

\*\*Context\*\*

- Multiple existing grassroots initiatives need to be scaled-up for strategic benefits (P20).

- Awareness about IS benefits can remain limited if most stakeholders have traditional infrastructure backgrounds (P09, P18).

- Lack of awareness can lead people to make assumptions about IS which might not be true (P01).

- Multiple and sometimes wrong interpretations can be solidified in the form of corporate policy documents, leading to a lot of confusion, misconceptions, and unwarranted reservation towards InnerSource (P09).

\*\*Forces\*\*

- InnerSource is often limited to people having a previous external experience in an Open-Source community (P17).

- If the InnerSource implementation is limited to projects that are deemed business priority, it can lead to teams starting up their own InnerSource initiative.

- People see it as a fashionable trend (for e.g. similar to Agile) and lack a deeper understanding of InnerSource (P01).

\*\*Solutions\*\*

- The IS initiative can be implemented via communities of practice based upon the developer's specialization such as architects, engineers, API, Security/DevSecOps, Quality etc. This also helps in encouraging knowledge sharing and code reuse (P23).

- Existing frameworks such as the "Open Serious Community Framework" can be used to generate community engagement (P07).

- Community building can be done by establishing Internal community forums for engineers with weekly meetings to share projects, ideas, and questions (P01).

- Establish a formal training program using material from the InnerSource Commons and applying the IS maturity model to identify gaps in IS implementation (P10).

\*\*Resulting Context\*\*

- Key benefits are increased communication (P03), more efficient work due to de-siloing (P08), greater discoverability (P08), and increased trust amongst people (P01).

\*\*Limitation/Blockers\*\*

- Excessive segmentation by specialization can have unintended consequence of reinforcing silo effect (P25).

- The creation of a trust environment is only the first step towards meaningful code collaboration (P01).

- This strategy needs a strong leader that can overcome business boundary issues, include egos, encourages collaborators, and provides clarity (P04, P08).

- The communities can get side-tracked into the promotion of other initiatives to the detriment of IS (P03).

- Differing implementations and viewpoints enhance IS as it is a behaviour to be ingrained (P04).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 10 panelists (P01, P03, P04, P08,P10,P11,P13, P15, P17, P23) to have been implemented in their organizations.

# 14Sample Pattern 14

\*\*Title\*\*

Create a unified code repository (S15)

\*\*Patlet\*\*

Existing technology infrastructure can pose problems with code transparency and the interconnections with different elements can further complicate this scenario. Creation of a unified source code inventory which collects IS and legacy repositories in one place will help in preventing duplication and enhance transparency.

\*\*Problem\*\*

Technology framework covers infrastructure areas such as development environment, tool stacks, and platforms. One of the issues is the organization trying to adapt the existing tool stack to InnerSource. Another issue is a lack of transparency which makes it difficult to scan code bases for duplication, poor code, and technical debt. The dynamic nature of the infrastructure with a complex web of relationships between different elements (projects, repositories, products etc.) also causes issues.

\*\*Context\*\*

- Visibility of code assets is needed to prevent stale code and code duplication.

- Having a good API for the tool stack is essential to extract the relevant project information, do automation, and monitoring for IS projects (P11).

- Lack of transparency can create barriers where contributors are not able to see if their contributions have passed Continuous Integration (CI) (P20).

\*\*Forces\*\*

- Multiple business units each with their own platforms (e.g., GitLab, Azure DevOps) leads to fragmented tooling and problems in adopting IS (P17, P23).

\*\*Solutions\*\*

- Creation of a unified source code inventory which collects IS and legacy repositories in one place will help in preventing duplication and enhance transparency.

\*\*Resulting Context\*\*

- Code contribution and code scanning inside organization becomes simple with a unified repository (P03, P23).

- IS implementation becomes easier when compared to adapting IS to fragmented repositories (P08).

\*\*Limitation/Blockers\*\*

- In case of large organizations, the effort of identifying duplicate code and assigning to shared repositories outweighs the benefits (P01, P04,P09,P11).

- Discoverability of the new shared repository can be an issue (P04).

- Tool migration requires dedicated time, effort, and support (P23).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 5 panelists (P03, P08, P09, P17, P18, P23,and P25) to have been implemented in their organizations.

# 15Sample Pattern 15

\*\*Title\*\*

Run automated tests for code quality (S19)

\*\*Patlet\*\*

Running automated tests prior to contributions review by maintaining team will reduce load on the maintaining team, enhance code quality, and give the contributor a clear understanding of quality requirements.

\*\*Problem\*\*

The momentum of adding new features to code must be maintained to prevent code from going stale (i.e., going out of date). Additionally, developers assume that the code has value if it is being constantly updated, Otherwise, they risk losing interest in contributing.

\*\*Context\*\*

- Most developers lack time to contribute on a regular basis to keep code fresh(P09).

- Existence of a multitude of innovative projects driven by few committed people makes it difficult for contributors to prioritize with their limited time.

- Code is not updated due to lack of feature requests from product manager and no technical requirement from the developer team (P08).

\*\*Forces\*\*

- IS projects which are high use but low maintenance risk not being updated when changes are needed (P20).

\*\*Solutions\*\*

- Implementation of multiple code scanning processes (such as pre-commit hooks, unit tests etc) in the development pipeline irrespective of whether code contributor is from within the team or external (P01, P09).

- Integration of automated tools such as SonarQube, Jenkins, and Action Runners from GitHub into the platform (P09).

\*\*Resulting Context\*\*

- Results in a fast feedback loop, detection of security vulnerabilities, confidence in the stability and quality of code (P03, P08 P09, P17, and P20).

\*\*Limitation/Blockers\*\*

- A balance must be struck between adding new features and managing changes to the product. Constant addition of features can be viewed as more work by the client (P04).

- Project team needs to develop the automated testcase (P08).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 5 panelists (P03, P04, P08, P09, and P20) to have been implemented in their organizations.

# 16Sample Pattern 16

\*\*Title\*\*

Demonstrate benefits of IS by real examples (S22)

\*\*Patlet\*\*

A lack of knowledge about InnerSource can manifest in senior management not understanding the cultural transformation it entails. Senior management can be educated on the long-term nature and benefits of IS by demonstrating the productivity benefits by means of pilot projects. Achieving success in pilot IS projects can be a strong factor for senior management to scale-up IS. These success stories can be presented in a neutral fashion, at forums which involve senior management to bolster the case in a subtle manner.

\*\*Problem\*\*

InnerSource is perceived to be a technical issue rather than a cultural one. This cause manifests as different symptoms including IS being adopted on a piecemeal basis in the company and senior management exhibiting a preference for ready-made solutions. Another related issue is the rise of scepticism due to the inability to measure IS benefits as it is a long-term process with several intangibles such as behavioral changes in employees.

\*\*Context\*\*

- Established KPI's designed to measure quantitative performance data are inappropriate to measure InnerSource benefits. Senior management wants to understand the Return on Investment (ROI), which is difficult in the absence of tailored metrics.

- Senior management does not have an Open-Source background (P09).

- Senior management wants to kickstart IS at the technical level without understanding the long-term cultural transformation aspect (P25).

- Making IS the norm across the organization can result in shared code without the adoption of IS philosophy by the participating teams (P11).

- Future maintainability concerns and a low level of trust in IS projects leads to preference for third party products (P23).

\*\*Forces\*\*

- In large organizations a considerable number of senior managers come from a non-software development background. This means they are not aware of methodologies such as InnerSource and see it more as a tool to fix technical problems(P09).

\*\*Solutions\*\*

- In the initial phase rather than KPI's, focus is directed on measures which show the adoption of IS like the number of users, repositories, traffic, pull requests, breakdown per business unit (P23).

- Implementing pilot projects which are clearly defined within a SMART framework and align with the larger business objectives (P25). This raises the possibility of success and will help in sustaining the IS program (P08, P09). The successful projects can then be showcased as ideal case studies through the organisation and also give access to people that are practicing InnerSource and thus are seen as trusted advisors for their own InnerSource journey (P04).

- A regularly updated dashboard which displays IS project data and metrics on a daily basis (P17).

- Adopt KPI's from external knowledge sources which are relevant to IS such as change lead time, deployment frequency, change failure rate, knowledge transfer, and Mean Time to Recovery (MTTR) (P10, P17, P23).

\*\*Resulting Context\*\*

- Projects implementing InnerSource principles show better scores on measures such as change lead time, deployment frequency, change failure rate, knowledge transfer, and Mean Time to Recovery (MTTR) (P17).

- The case studies of successful products or services using InnerSource are used to drive innovation and engineering efficiency (P04).

- The pilot projects provide a safe zone for teams to experiment (P13).

- Increased collaboration over time leads to less overhead for the maintaining team (P04).

\*\*Limitation/Blockers\*\*

- It can be challenging to define business case and outcomes (P23).

- For IS to become a companywide phenomenon, there needs to be robust communication effort to proactively sharing the success stories (P20).

- To be relevant to a wider audience there needs to be a case study for each type of product or service (P04).

- Cognitive dissonance in people where they appreciate the success of the pilot projects but do not apply it to their project context (P03).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 10 panellists (P04, P07, P08, P09, P11, P15, P17, P20, P23 and P25) to have been implemented in their organizations.

# 17Sample Pattern 17

\*\*Title\*\*

Free up developer time to contribute (S27)

\*\*Patlet\*\*

Lack of free time can prevent many people from contributing to InnerSource projects. Developer time can be freed up by either decreasing their workload with support from product managers and product owners, creating a mechanism whereby team members can request discretionary time to contribute from their managers, or making clear time allocations in developer schedule.

\*\*Problem\*\*

Heavy workloads and multiple priorities prevent engineering teams from actively contributing to InnerSource projects. The lack of time for contributions is largely due to lack of middle-management support which in turn is influenced by budgetary concerns and InnerSource not being part of the KPI's.

\*\*Context\*\*

- Teams are funded based on their productivity which means even if a motivated contributor is interested to help another team with their code review request, they will be questioned by middle-management for their deviation from regular work.

- The absence of explicit agreements and a central coordinating body such as the ISPO

- (InnerSource Promotion Office) contributes to a lack of formal time allocation for contributions.

\*\*Forces\*\*

- Established software development methodologies such as Scrum with its emphasis on delivering features in set timeframes (sprints) also tends to work against allocation of time towards InnerSource which is a long-term process.

- InnerSource is seen as a technical fix rather than a cultural transformation.

\*\*Solutions\*\*

- Create time allocations for learning, innovation, and contribution. This can range from 20% to 5% depending on the organization (P07, P09, P18).

- An indirect method of providing opportunity to engage with InnerSource is to conduct time-boxed hackathons without the pressure to achieve a specific business goal (P04).

\*\*Resulting Context\*\*

- This will help retain talent within the company and help sustain a culture of innovation and improvement (P07, P09, P18).

\*\*Limitation/Blockers\*\*

- Despite time allocations the engagement for documentation improvements, best practice methodology, commitments to standardizing repositories can remain low (P15).

- The time allocation might not be used exclusively by people for InnerSource and can get distributed amongst other areas of interest (for e.g. Continuous Integration, continuous deployment, security etc.) (P20).

- The organization needs to function in a team-based approach where individual or groups of teams operate in the IS approach (P25).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 4 panellists (P07, P15, P18, and P20) to have been implemented in their organizations.

# 18Sample Pattern 18

\*\*Title\*\*

Explain benefits and goals of InnerSource (S30)

\*\*Patlet\*\*

Team culture can make it difficult to implement InnerSource and inculcate collaborative behaviours. This can be resolved by motivating teams to take-up IS it is important by explaining the tangible benefits of IS such as the ability to unblock, reduction in duplication of effort, development of team members, networking with colleagues, and collaborative learning. Additionally, the clear definition of goals along the lines of engineering efficiency, speed of innovation, and developer satisfaction can help in motivating people.

\*\*Problem\*\*

Ingrained team culture is difficult to change and takes persistent effort to get the teams to collaborate with other teams and explore InnerSource. This culture also leads to an unwillingness to look holistically at a problem and identify issues on which the team can collaborate with other teams.

\*\*Context\*\*

- People are comfortable with a certain pattern of working and this leads to a reluctance to take up InnerSource (P01, P04).

- While engineers are happy to adopt IS, middle management are reluctant to implement collaborative ways of working (P08, P18).

- The company funding structure disincentivizes sharing of code as teams funded based on software exclusive to them (P11).

\*\*Forces\*\*

- Policies such as compliance and security affect collaborative initiatives such as IS (P04).

\*\*Solutions\*\*

- focus on the tangible benefits for the team which include ability to unblock, reduction in duplication and repetition of effort, development of team members, networking with colleagues, and collaborative learning (P04, P14, P20, P23).

- Utilize venues such as company conferences can be used to broadcast the benefits of IS (P23).

- Managers take the initiative to educate team members on IS (P09).

\*\*Resulting Context\*\*

- There is a positive increase in team productivity and in the consistency of IS as teams and managers understand the benefits of IS (P08, P09).

- Explaining the benefits makes IS more visible and encourages people to look through the documentation. Adoption of IS is the next logical step once people are convinced of the benefits (P04).

\*\*Limitation/Blockers\*\*

- It is difficult to measure the tangible benefits of IS due to the long-term nature of InnerSource (P08, P17).

- There is a large communication overhead as a few self-motivated IS advocates will not be able to cover a large organization (P20).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 7 panellists (P04, P08, P09, P13, P17, P20, and P23) to have been implemented in their organizations.

# 19Sample Pattern 19

\*\*Title\*\*

Personalized interaction to address biases (S31)

\*\*Patlet\*\*

Lack of awareness of InnerSource can lead people to make wrong assumptions about InnerSource. Presentations and one-on-one discussions can help dispel biases regarding IS. This is particularly important when engineers come from a traditional software development background and are ignorant of the IS workflow. One-to-one discussions can particularly help remove worries especially when engineers have experienced issues with organizational priorities on previous occasions.

\*\*Problem\*\*

Ignorance of IS practices can lead people to have pre-conceived notions about IS. One assumption is the lack of a quality control process due to the diversity of contributors. Another effect is ambiguity in the responsibility of the maintaining team or ignorance of the iterative nature of pull requests.

\*\*Context\*\*

- While there is support for Open-Source initiatives in the engineering community within the organization, there can be misconception that InnerSource may dilute the organizations commitment to Open Source.

- There is a lack of knowledge about tools such as GitHub and InnerSource practices.

- Ignorance about InnerSource practices led to a less smooth workflow in the initial stages of implementation.

- People are reluctant to contribute due to a perception that getting benefits from IS needs a lot of work of work.

- Lack of knowledge sharing leads to a poor understanding of IS which in turn results in poor quality pull requests (P18).

- Cursory knowledge also leads to people across all levels assuming that IS projects come with "no warranty" (P11).

\*\*Forces\*\*

- Managers have implicit trust in the work of their teams, at the same time are reluctant to take code from other teams due to trust issues. This results in different levels of tolerance for quality from own team versus other teams (P01).

- For developers will less experience collaboration is a significant paradigm shift as they are used to incentives based on individual achievement (P04).

\*\*Solutions\*\*

- Presentations and one-on-one discussions can help dispel biases regarding IS. This is particularly important when engineers come from a traditional software development background and are ignorant of the IS workflow (P11).

- The Product Owner takes accountability for sharing the vision and strategic direction for his product and explains this to people and ensures the documentation around it is easy to locate (P18).

\*\*Resulting Context\*\*

- Explaining the benefits makes IS more visible and encourages people to look through the documentation. Adoption is the next logical step once people are convinced of the benefits (P04).

- One-to-one discussions can particularly help remove worries especially when engineers have experienced issues with organizational priorities on previous occasions (P15).

- The discussions help clarify doubts and help the IS advocates understand the different perspectives and backgrounds of people. The problems which surface during the discussions and the solutions can be used in similar discussions later (P11).

\*\*Limitation/Blockers\*\*

- Awareness, acceptance and adoption can be slow to gain traction due to the magnitude of competing priorities and limited cognitive bandwidth (P15).

- It is difficult to scale one-to-one interaction in large organizations (P04, P09).

- It can be difficult to convince managers and engineers who are not from a software development and open source background (P09).

\*\*Known Instances (optional)\*\*

- This pattern has been identified by 7 panellists (P01, P01, P11, P15,P20, and P23) to have been implemented in their organizations.