Santiago de Cali, 28 de mayo de 2021

Ingeniera
Luisa Fernanda Rincón Pérez
Directora Maestría en Ingeniería de Software
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Proyecto de Grado y posteriormente optar por el título de Magíster en Ingeniería de Software, me permito presentar como Director a: Juan Carlos Martínez Arias identificado con C.C. 7.549.319, del Proyecto de Grado denominado "IMPLEMENTACIÓN DE UN GOBIERNO AUTOGESTIONADO PARA EL REPOSITORIO DE SERVICIOS COMUNES DEL LABORATORIO DIGITAL DEL BANCO DE OCCIDENTE BAJO LAS PRÁCTICAS INNERSOURCE COMO ESTRATEGIA DE REUTILIZACIÓN DE SOFTWARE", el cual será realizado por el estudiante Jesús Alexander Andrade Sánchez con código 00007939087.

Atentamente,

Jesús Alexander Andrade Sánchez
C.C. 11812231 de Quibdó

Juan Carlos Martínez Arias
C.C. 7549319 de Armenia (Q)

Santiago de Cali, 28 de mayo de 2021

Ingeniera
Luisa Fernanda Rincón Pérez
Directora Maestría en Ingeniería de Software
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Proyecto de Grado y posteriormente optar por el titulo de Magíster en Ingeniería de Software, nos permitimos presentar a su consideración el anteproyecto de Trabajo de Grado denominado "IMPLEMENTACIÓN DE UN GOBIERNO AUTOGESTIONADO PARA EL REPOSITORIO DE SERVICIOS COMUNES DEL LABORATORIO DIGITAL DEL BANCO DE OCCIDENTE BAJO LAS PRÁCTICAS INNERSOURCE COMO ESTRATEGIA DE REUTILIZACIÓN DE SOFTWARE" el cual será realizado por el estudiante Jesús Alexander Andrade Sánchez con código 00007939087 perteneciente a la Maestría en Ingeniería de Software, bajo la dirección del profesor Juan Carlos Martínez Arias.

El suscrito director del Proyecto de Grado autoriza para que se proceda a hacer la evaluación de este Anteproyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

Jesús Alexander Andrade Sánchez
C.C. 11812231 de Quibdó

Juan Carlos Martínez Arias
C.C. 7549319 de Armenia

Documentación anexa:

Ficha Resumen del Anteproyecto del Proyecto de Grado (PDF).
Una copia digital (PDF) del Anteproyecto del Proyecto de Grado.

Pontificia Universidad
JAVERIANA
Cali

IMPLEMENTACIÓN DE UN GOBIERNO AUTOGESTIONADO PARA EL REPOSITORIO DE SERVICIOS COMUNES DEL LABORATORIO DIGITAL DEL BANCO DE OCCIDENTE BAJO LAS PRÁCTICAS INNERSOURCE COMO ESTRATEGIA DE REUTILIZACIÓN DE SOFTWARE

ALEXANDER ANDRADE
Código. 00007939087

Anteproyecto de trabajo de grado para optar al título de
Magister en Ingeniería de Software

Director
JUAN CARLOS MARTÍNEZ ARIAS

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN INGENIERIA DE SOFTWARE
SANTIAGO DE CALI, MAYO 30 DE 2021

FICHA RESUMEN
ANTEPROYECTO DE TRABAJO DE GRADO

POSIBLE TÍTULO:

1. ÁREA DE TRABAJO: Reutilización de código de software.
2. TIPO DE PROYECTO (Aplicado, Innovación, Investigación): Innovación.
3. ESTUDIANTE: Jesús Alexander Andrade Sánchez.
4. CORREO ELECTRÓNICO: alexander.andrade@gmail.com
5. DIRECCIÓN Y TELÉFONO: Cll 18 # 67-48 apto 512, 317-241-5118
6. DIRECTOR: Juan Carlos Martínez Arias
7. VINCULACIÓN DEL DIRECTOR: Pontificia Universidad Javeriana - Cali
8. CORREO ELECTRÓNICO DEL DIRECTOR: juancmartinez@javerianacali.edu.co
9. CO-DIRECTOR (Si aplica):
10. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Banco de Occidente
11. OTROS GRUPOS O EMPRESAS:
12. PALABRAS CLAVE (al menos 5): InnerSource, Ingeniería de Software, desarrollo de software basado en componentes reutilizables, reusabilidad, activo de software, reutilización de software, línea de productos.
13. FECHA DE INICIO: Julio 2021
14. DURACIÓN ESTIMADA (En meses): 8 meses
15. RESUMEN:

Este anteproyecto abordará cuestiones relacionadas a las dinámicas de codificación de software en el paradigma de Desarrollo Basado en Componentes a través de las prácticas InnerSource (ISS), utilizadas por empresas como Paypal o Microsoft, las cuales incorporan estrategias y cultura de desarrollo de Software Libre dentro de los confines de las empresas como desarrollo colaborativo, desarrolladores no co-localizados, fomento para dar y recibir aportes de código, reutilización, y reducción de la burocracia y otros comportamientos “tribales”, entre otras; siendo este el primer trabajo académico realizado, a saber, en español y en Latinoamérica sobre ISS. Según (Sommerville, 2011), la reutilización es la técnica de Ingeniería de Software avanzada que ayuda a reducir los costos de producción y mantenimiento de software, y aumentar la calidad y velocidad de entrega de los sistemas y esto es lo que ha buscado el Laboratorio Digital del Banco de Occidente al establecer que todas las aplicaciones se realicen como microservicios, ubicando los reutilizables en el repositorio de Servicios Comunes (RSC) pero el Laboratorio, al igual que 59% de los encuestados por (Capilla et al., 2019) no tiene prácticas de reutilización definidas por lo tanto, no logra asegurar la confianza en el código compartido debido a la escasa documentación, poca claridad sobre responsabilidades o la baja trazabilidad de impactos en otros

productos que tendrían las contribuciones al código; lo que lleva a reinventar soluciones existentes afectando el *time-to-market*. La organización conoce la situación y necesita resolverla cumpliendo dos restricciones: utilizar métodos comprobados por líderes de la industria del software y ser autogestionada por los desarrolladores. La revisión bibliográfica encontró que las prácticas ISS cumplen con las necesidades planteadas. Así, el principal objetivo de este trabajo es Implementar un gobierno autogestionado para el RSC del Laboratorio bajo las prácticas ISS como estrategia de Reutilización de Software. Este trabajo se realizará en cuatro fases correspondientes al diagnóstico de la situación, diseño de una solución personalizada, implementación y evaluación de la solución. Los resultados esperados, en términos de entregables son: Sitio ISS para el inventario de componentes de software migrados al Gobierno ISS, Plantilla de GitHub para los nuevos, migración al Gobierno ISS de todos los componentes del RSC, instructivos editables del Gobierno ISS y monografía con documentación del proceso realizado para llevar a cabo el trabajo, lo cual podría servir como guía para aplicarlo en otras organizaciones o realizar investigaciones sobre estas prácticas y nuestra cultura laboral.

TABLA DE CONTENIDO

1 LISTA DE FIGURAS

2 LISTA DE TABLAS

3 GLOSARIO

ISS: InnerSource Software.

RSC: Repositorio de Servicios Comunes.

Componente (reutilizable de software): De acuerdo con (Sommerville, 2011, pág. 453) los componentes son abstracciones de alto nivel en comparación con los objetos y se definen mediante sus interfaces. Por lo general, son más grandes que los objetos individuales y todos los detalles de implementación se ocultan a otros componentes. Los servicios o microservicios son un buen ejemplo de este término.

Activo (reutilizable) de software: es un producto diseñado expresamente para ser empleado de forma recurrente en el desarrollo de muchos sistemas y aplicaciones. Ejemplos de activos reutilizables son: algoritmos, patrones de diseño, esquemas de base de datos, arquitecturas de software, especificaciones de requerimientos, de diseño y de prueba, entre otros (Montilva et al., 2003, pág. 2).

4 DEFINICIÓN DEL PROBLEMA

4.1 PLANTEAMIENTO DEL PROBLEMA

Uno de los principales desafíos que enfrentan las empresas es aumentar la velocidad del desarrollo de software al encontrar formas más eficientes para que los desarrolladores colaboren entre sí, compartan conocimientos, reutilicen el código desarrollado y reduzcan los comportamientos “tribales”. En la última década, la Transformación Digital ha creado un sentido de urgencia para que todo tipo de empresas desarrollen estas habilidades. Sumado a esto, la irrupción de la COVID-19 y los confinamientos impuestos a nivel mundial durante el último año, han llevado a las empresas a acelerar décadas de planes de digitalización para dar respuesta a esta nueva realidad de interactuar con los clientes sin contacto físico y han surgido nuevas necesidades en la dinámica de los equipos de desarrollo de software en entornos no co-localizados.

Para transformar su negocio, Banco de Occidente cuenta desde 2019 con un Laboratorio Digital que desarrolla software como experimentos para resolver las necesidades de sus clientes o incluso intentar anticiparlas. Este laboratorio está compuesto por diferentes "células" de desarrolladores, cada una a cargo de un producto que responde a necesidades únicas. Desde 2020, el Laboratorio trabaja de forma 100% remota, lo que añade una nueva capa de complejidad a sus procesos.

Todos los desarrollos del Laboratorio se llevan a cabo bajo el enfoque de microservicios, como componentes reutilizables para lograr los objetivos de un producto en particular. Esto ha sido visto por las directivas como una oportunidad para reutilizar software bajo el paradigma de Desarrollo Basado en Componentes que, según (Sommerville, 2011, pág. 426) desde principios de la década de 2000 ha sido la respuesta a las crecientes demandas para reducir los costos de producción y mantenimiento del software, y aumentar la calidad y velocidad de entrega de los sistemas. Para ello, el Laboratorio ha habilitado un repositorio de códigos denominado de Servicios Comunes (RSC), donde las células pueden localizar desarrollos que puedan ser de utilidad para otros equipos de trabajo. Sin embargo, el Laboratorio, al igual que el 59 % de los encuestados por (Capilla et al., 2019, pág. 11), no tiene prácticas de reutilización definidas, por lo que el RSC no promueve ni asegura, entre otros, la confianza en el código depositado debido a situaciones como la escasa o nula documentación, poca claridad sobre responsabilidades o la baja trazabilidad de impactos que pueden haber en otros productos al hacer contribuciones al código; lo que se traduce en esfuerzos aislados para hacer soluciones que ya existen (“reinventar la rueda”) con su consecuente afectación a los tiempos de salida al mercado (*time-to-market*). Esta situación ampliamente conocida por las directivas del Laboratorio da comienzo a este trabajo de

grado y de cuyo primer acercamiento surgieron una serie de interrogantes que orientarían la búsqueda de una solución:

- ¿Cómo **garantizar que los desarrolladores contribuyan** al repositorio de Servicios Comunes (RSC)?
- ¿Cuál es la mejor manera de **estandarizar y obtener consistencia** en la documentación y códigos almacenados en este repositorio?
- ¿Cómo se asegura de que el RSC **genere un alto nivel de confianza** para que las células aprovechen estas soluciones en lugar de emprender esfuerzos individuales?
- ¿Cómo **facilitar el aporte de mejoras** al RSC por parte de cualquier desarrollador del Laboratorio independientemente de la célula a la que pertenezca?
- ¿Cómo se asegura de que quienes aporten tengan la **visión sobre el impacto que tendrán sus mejoras en otros productos**?
- ¿Cómo lograr que todas las Células se **coordinen y auditen el repositorio de forma autónoma**, es decir, sin que exista una estructura de administración distinta a los propios desarrolladores?
- ¿Qué **buenas prácticas de coordinación se podrían extraer de proyectos con miles de colaboradores** como Linux?
- Y finalmente, ¿**cómo resuelven estas inquietudes los grandes desarrolladores de software** como Google, Microsoft, entre otros?

El Laboratorio ha restringido la búsqueda a soluciones utilizadas en referentes de la industria y no se equivoca en querer imitar a otros, ya que la imitación de métodos que han demostrado ser exitosos es una práctica de mejora común entre las empresas que desarrollan software (Stol et al., 2011, pág. 1). En este orden de ideas, la revisión bibliográfica determinó que los referentes citados resuelven estas situaciones a través de las prácticas InnerSource (ISS). Estas podrían ayudar al Laboratorio a establecer una gobernanza sobre el RSC para fomentar de manera autogestionada el intercambio no solo de microservicios (componentes de software), como sucede en la actualidad, sino también de los activos de software relacionados con estos como el código, arquitecturas, especificaciones y diseños, y contribuir a la distribución de los conocimientos adquiridos durante la investigación previa al desarrollo, de las decisiones técnicas tomadas y de las buenas prácticas entre toda la plantilla de desarrolladores.

Por todo lo anterior, es necesario establecer un gobierno para el repositorio de Servicios Comunes bajo las prácticas ISS como estrategia de reutilización, que se adapte al Laboratorio Digital del Banco de Occidente y sus necesidades; siendo éste, a saber, el primer trabajo académico en español, y en Latinoamérica, que se realiza sobre este tema y que podría llegar a convertirse en un referente para la aplicación e investigación de estas prácticas en el mundo de habla hispana.

4.2 FORMULACIÓN DEL PROBLEMA

De acuerdo con el planteamiento del problema, se realiza la siguiente formulación:

> ¿Cómo implementar un gobierno autogestionado para el repositorio de Servicios Comunes del Laboratorio Digital del Banco de Occidente bajo las prácticas InnerSource como estrategia de Reutilización de Software?

5 OBJETIVOS DEL PROYECTO

5.1 OBJETIVO GENERAL

Implementar un gobierno autogestionado para el repositorio de Servicios Comunes (RSC) del Laboratorio Digital del Banco de Occidente bajo las prácticas InnerSource (ISS) como estrategia de Reutilización de Software.

5.2 OBJETIVOS ESPECÍFICOS

1. Diagnosticar los inconvenientes actuales relacionados con la gestión del RSC que limitan la reutilización de los componentes de software contenidos en este repositorio.
2. Diseñar un gobierno autogestionado para el RSC bajo las prácticas ISS que resuelva los problemas diagnosticados con respecto a la gestión de este repositorio con el fin de reutilizar los componentes de software que contiene.
3. Implementar el Gobierno ISS diseñado en todos los componentes de software del RSC.
4. Validar que la implementación del Gobierno ISS resuelva los problemas diagnosticados en materia de gestión del RSC que limitan la reutilización de los componentes de software contenidos en este repositorio.

6 RESULTADOS ESPERADOS

Los resultados esperados, en términos de entregables, se enumeran a continuación:

- Para el Laboratorio Digital del Banco de Occidente:
  - Sitio InnerSource (ISS) que mantiene un inventario de los componentes de software migrados al Gobierno ISS.
  - Plantilla de GitHub para los nuevos proyectos con toda la documentación necesaria.
  - Migración al Gobierno ISS de todos los componentes de software del repositorio de Servicios Comunes (RSC) que existan en el momento de la realización de este proyecto.
  - Instructivos editables del Gobierno ISS.
- Para la Universidad Pontificia Javeriana y comunidad académica:
  - Monografía con documentación del proceso llevado a cabo para realizar el trabajo de grado. Este a su vez contendrá copias de los documentos entregados a la organización donde se ofuscará la información privada de la empresa, así como recomendaciones para futuros trabajos de investigación sobre el marco teórico de esta monografía o del proyecto implementado en la organización.

7 ALCANCE

El alcance del presente trabajo de grado, que está delimitado por los OBJETIVOS DEL PROYECTO y los RESULTADOS ESPERADOS, es implementar un gobierno autogestionado para el repositorio de Servicios Comunes del Laboratorio Digital del Banco de Occidente bajo las prácticas InnerSource (ISS) como estrategia de Reutilización de Software. Como delimitaciones (alcance) se tiene que:

- La solución no incurrirá en costo alguno, ni requerirá la adquisición o desarrollo de Software, ya que aprovecha las herramientas existentes en el Laboratorio.
- La evaluación del proyecto se realizará mediante encuestas dirigidas a todos los desarrolladores del Laboratorio para identificar si se han resuelto los problemas del repositorio de Servicios Comunes (RSC) que limitan la reutilización de los componentes de software depositados en este repositorio. Dentro de la evaluación no se contabilizará el número de aportes o aumento de contribuyentes tras la implementación del Gobierno ISS, ya que esto no puede ser influenciado por este trabajo de grado, pues esto depende de la necesidad de reutilización o del surgimiento de nuevas iniciativas comunes.

7.1 EXCLUSIONES

Quedarán fuera del alcance de este trabajo de grado:

- Implementaciones fuera del contexto específico del trabajo de grado. No obstante, durante el desarrollo de este proyecto o después su finalización, el Gobierno ISS puede implementarse fuera del contexto original sin que ello implique participación, responsabilidad, notificación a, o soporte de este trabajo de grado.
- El soporte de la solución, su continuidad o evolución, tras la entrega del trabajo de grado.
- No se desarrollarán automatizaciones como informes de uso del repositorio de Servicios Comunes o buenas prácticas utilizadas en general, por proyecto o por desarrollador.
- Gestión o gobierno de activos de software que se encuentran en el RSC y no es posible establecer una célula responsable o usuaria.
- Cualquier otra actividad, tema o elemento no mencionado en el alcance.

8 JUSTIFICACIÓN

La industria financiera está atravesando una vertiginosa Transformación Digital que la impulsa no solo a crear nuevos productos, sino también a impactar en el mercado en el menor tiempo posible, incluso tratando de adelantarse a su competencia. La reutilización de software es una de las estrategias que ha encontrado el Laboratorio Digital del Banco de Occidente para acelerar su velocidad y ritmo de desarrollo, y para ello ha establecido un repositorio de Servicios Comunes (RSC); pero la escasa o nula documentación y la ausencia de reglas de uso claras han hecho que los desarrolladores no confíen en los desarrollos allí depositados, teman hacer mejoras en el código existente para no impactar a otros productos, mientras que otros se abstienen de colaborar para evitar dar soporte a modificaciones hechas a posteriori por otros equipos que no documenten ni se responsabilicen. Es urgente solucionar esta situación para lograr la cadencia deseada por la organización, ya que la empresa es consciente de esta limitación, pero hasta el momento no ha iniciado esfuerzos para solucionarla.

Este trabajo pretende resolver la situación estableciendo un gobierno autogestionado para el uso del RSC que incentive la reutilización de los componentes de software existentes y fomente la participación de todos los desarrolladores del Laboratorio, incluso en soluciones en proyectos diferentes a los que están asignados. El impacto de este trabajo es de carácter organizacional, no limitado a un solo proyecto de la empresa, ya que promueve un cambio cultural en las relaciones sociales necesarias para producir software a nivel industrial, incidiendo positivamente en el clima laboral, la gestión del conocimiento, en la velocidad de desarrollo y calidad del producto, y reduciendo el *time-to-market* de las soluciones desarrolladas. Como aporte al ámbito académico, este sería el primer trabajo en español, y en Latinoamérica, que se realiza sobre InnerSource (ISS) y que podría llegar a convertirse en un referente para la aplicación e investigación de estas prácticas en el mundo de habla hispana.

9 MARCO TEÓRICO DE REFERENCIA Y ANTECEDENTES

A lo largo de esta sección se muestra los conceptos básicos de la Reutilización de Software como estrategia para aumentar la velocidad de desarrollo y reducir costos. También se encontró que es común entre muchos referentes del desarrollo de software el uso de las prácticas InnerSource (ISS) para crear el ambiente propicio para favorecer la reutilización de software.

Tras establecer este marco teórico, se realizó un recuento de los trabajos relacionados que aportan información relevante para el presente trabajo. También se analizó el estado del arte, es decir, cuál es la situación actual de la Reutilización de Software y de las prácticas ISS.

9.1 REUTILIZACIÓN DE SOFTWARE

De acuerdo a la IEEE, "la reutilización de software implica capitalizar el software y los sistemas existentes para crear nuevos productos" (IEEE, 2010). El término Reutilización de Software se acuñó en la primera Conferencia Internacional sobre Ingeniería de Software (ICSE) en 1968 (Capilla et al., 2019, pag. 1). Desde entonces, academia e industria han estudiado la reutilización de software para aumentar la productividad y reducir costos (Barros-Justo et al., 2019, pag. 1), aumentando cada año el número de publicaciones por lo que se considera que sigue siendo un tema de interés en la Ingeniería de Software. Además, la reutilización de software puede describir tanto el uso de componentes preexistentes para desarrollar nuevas aplicaciones como la mejora de un producto antiguo (sistemas heredados o legados / *legacy*) (Barros-Justo et al., 2019, pag. 2).

La Tabla 1 relaciona una serie de términos, enfoques y ámbitos de aplicación de la reutilización de software, de acuerdo a la revisión literaria realizada por (Barros-Justo et al., 2019, pags. 1 y 2).

Tabla 1. Clasificación de la Reutilización de Software

Términos relacionados	1. Desarrollo basado en componentes, *Component Based Development* (CBD), 2. Líneas de productos de software, *Software Line Products* (SPL), 3. Desarrollo dirigido por modelos, *Model Driven Development* (MDD), 4. Ingeniería de dominio (o análisis de dominio), 5. Comercial de caja, *Commercial-Of-The-Shelf* (COTS), 6. Entre otros.
Enfoques	7. Con reutilización: desarrollo utilizando componentes preexistentes.

principales	8. Para la reutilización: desarrollo de componentes reutilizables.
Ámbito de aplicación	9. Reutilización vertical: reutilización de software en un dominio de aplicación determinado. 10. Reutilización horizontal: reutilización de componentes en varios dominios de aplicación.

Fuente: Propia, basada en (Barros-Justo et al., 2019, pags. 1 y 2)

9.2 PRÁCTICAS INNERSOURCE

El ISS es formalmente definido como "El uso de principios y prácticas de Desarrollo de Software Libre (OSSD: *Open Sourced Software Development*) dentro de los confines de una empresa" (ISC, 2021, pág. 3). *Inner Sourced* Software (ISS) promete resolver los problemas del desarrollo de software tradicional facilitando la reutilización del software y permitiendo que las partes dentro de una organización colaboren más allá de los límites internos de la organización (Capraro, 2020 pág. V). Dentro de los casos de éxito de esta práctica se encuentran Google, Uber, SAP, Paypal, Nike, American Airlines, BBC, Europace, Robert Bosch GmbH, DB Systel, Elbit Systems, Entelgy, Zylk, Bitergia, Hewlett-Packard, Alcatel-Lucent, Philips Healthcare, IBM, Nokia o el Laboratorio de Propulsión Jet (JPL) de la NASA (ISC, 2021) (Edison et al., 2018, pág. 1) (Stol & Fitzgerald, 2014, pág. 1) (Mittman, 2018).

Para llevar acabo un diagnóstico del estado del uso de ISS se analizaron artículos, tesis y libros sobre el tema, teniendo en cuenta criterios como su relevancia en cuanto a citaciones, así como la relación de casos de éxito en empresas reconocidas. A partir de este análisis, se determinó el impacto del ISS en las organizaciones, sus factores de éxito o fracaso, de tal forma que permitan plantear estrategias para adaptar el marco dentro una empresa local. Este análisis también sirve como un punto de partida para nuevas investigaciones y sus posibilidades de adaptación a otras empresas.

Para realizar la investigación se hicieron búsquedas inicialmente en Google sobre prácticas de colaboración de software en empresas como Google o Microsoft. Pronto, las búsquedas dirigieron hacia las prácticas ISS y con esta información se procedió a buscar en Google Scholar con los términos "*Inner Source*" y "Software", filtrando resultados entre 2016 y 2021. El filtro por idioma español no arrojó resultados. Luego se hizo esta misma búsqueda en la editorial tecnológica O'Reilly donde se encontraron varios libros al respecto. De los resultados arrojados por la búsqueda, se eligieron 5 artículos, 1 tesis de doctorado y 3 libros, que resultan ser los mas referenciados sobre el tema.

9.2.1 Aspectos generales

En las empresas que desarrollan software se presentan diferentes problemas a la hora de coordinar el conocimiento adquirido por las células de desarrolladores, así como la reutilización del software desarrollado para apalancar otros desarrollos. El ISS se presenta como una forma de crear comunidades de desarrolladores, como las de OSSD, dentro de las instalaciones de la organización habilitando que cada desarrollador pueda contribuir, retroalimentar y más, a los repositorios de los proyectos de la organización (Mittman, 2018, págs. 5-6).

De acuerdo a Tim O'Reilly[1] "El éxito del Software Libre que tiene más que ver con sus capacidades de colaborar que con el simple hecho de programar" (O'Reilly, 2000) ha hecho que las empresas se interesen en adoptar estas prácticas para sus procesos de software; fenómeno que O'Reilly acuñó como *Inner Source* (ISS) (O'Reilly, 2000).

La implementación de ISS cuenta con el amplio apoyo de las investigaciones académicas, casos de éxitos en grandes compañías que desarrollan software para uso interno y externo, y la comunidad Inner Source Commons (ISC), fundada por PayPal en 2015 como un foro abierto a la discusión y mejora de las prácticas (Mittman, 2018, pág. 37).

9.2.2 Las ventajas del OSSD deseadas por las empresas

Intentando averiguar sobre las buenas prácticas para el desarrollo de software común en las organizaciones, se hizo evidente que era necesario antes entender las buenas prácticas en el desarrollo de los proyectos de código abierto (OSS: *Open-Sourced Software*), entre las que se encuentran:

- La colaboración abierta fomenta una mayor contribución y por ende la innovación (Mittman, 2018, pág. 8). El acceso universal a todos los artefactos de desarrollo, como el código y la documentación, permite que cualquiera pueda inspeccionar y enviar contribuciones (Stol & Fitzgerald, 2014, pág. 1) mientras el proyecto y la organización se benefician de la creatividad y diversidad de perspectivas de toda la plantilla de desarrolladores (Oram, 2021, pág. 8).
- Aumenta la calidad de los desarrollos mediante la revisión por pares independientes de las contribuciones de otros en la "comunidad" de desarrolladores (Mittman, 2018, pág. 8) (Stol & Fitzgerald, 2014, pág. 1). Establecer un sistema formal y objetivo para probar cada contribución, basados en técnicas de pruebas continuas (*continuous testing*), fomenta la confianza entre los desarrolladores y permite determinar cuales de ellos pueden recibir mayores responsabilidades respecto al proyecto (Oram, 2021, págs. 10-11).
- Documentación completa depositada en el repositorio del código (Oram, 2021, pág. 17).

[1] Tim O'Reilly, fundador de la editorial tecnológica O'Reilly.

- Comentarios tempranos y lanzamientos frecuentes para mantener vivo un proyecto y mejorarlo rápidamente (Stol & Fitzgerald, 2014, pág. 1).
- Las personas que se encuentran a grandes distancias geográficas, en momentos distintos, pueden trabajar en el mismo código o contribuir con diferentes archivos de código al mismo proyecto (Oram, 2021, pág. 7).
- La comunicación tiende a escribirse y ubicarse en sitios públicos en lugar de compartirse informalmente de boca en boca, lo que proporciona una historia del proyecto, así como oportunidades de aprendizaje para los nuevos miembros del proyecto. Autoselección de tareas para permitir a los desarrolladores identificar partes del software que creen que pueden mejorar o defectos que pueden corregir (Oram, 2021, págs. 7-8) y (Stol & Fitzgerald, 2014, pág. 1).

9.2.3 Ventajas del ISS

Grandes compañías como Google, Uber, SAP, Paypal, Nike, American Airlines, BBC, Europace, Robert Bosch GmbH, DB Systel, Elbit Systems, Entelgy, Zylk, Bitergia, Hewlett-Packard, Alcatel-Lucent, Philips Healthcare, IBM, Nokia o el Laboratorio de Propulsión Jet (JPL) de la NASA (ISC, 2021) (Edison et al., 2018, pág. 1) (Stol & Fitzgerald, 2014, pág. 1) (Mittman, 2018) han implementado el marco ISS con el ánimo de obtener las ventajas de OSS dentro de los desarrollos internos (del inglés "*inner*") y lograr otras ventajas en el ámbito corporativo como:

- Aumento de productividad, eficiencia y efectividad (Capraro, 2020, pág. 162) ya que no siempre los desarrolladores tendrán que iniciar desde cero (Mittman, 2018, pág. 9). La velocidad de desarrollo es potencialmente tan grande como todo el personal de desarrollo de la organización, lo que puede resultar en un tiempo de comercialización más rápido (*time to market*) (Stol & Fitzgerald, 2014, pág. 2) y (Capraro, 2020, pág. 162). El desarrollo se vuelve más rápido a medida que los programadores aprenden a usar pruebas unitarias, cobertura de código e integración continua para eliminar errores en una etapa temprana del desarrollo. Los comentarios escritos intercambiados entre los miembros del equipo, aunque toman algo de tiempo, se pagan sobradamente al ayudar a los nuevos programadores a aprender el sistema más rápido (Oram, 2021, pág. 8).
- Reducción de costos de desarrollo al aumentar nivel de reutilización de software (Capraro, 2020, pág. 162) a través de productos y componentes de software que están disponibles como proyectos ISS para que los use cualquier persona dentro de una organización (Stol & Fitzgerald, 2014, pág. 1) (Oram, 2021, pág. 8) reduciendo la duplicación y desperdicio de esfuerzos (Mittman, 2018, pág. 9). Esto también logra una reducción de dependencia entre el equipo desarrollador original y sus contribuidores (Capraro, 2020, pág. 162).
- Calidad mejorada mediante el aprovechamiento de la Ley de Linus[2]: "*Dado un número suficientemente elevado de ojos, todos los errores se vuelven obvios"* (Stol & Fitzgerald, 2014, pág. 1). Además, el explicar un tema complejo a varias personas que quieran contribuir permite reconocer posibles mejoras arquitecturales (del software) (Oram, 2021, pág. 19).

[2] Linus Torvalds, iniciador y líder *kernel* (núcleo) del Sistema Operativo Linux.

- Mayor innovación en el desarrollo de software (Capraro, 2020, pág. 162).
- Mejora en la documentación del código, tanto formalmente (como comentarios y documentación en el código) como informalmente (en listas de discusión). La documentación proporciona un historial del proyecto y ayuda a personas externas a comprenderlo, y comprender las decisiones tras los cambios, lo que aumenta la contribución (Oram, 2021, págs. 8-9).
- Colaboración entre equipos relativamente fluida. El código contribuido rara vez tiene que ser reescrito por el equipo que lo recibe, y no se requieren discusiones a un alto nivel gerencial (Oram, 2021, pág. 8). El análisis del código contribuido por otros ayuda a crear nuevas y sofisticadas habilidades dentro del equipo (Oram, 2021, pág. 20). Los equipos distantes geográficamente se ven mayormente beneficiados (Capraro, 2020 pag. 122).
- Mayor movilidad del personal como efecto de que los desarrolladores se familiaricen con varios proyectos de software, así como con las herramientas utilizadas en un repositorio de plataforma central (Stol & Fitzgerald, 2014, pág. 2).
- Acelera la incorporación (*On-boarding*) de nuevos desarrolladores (Mittman, 2018, pág. 13) al reducir tiempos y costos relacionados (Oram, 2021, págs. 14-15).
- Fortalecimiento de los procesos, la confianza y la alineación mediante la toma de decisiones transparente (Mittman, 2018, pág. 7).
- Rompimiento de silos (Capraro, 2020, pág. 162) como resultado de una cultura de la colaboración, apertura y transparencia (Mittman, 2018, pág. 13), logrando distribuir los costos y riesgos en toda la organización y mayor intercambio de información (Capraro, 2020 pag. 162).
- Apoya a la mejora del clima laboral: aumenta la satisfacción laboral al permitirles a todos aportar, crea sentido de comunidad, ayuda a mejorar las habilidades de todos los contribuidores, a crear reputación, a fomentar la creación de mentores, a que cada desarrollador persiga sus intereses, a compartir ideas en igualdad de condiciones y aprender más fácilmente de sus compañeros (Mittman, 2018, pág. 11 y 14) y (Capraro, 2020 pags. 21 y 162).

9.2.4 Anatomía de un proyecto ISS

9.2.4.1 Elementos claves del ISS

Son cuatro elementos clave que constituyen ISS (Capraro, 2020, pág. 3):

1. Cultura de colaboración abierta: equidad, meritocracia y auto-organización.
2. Entorno de desarrollo abierto.

3. Comunidades alrededor del software: .
4. Al menos un escenario de ISS específico:
    - Reutilización participativa: participar del desarrollo y mantenimiento del desarrollo reutilizado.
    - Autoselección de tareas: desarrollo realizado durante el horario laboral.
    - Voluntariado: desarrollo realizado fuera del horario laboral.
    - Proyectos de desarrollo colaborativo.

9.2.4.2 Modelos de ISS

Se encuentran dos modelos para implementar ISS (Stol et al., 2011, pág. 7):

- Basado en infraestructura: donde la organización pone a disposición las herramientas comunicativas y de gestión de código.
- Basado en proyectos: donde la organización delega la responsabilidad de los "recursos compartidos" a un equipo.

La Tabla 2 muestra las principales diferencias entre ambos modelos:

Tabla 2. Diferencias claves de los modelos de ISS

Característica	**Basado en infraestructura**	**Basado en proyectos**
Reutilización	Oportunista, ad-hoc. Maximice el número de proyectos a compartir dentro de la organización.	Planificado estratégicamente. Optimización de la reutilización de activos críticos.
Soporte	Opcional, difiere según el proyecto y depende del propietario / encargado del mantenimiento y de la actividad de la "comunidad".	El esencial para el éxito del ISS.
Propietario	Creador / propietario individual del proyecto.	Equipo central.
Tipos de paquetes de software	Paquetes de software diversos (por ejemplo, utilidades, herramientas, compiladores, *shells*).	Activos críticos (por ejemplo, plataforma de una línea de productos de software). Tecnología primaria, en lugar de herramientas y utilidades.

Fuente: (Stol et al., 2011, pág. 7).

9.2.4.3 Estructura

Muchos proyectos OSSD siguen una estructura organizativa similar, una que muchas empresas pueden tener en cuenta al configurar equipos multifuncionales para los proyectos de ISS (Mittman, 2018, pág. 15):

- Mantenedores: Colaboradores que son responsables de impulsar la visión y gestionar los aspectos organizativos del proyecto. No son necesariamente los propietarios o autores originales del código. Entre estos se encuentran: Desarrolladores, gerentes de productos y otros tomadores de decisiones clave (Mittman, 2018, pág. 15).
- Colaboradores: todos los que han contribuido con algo al proyecto. Entre estos se encuentran: Desarrolladores, DevOps, UX/UI, Analítica y otros roles (Mittman, 2018, pág. 15).
- Miembros de la comunidad: personas que utilizan el proyecto. Pueden participar activamente en conversaciones o expresar su opinión sobre la dirección del proyecto (Mittman, 2018, pág. 15).

9.2.4.4 Herramientas básicas

Entre las herramientas básicas para la adopción de ISS se encuentran, sin que sean las únicas:

- Sistemas de control de versiones de fuentes como GitHub®, que permitan documentar los cambios en el código, así como la solicitud de revisiones para integrar los cambios (Mittman, 2018, pág. 17).
- En el sistema de control de versiones se deben gestionar "*issues* (propuestas)", "solicitudes de características" (nuevas funcionalidades) o "reportes de *bugs*", a manera de registrar las discusiones para consulta posterior (Mittman, 2018, pág. 17).
- Sistemas de chat empresarial que persistan las conversaciones y estas se puedan consultadas de por otros como sucede en Slack® (Mittman, 2018, pág. 17).

9.2.4.5 Gestión de contribuciones

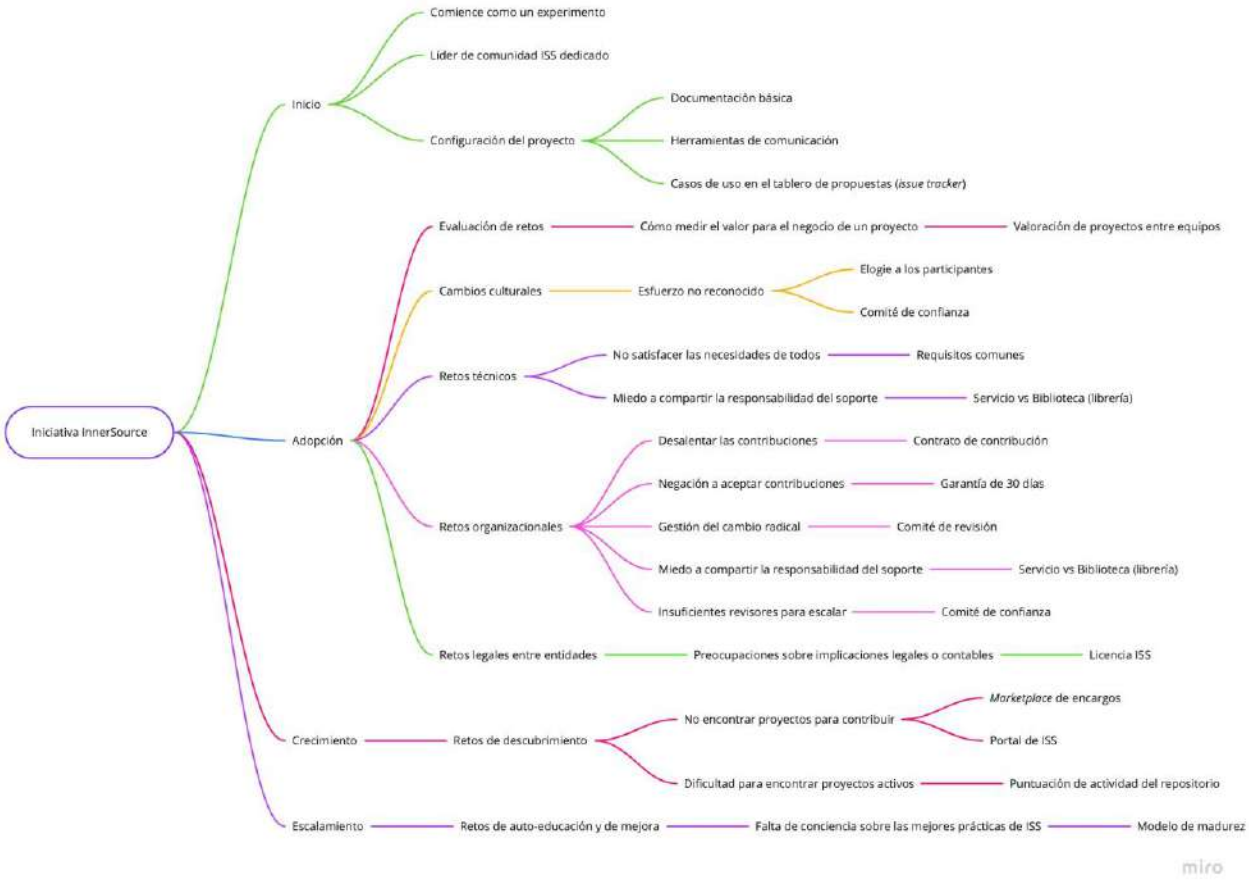
Para que las contribuciones sean realmente efectivas se deben establecer y seguir una serie de normas:

- Establecer reglas sobre las características que debe tener una contribución para ser aceptada, así como el tamaño máximo que debe tener una *pull-request* (Mittman, 2018, pág. 31).
- Hacer que el proyecto ISS sea fácil de encontrar (Mittman, 2018, pág. 34):
    - Breve descripción del proyecto: título y *tagline*.
    - Enlace al sitio web del producto.
    - Uso de herramientas de GitHub® como *issues* (*Bug*, Pregunta o Solicitud de función) y *pull-requests*.
    - Uso de temas o tópicos para promover la visibilidad.
    - Código de conducta, pautas de contribución y un archivo de licencia.
- Todos los proyectos deben estar abiertos a todos los desarrolladores para su revisión y posible aporte (Mittman, 2018, pág. 35).

9.2.5 Retos del ISS

Estas atractivas ventajas han hecho que un buen número de empresas, sobre todo grandes, reporten éxitos relacionados con el uso de esta práctica, pero la misma supone retos importantes a la hora de integrar los desarrollos comunes dentro de los productos (Stol et al., 2011, pág. 1). Los retos mas comunes han sido resueltos de forma comprobada y repetible, y documentados por la comunidad Inner Source Commons (ISC) en forma de patrones (ISC, 2021, págs. 3-4). La Figura 1 muestra el mapa mental de relaciones entre los principales retos de ISS y los patrones recopilados por la comunidad ISC.

Figura 1: Mapa mental de los principales retos de ISS y sus patrones.



Fuente: Traducción propia basada en (ISC, 2021, pág. 7).

Estos patrones deben usarse con cuidado, no se pueden aplicar indiscriminadamente, en la mayoría de los casos, deberá adaptar la solución dada a su situación particular (ISC, 2021, pág. 4). En la Tabla 3 se presenta un resumen de los patrones documentados por la ISC hasta la fecha de redacción de este artículo, contrastando el problema que resuelve, la solución y la empresa que ha reportado casos de éxito. Estos elementos se son relacionados por la investigación independiente de (Capraro, 2020, pág. 122).

Tabla 3. Breve resumen de los patrones ISC

Patrón	Problema	Solución	Caso
Garantía de 30 días	Existe renuencia natural a aceptar contribuciones. Algunos aducirán que es mejor desarrollar la solución antes que analizar el código externo (Oram, 2021, pág. 20); o que el nuevo código comprometerá el futuro de la solución (Stol et al., 2011) citado por (Oram, 2021, pág. 20).	El contribuidor se compromete a proporcionar correcciones de errores al equipo receptor, lo que aumentará el nivel de confianza entre ambos y aumentará la probabilidad de aceptación de las contribuciones.	Paypal
Requisitos comunes	El código común no satisface las necesidades de todos los equipos de proyecto que desean usarlo.	Alineación y refactorización de requisitos.	Paypal
Herramientas de comunicación	Los equipos usuarios tienen problemas para obtener ayuda y ponerse en contacto con el equipo del proyecto.	Configurar y documentar herramientas que permitan que las discusiones sean visibles, almacenadas y fáciles de encontrar.	Europace, Paypal
Contrato de contribución	La gerencia a la que pertenece el desarrollador le desalienta a que contribuya a ISS.	Realizar contratos y acuerdos formales donde deje claro, entre otras, cuanto porcentaje de tiempo del contribuidor necesita la organización para ISS.	Robert Bosch GmbH
Valoración de proyectos entre equipos	Es difícil vender el valor de los proyectos de ISS entre equipos que no tienen un impacto directo en los ingresos de la empresa.	Crear valoraciones basadas en datos para visibilizar los aportes.	Nike
Líder de comunidad ISS dedicado	¿Cómo se asegura de que una nueva iniciativa de ISS aumente su impacto en su comunidad de desarrolladores?	Seleccione personas con habilidades técnicas y de comunicación para liderar y garantizar el éxito en el inicio de una iniciativa de ISS.	Robert Bosch GmbH

Patrón	Problema	Solución	Caso
Marketplace de encargos	Los contribuidores no saben como participar en un proyecto ISS puede beneficiarle; tampoco saben como indicarle a su gerencia el compromiso con un proyecto. Los gerentes no tienen como realizar seguimiento y recompensar a quienes aporten a un proyecto ISS.	Liste las necesidades de la organización a manera de "encargos" (gigs), con requisitos explícitos de tiempo y habilidades. Esto dará a los gerentes una visión global de las necesidades de la empresa.	SAP
Licencia ISS	Dos entidades legales que pertenecen a la misma organización desean compartir ISS entre sí, pero les preocupan las implicaciones legales o contables.	Crear licencias de uso de software y código ISS.	DB Systel
Portal de ISS	Los proyectos no atraen contribuciones. Los contribuidores no tienen una manera fácil de descubrirlos.	Cree un sitio web interno que indexe la información disponible de los proyectos ISS para facilitar que los que atraigan contribuidores.	SAP, Elbit Systems, American Airlines
Casos de uso en el Tablero de Propuestas (*Issue Tracker*)	El proyecto ISS no logra que los planes y el progreso sean transparentes, tampoco el contexto para los cambios.	Aumente los casos de uso para que el Tablero de Propuestas (*Issue Tracker)* del proyecto también sirva para la lluvia de ideas, la discusión de la implementación y el diseño de funciones.	Europace
Modelo de madurez	La organización ha comenzado a adoptar ISS. La práctica se está extendiendo a varios proyectos. Sin embargo, no se comprende bien lo que constituye un proyecto ISS.	Proporcionar un modelo de madurez que permita a los equipos realizar una autocomprobación y descubrir patrones y prácticas que aún no conocen.	Entelgy, Zylk, Bitergia
Elogie a los participantes	Olvidar agradecer a un contribuidor puede afectar su compromiso y disuadir a que otras personas aporten.	Después de una contribución, es importante agradecer al colaborador por su tiempo y esfuerzo.	Nike

Patrón	Problema	Solución	Caso
Puntuación de actividad del repositorio	Los contribuidores potenciales desean encontrar proyectos activos de ISS que necesiten su ayuda.	Calcule un puntaje de actividad del repositorio para cada proyecto, por ejemplo, en el Portal de ISS, para facilitar al contribuidor la decisión del proyecto al que desea contribuir.	SAP
Comité de revisión	El modelo de trabajo de ISS es una desviación radical de los enfoques más tradicionales, tanto para desarrolladores como para gerentes.	Establezca un comité como interfaz entre la iniciativa ISS y los altos directivos que participan en ella. Esto los familiarizará con la iniciativa y hará que la apoyen, ya que les otorga un cierto nivel de supervisión y control sin fomentar la microgestión (*micromanagement*).	Robert Bosch GmbH
Servicio vs. Biblioteca (librería)	Los equipos pueden ser reacios a trabajar en bases de código comunes debido a la ambigüedad sobre quién será responsable de responder ante una "caída" de ese servicio.	Es posible implementar la misma solución como servicio en entornos independientes con escalamiento de soporte separados o compartir códigos en bibliotecas.	Europace
Documentación básica	Los nuevos colaboradores de un proyecto de ISS tienen dificultades para determinar quién mantiene el proyecto, en qué trabajar y cómo contribuir.	Proporcionar documentación en archivos estándar como README.md/CONTRIBUTING.md permite un proceso de autoservicio para encontrar respuestas a las preguntas más comunes.	Europace, Paypal
Comience como un experimento	Se considera una iniciativa de ISS pero no se inicia porque la gerencia no está segura de su resultado y, por lo tanto, no está dispuesta a comprometerse con una inversión.	Comenzar la iniciativa de ISS como un experimento de tiempo limitado para facilitar el apoyo y respaldo de los gerentes.	Robert Bosch GmbH

Patrón	Problema	Solución	Caso
Toma de decisiones transparente entre equipos mediante RFC	¿Cómo dar mas transparencia a los procesos, atraer contribuidores desde instancias tempranas del proyecto y que estos aporten desde el inicio pudiendo encontrar errores de diseño?	La publicación de documentos internos de RFC (*Request For Comments*: Solicitud de Propuesta Formal) permite debatir en las primeras etapas del proceso de diseño y aumenta las posibilidades de crear soluciones con un alto grado de compromiso de todas las partes involucradas.	Google, Uber, BBC, Europace
Comité de confianza	Muchos proyectos de ISS se encontrarán en una situación en la que recibirán constantemente comentarios, funciones y correcciones de errores de los colaboradores.	Reconocer y recompensar el trabajo de los colaboradores más activos promoviéndolos al comité distribuye el control haciendo que las revisiones y aprobaciones de código sean más ágiles.	Nike, Paypal

Fuente: Propia, basada en (ISC, 2021).

El establecimiento de esta gobernanza no ahuyentará la colaboración, al contrario, la investigación de (Capraro, 2020, pág. 132) demuestra que la aumenta en comparación a los proyectos ISS con pocas o nulas restricciones.

A pesar de la lista plasmada en la Tabla 3, el ISS también presenta unos retos de alto nivel, que son propios a cualquier cambio cultural, y que requerirán vigilar constantemente como:

1. Algunas empresas ya implementan prácticas de ISS aunque no lo llaman de esta forma (Capraro, 2020, pág. 63), lo cual reduce el número de resultados durante las búsquedas bibliográficas.
2. A diferencia del OSSD el ISS no es una actividad voluntaria (Mittman, 2018, pág. 32).
3. ISS no es una metodología definida y cada implementación es hecha a la medida de cada organización (Stol & Fitzgerald, 2014, págs. 1-2).
4. No hay métodos formales para medir el impacto del ISS (Capraro, 2020, pág. 2).
5. El ISS depende que la empresa tenga un esquema de integración y despliegue continuo (CI/CD, DevOps) (Mittman, 2018, pág. 20).
6. Renuencia para contribuir: Contribuir a un activo compartido es naturalmente más complejo. Los desarrolladores generalmente no están acostumbrados a pensar y diseñar soluciones que fueran más generales que su propia línea de producto (Stol et al., 2011, págs. 31-32).
7. Cambio cultural a nivel organizacional: la empresa debe realizar acciones que fomenten el intercambio de conocimientos y aliente la recepción de contribuciones de toda la plantilla de

desarrolladores (Mittman, 2018, pág. 11) (Stol et al., 2011, pág. 33).

9.2.6 Conclusiones

La documentación consultada resulta ser la mas referenciada acerca de ISS y entrega información importante sobre los aspectos positivos y retos que tiene implementar ISS en una organización. Las empresas pueden utilizar ISS no solo por las ventajas relacionadas con mejora de calidad y reutilización tanto del código como del conocimiento dentro de su plantilla de programadores, sino también, ahora por el advenimiento del trabajo remoto como respuesta a la COVID-19, por las ventajas de comunicación en entornos remotos heredada de los esquemas de OSSD y que de acuerdo a (Capraro, 2020, págs. v y 122) era una de las características mas buscadas por la industria antes de la pandemia.

La práctica de ISS contribuye a la organización agilizando su velocidad de respuesta a las necesidades del mercado. Los proyectos podrán contar con múltiples contribuidores que ayuden a identificar y resolver *bugs* o que aporten nuevas características. Con ISS las nuevas características identificadas por "agentes externos" al proyecto pueden ser desarrolladas por ellos mismos sin pasar por el cuello de botella de esperar a que el equipo "dueño" del proyecto tenga el tiempo suficiente para priorizar la tarea en su *backlog*, refinarla y finalmente desarrollarla, lo que podría tardar algunos meses. Por otro lado, ISS puede ayudar a desarrollar en los programadores habilidades profesionales de alto valor relacionadas con cultura de la colaboración; sofisticación en la documentación y la forma de comunicar, tanto para justificar a otros sus decisiones como para rechazar contribuciones a sus proyectos.

Es importante resaltar que ISS puede ser el vehículo para lograr cambios de gran impacto en las organizaciones, ya que de acuerdo con (Oram, 2021, pág. 21) "Los programadores modernos aprenden a prosperar con el cambio. Aparte de las nuevas tecnologías y herramientas, un cambio cultural puede ayudarles a mantenerse ágiles y a actualizar sus habilidades. ISS es un paso hacia todos estos logros". Estos cambios suponen también una oportunidad para descubrir talentos dentro de la misma plantilla de desarrolladores, pues estos ya no tendrían que quedarse únicamente en el rol, producto o la tecnología para la que han sido contratados, sino que pueden demostrar su valía o desarrollar sus habilidades para aportar a la organización desde otros puntos.

El ISS no es una práctica perfecta que resuelva todos los problemas, además de los retos identificados por las referencias de este artículo, la revisión literaria también generó unas recomendaciones que, aunque no están explícitamente escritas en esos textos, es considerable listar:

- Hacer un código abierto (no confundir con OSSD), mejor desarrollado, mejor documentado, con muchos revisores, es mas complejo y de forma natural demora un poco más. Las empresas deben ser capaces de ver que esta demora es menos costosa que la reinvención de soluciones en cada equipo.

- La apertura del proyecto no indica que todo su código debe estar listo para ser reutilizado y contar con una documentación detallada, pero si exige que desde las fases de diseño se piense que componentes estarán abiertos y hacer las gestiones para que sea posible su aprovechamiento por la comunidad de desarrollo de la organización.
- Cada organización debe ser capaz de redactar un manual que permita identificar que proyectos, o partes de estos, deben tratarse como ISS o no.
- Las organizaciones deben tener claro que no es obligatorio abrir los proyectos al público; que este es un ideal de la práctica ISS.

Los hallazgos de esta revisión bibliográfica proveen valiosas pistas para que las empresas interesadas en adoptar ISS lo hagan conociendo de ante mano las ventajas y retos y esto les permita elaborar un plan aterrizado de adopción. Puesto que sus ventajas en lo concerniente a trabajo colaborativo, reutilización de software y aumento de calidad y velocidad de entrega de productos es mayor a cualquier reto, y los retos de implementar ISS son naturales a cualquier cambio organizacional, (Stol et al., 2011, pág. 12) recomienda imitar las experiencias de HP, Alcatel o Phillips Healthcare: enfocarse en aquello que funciona antes que en los retos. ISS es una práctica en constante evolución, que cada empresa debe ir adecuando en el camino, pero en este camino no está sola, pues, además de la academia, cuenta con la comunidad Inner Source Commons que documenta de forma rigurosa las experiencias y casos de éxito de ISS provenientes de todas las latitudes, donde se destacan, entre sus contribuidores, actores relevantes de la industria del software.

9.3 TRABAJOS RELACIONADOS

9.3.1 Oportunidades para la reutilización de software en un mundo incierto: del pasado a las tendencias emergentes (Capilla et al., 2019)

Título original: *Opportunities for software reuse in an uncertain world: From past to emerging trends*.

Resumen: Este *paper* hace una revisión bibliográfica sobre la historia de la reutilización de software, sus motivaciones, la situación actual y recopila tendencias futuras sobre el tema en cuestión.

9.3.2 Implementación de un repositorio para el catalogo, búsqueda y uso de componentes

software reutilizables en el desarrollo de aplicaciones web (Vargas-Fandiño et al., 2020)

Resumen: Este *paper* muestra el proceso de creación de un repositorio de elementos reutilizables en una organización, en este caso la Universidad Francisco de Paula Santander (Cúcuta, Colombia), y la posterior evaluación. Este trabajo ha servido como instrucción e inspiración para entender que la reutilización no solo es un problema de la industria, sino que también se presenta como un interesante tema de investigación académica.

9.3.3 Inicie con InnerSource, las claves para la colaboración y productividad dentro de su compañía (Oram, 2021)

Título original: *Getting started with InnerSource, keys to collaboration and productivity inside your company*.

Resumen: este booklet de la editorial O'Relly hace una serie de entrevistas a personas que hicieron parte del proceso de adopción de ISS en la empresa financiera PayPal.

9.3.4 Entendiendo la lista de chequeo de InnerSource, como catapultar la colaboración en su empresa (Bonewald & Safari, 2017)

Título original: *Understanding the InnerSource Checklist, How to Launch Collaboration Within Your Entreprise*.

Resumen: Este libro de la editorial O'Relly trae a la Directora de ISS en PayPal compartiendo un análisis del proceso de montaje y la idea de implementar una lista de chequeo que permita a otras organizaciones dar el salto.

9.3.5 Adoptando InnerSource, principio y casos de estudio (Cooper et al., 2018)

Título original: *Adopting InnerSource, Principles and Case Studies.*

Resumen: En este libro de la editorial O'Relly se presenta Deinse Cooper quien fue la encargada del montaje de ISS en Paypal y fue fundadora de ISC y al académico Klass-Jan Stol que ha conducido investigaciones sobre OSS e ISS en los últimos 10 años. En este libro se hace un

estudio concienzudo sobre los procesos de OSS, la filosofía o estilo de OSS de Apache (The Apache Way) y los procesos de implementación de ISS en Bell Laboratories, Bosch, Paypal, Europace y Ericsson; entregando no solo el relato de la estrategia llevada acabo, sino también lecciones aprendidas y consejos. Como parte de sus conclusiones se indica que ISS más que un tema de herramientas lo es de cultura, una situación similar a la que ocurre en el Laboratorio ya que tienen todas las herramientas necesarias, pero no logra impulsar dentro de los desarrolladores la reutilización.

10 METODOLOGÍA

En esta sección se presenta la metodología que se seguirá para desarrollar el proyecto de grado y cumplir con lo planteado en la sección OBJETIVOS DEL PROYECTO. Este trabajo se abordará en cuatro fases correspondientes al diagnóstico de la situación, diseño de una solución personalizada (a la medida), implementación de dicha solución y finalmente una evaluación que permita confirmar si esta solución resuelve las situaciones diagnosticadas en la primera fase.

De forma transversal a toda la metodología, se llevará a cabo la redacción del Trabajo de Grado, completándolo con los resultados de cada fase. A su vez, se mantendrán reuniones semanales con el Director de trabajo de grado para realizar los ajustes pertinentes a fin de cumplir con los objetivos académicos propuestos por la Universidad.

10.1 FASE 1: DIAGNOSTICAR

En el diagnóstico se realizarán las acciones necesarias para identificar los problemas actuales respecto a la gestión del repositorio de Servicios Comunes (RSC) que limitan la reutilización de los componentes de software contenidos en este repositorio. Las actividades a realizar en esta fase se detallan a continuación.

10.1.1 Identificar los problemas

Para diagnosticar la situación actual, se conversará con las directivas del Laboratorio profundizando en los cuestionamientos abordados en la sección DEFINICIÓN DEL PROBLEMA.

10.1.2 Priorizar los problemas identificados

Junto a las directivas del Laboratorio se realizará una priorización y posible cribado de los problemas identificados, manteniendo aquellos que dificultan la reutilización de los componentes de software almacenados en el RSC.

10.1.3 Recopilar la información respecto a los problemas priorizados

Esta recopilación de información tendrá dos tareas principales:

1. Realizar una revisión literaria sobre los problemas priorizados en la actividad anterior para comprender mejor la situación y las formas correctas de resolverla.
2. Realizar una encuesta a todos los desarrolladores sobre la reutilización de software y su percepción acerca de los problemas identificados.

10.2 FASE 2: DISEÑAR

Para diseñar un gobierno ISS personalizado que resuelva los inconvenientes detectados en FASE 1: DIAGNOSTICAR, será necesario realizar las siguientes actividades:

10.2.1 Buscar posibles soluciones

Después de identificar los problemas en la fase anterior, en cuanto a la reutilización de los componentes de software almacenados en el RSC, el Gobierno ISS incluirá formas de asegurar su solución haciéndolo “a la medida” de las necesidades del Laboratorio.

10.2.2 Desarrollar el Gobierno ISS

El desarrollo del Gobierno ISS “a la medida” es la parte más importante de este trabajo de grado ya que hay muchas definiciones por levantar, acuerdos por hacer y textos por redactar. Entre las tareas requeridas para realizar en esta actividad encuentran:

1. Definir las reglas de juego ISS adaptadas al Laboratorio.
2. Definir la licencia de software que se aplicará al RSC.

3. Proponer a Talento Humano (TTHH) un modelo de recompensas para el Gobierno ISS.
4. Crear plantilla base para nuevos proyectos ISS con todos los documentos del Gobierno ISS.
5. Redactar manuales del Gobierno ISS.

10.2.3 Realizar piloto del Gobierno ISS

Una vez desarrollado el Gobierno ISS, este se presentará a las directivas del Laboratorio y se realizarán los posibles ajustes que puedan surgir, para lo cual se llevarán a cabo las siguientes tareas:

1. Realizar un inventario de los componentes de software del RSC. Este inventario tendrá información sobre las células usuarias, la URL del repositorio y la descripción del servicio.
2. Implementar un portal que centralice el inventario de componentes. Este portal también tendrá un inventario de cada célula, sus miembros, roles y números de contacto.
3. Elegir, del inventario, un componente "semilla" para la iniciativa. Debe ser un proyecto que esté utilizando en la actualidad y que tenga pocas células usuarias, preferiblemente una, para facilitar las gestiones y la recepción de retroalimentación.
4. Establecer un comité de ISS que estará integrado por las directivas del Laboratorio, un arquitecto usuario del proyecto semilla y un desarrollador, preferiblemente de una célula distinta al arquitecto.
5. Solicitar permisos de escritura sobre el repositorio del componente "semilla".
6. Brindar charla sobre ISS a la(s) célula(s) usuaria(s) del componente semilla.
7. Ajustar el Gobierno con los comentarios de la(s) célula(s) usuaria(s) del componente semilla.
8. Implementar el Gobierno ISS en el componente semilla.
9. Determinar si se puede realizar un *merge* a la rama *master* sin re-despliegue, ya que solo habrá cambios a nivel de documentación del repositorio del componente y no de su código. Esto tendría menos impacto y reduciría los riesgos. Sin embargo, sin importar el resultado de esta tarea, este paso a la rama *master* será llamado "re-despliegue" en el resto de la metodología.
10. Re-desplegar el componente "semilla" y capturar lecciones aprendidas.
11. Evaluar con el Comité ISS la implementación del Gobierno en el componente semilla.
12. Realizar ajustes al Gobierno base con los comentarios del Comité ISS.

10.3 FASE 3: IMPLEMENTAR

En la implementación se llevarán a cabo las acciones necesarias para poner en funcionamiento un gobierno autogestionado en el RSC bajo las prácticas ISS como estrategia de Reutilización de Software para solucionar los problemas diagnosticados en la fase anterior. Los

siguientes ítems muestran en detalle las acciones a realizar y que fueron definidas tras el análisis de los documentos consultados en el apartado TRABAJOS RELACIONADOS.

10.3.1 Preparar la implementación

La implementación del Gobierno ISS puede ser crítica, ya que, aunque no se trata de modificación de código, contará con documentación (por ejemplo: acuerdos) que estarán en el repositorio de código de los componentes, lo que requerirá un despliegue para llevarlo a las ramas principales del gestor de versiones (GitHub); esto podría causar una indisponibilidad momentánea mientras se redespliega. Para hacer que la transición tenga el menor impacto posible, se deberá establecer un plan para lo cual se realizarán las siguientes acciones:

1. Priorizar con el Comité ISS los componentes del inventario a los que se implementará el Gobierno ISS. Se recomendará migrar primero los componentes con menor número de células usuarias y de menor criticidad para el negocio, pero el resultado final será una negociación con el Comité.
2. Establecer con el Comité ISS el plan de implementación. Este plan tendrá fechas y un arquitecto responsable por cada componente del RSC.
3. Solicitar permisos de escritura sobre los repositorios de todos los componentes priorizados.
4. Brindar charla sobre ISS a cada célula del Laboratorio (desarrolladores). Esta charla es importante para ganar adeptos al proyecto y reducir la fricción ante la “tarea extra” del re-despliegue.
5. Informar a todas las células por escrito sobre las fechas de implementación. En dicha comunicación se les solicitará la aprobación junto con respuestas sobre los posibles riesgos y propuestas de nuevas fechas en caso de que necesiten realizar ajustes.
6. Ajustar el plan de acuerdo con las respuestas de las células.

10.3.2 Implementar el Gobierno ISS en el resto de componentes del RSC

Implementar el Gobierno ISS en todos los componentes del RSC priorizados en el inventario de acuerdo con el plan establecido en la actividad anterior requerirá contar con un tablero público en JIRA que permita a todos monitorear la ejecución de la implementación. Para ello será necesario:

1. Establecer tablero JIRA con el seguimiento del re-despliegue de cada componente con las siguientes columnas:
    - Por agendar.
    - Agendado.

- Desplegado en ambiente de desarrollo.
- Desplegado en ambiente de pre-producción (*stagin*).
- Con plan de despliegue (*release plan*).
- Con PR a Master: o sea, con solicitud de integración al ambiente de producción: *Pull Request* (PR) a *master*.
- Con ticket de despliegue.
- Finalizado: se ha realizado el despliegue en producción.

2. Crear en el tablero JIRA tareas que representen a cada componente del RSC. Cada tarea:
   - Estará asignada al arquitecto responsable.
   - Estará atada a una épica con el nombre de la semana de despliegue para filtrar más rápidamente.
   - Tendrá hipervínculos al *release plan*, al PR a *master* y al ticket de despliegue.

10.3.3 Ejecutar re-despliegue de los componentes ajustados

Esta actividad consiste en ejecutar el re-despliegue como tal y se considerará finalizada cuando se haya ejecutado el plan establecido al inicio de esta fase.

10.4 FASE 4: VALIDAR

En la validación, se evaluará que la implementación del Gobierno ISS resuelve los problemas, detectados en la FASE 1: DIAGNOSTICAR, relacionados con la reutilización de los componentes de software almacenados en el Repositorio de Servicios Comunes (RSC). Para lograr este objetivo se llevarán a cabo las siguientes actividades.

10.4.1 Medir la efectividad del Gobierno ISS

Realizar observaciones de la efectividad del Gobierno ISS respecto a los problemas identificados será importante para comprender a los usuarios y tomar acciones correctivas.

10.4.2 Diseñar encuesta de medición de la efectividad del Gobierno ISS

La encuesta de medición de la efectividad del Gobierno ISS será la piedra angular de la validación. Para ello, se revisarán trabajos académicos similares para comprender la forma correcta de realizar una medición libre de sesgos.

10.4.3 Encuestar a todos los usuarios

Encuestar a los desarrolladores permitirá contrastar la encuesta previa al Gobierno ISS que se realizó en la Fase 1, actividad Recopilar la información respecto a los problemas priorizados en cuanto a la percepción tras la implementación del Gobierno ISS. Medir esta percepción de la efectividad del proyecto permitirá mejorarlo e incluso crear futuros proyectos para estudiar otros ángulos de la aplicación de las ISS en esta y otras organizaciones. Para llevar a cabo esta actividad se realizarán las siguientes tareas:

1. Realizar encuesta.
2. Tabular los resultados de la encuesta.
3. Comparar los resultados (actuales) con los datos previos a la implementación del Gobierno ISS, para determinar los cambios en la situación.
4. Redactar mejoras y conclusiones.

11 RECURSOS Y CRONOGRAMA

En esta sección se relacionan los recursos necesarios para ejecutar el trabajo de grado y el cronograma de actividades.

11.1 RECURSOS

11.1.1 Recursos humanos

En la Tabla 4 se relacionan los recursos humanos requeridos para la ejecución del proyecto.

Tabla 4. Relación de recursos humanos del proyecto

Rol	**Nombre**
Director	JUAN CARLOS MARTÍNEZ ARIAS Ha dirigido cerca de 27 proyectos de grados y postgrados de ingenierías y ha sido jurado de 30 comités de evaluación. El director cuenta con la siguiente formación: • Director de Posgrados Facultad de Ingeniería en Pontificia Universidad Javeriana Cali. • Maestría Disciplina académicaIngeniería con énfasis en Ing. de Sistemas y Computación - Pontificia Universidad Javeriana Cali. • Especialista Disciplina académicaAdministración con énfasis en Finanzas - Universidad ICESI. • Ingeniero de Sistemas con énfasis en Ing. de Software - Universidad Piloto de Colombia.
Codirector	N/A
Asesor	N/A

11.1.2 Otros recursos

Este trabajo de grado no requiere la adquisición adicional de recursos financieros, de hardware o de software por parte de ninguno de los involucrados. Para mayor claridad, el Laboratorio cuenta con las herramientas necesarias para llevar a cabo el trabajo de grado y por ello el enfoque de este proyecto está relacionado con la implementación de recursos sino más bien con la orquestación de estos en pos del propósito y el cambio cultural necesario para lograr los resultados esperados.

11.2 CRONOGRAMA

La Figura 2 representa el cronograma de las actividades relacionadas en el capítulo de METODOLOGÍA.

Figura 2. Cronograma del proyecto.

Tiempo		2021																				2022											
		Julio	Ago				Sep				Oct				Nov				Dic				Enero				Febrero				Marzo		
Objetivo	Actividad	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3
Diagnosticar la situación	Identificar los problemas																																
	Priorizar los problemas identificados																																
	Recopilar la información acerca de los problemas priorizados																																
Diseñar la solución	Buscar posibles soluciones																																
	Desarrollar el gobierno ISS																																
	Realizar piloto del gobierno ISS																																
Implementar la solución	Preparar la implementación																																
	Implementar el gobierno ISS a los demás componentes RSC																																
	Ejecutar re-despliegue de componentes ajustados																																
Validar la solución	Medir la efectividad del Gobierno ISS																																
	Diseñar encuesta de medición de la efectividad del Gobierno ISS																																
	Encuestar a todos los usuarios																																
Reuniones con el director																																	
Redacción del informe de proyecto																																	

Fuente: propia.

12 BIBLIOGRAFÍA

Barros-Justo, J. L., Benitti, F. B. V., & Matalonga, S. (2019). Trends in software reuse research: A tertiary study. *Computer Standards and Interfaces*, *66*, 103352. https://doi.org/10.1016/j.csi.2019.04.011

Bonewald, S., & Safari, an O. M. C. (2017). *Understanding the InnerSource Checklist: How to Launch Collaboration Within Your Enterprise*. O'Reilly Media.

Capilla, R., Gallina, B., Cetina, C., & Favaro, J. (2019). Opportunities for software reuse in an uncertain world: From past to emerging trends. *Journal of Software: Evolution and Process*, *31*(8), e2217. https://doi.org/10.1002/smr.2217

Capraro, M. (2020). *Measuring Inner Source Collaboration*. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Cooper, D., Stol, K. J., & Safari, an O. M. C. (2018). *Adopting InnerSource: Principles and Case Studies*. O'Reilly Media.

Edison, H., Carroll, N., Morgan, L., & Conboy, K. (2018). An investigation into inner source software development: Preliminary findings from a systematic literature review. *Proceedings of the 14th International Symposium on Open Collaboration, OpenSym 2018*, 1–10. https://doi.org/10.1145/3233391.3233529

IEEE. (2010). IEEE Standard for Information Technology--System and Software Life Cycle Processes--Reuse Processes. *IEEE Std 1517-2010 (Revision of IEEE Std 1517-1999)*, 1–51. https://doi.org/10.1109/IEEESTD.2010.5551093

ISC. (2021). *InnerSource Patterns*. InnerSource Commons. https://github.com/InnerSourceCommons/InnerSourcePatterns

Mittman, D. (2018). InnerSource at JPL: Collaboration around software in a science, technology, engineering and research enterprise. *Nasa - Jpl*.

Montilva, J., Arapé, N., & Colmenares, J. (2003). Desarrollo de software basado en componentes. *Actas Del IV. Congreso de Automatización y Control. Mérida, Venezuela*.

O'Reilly, T. (2000). *Keynote speaker: Tim O'Reilly---Open Source: The Model for Collaboration in the Age of the Internet*. Proceedings of the Tenth Conference on Computers, Freedom and Privacy: Challenging the Assumptions. https://web.archive.org/web/20150215185341/http://archive.oreilly.com/pub/a/oreilly/ask_tim/2000/opengl_1200.html

Oram, A. (2021). *Getting Started with InnerSource - Keys to collaboration and productivity inside your company* (2nd ed.). O'Reilly Media.

Sommerville, I. (2011). *Ingeniería del software* (9th ed.). Pearson educación.

Stol, K. J., Babar, M. A., Avgeriou, P., & Fitzgerald, B. (2011). A comparative study of challenges in integrating Open Source Software and Inner Source Software. *Information and Software Technology*, *53*(12), 1319–1336. https://doi.org/10.1016/j.infsof.2011.06.007

Stol, K. J., & Fitzgerald, B. (2015). Inner Source - Adopting Open Source Development Practices in Organizations: A Tutorial. *IEEE Software*, *32*(4), 60–67. https://doi.org/10.1109/MS.2014.77

Vargas-Fandiño, J. C., Sandoval-Ramirez, J. J., & Vera-Rivera, F. H. (2020). Implementación de un repositorio para el catálogo, búsqueda y uso de componentes software reutilizables en el desarrollo de aplicaciones web. *Revista UIS Ingenierías*, *19*(2), 11–20. https://doi.org/10.18273/revuin.v19n2-2020002