A. I. PROPUESTA DEL PROYECTO

1. Información básica del proyecto

1.1 Tipo de proyecto

SEMILLA

1.2 Tipo de investigación

Aplicada.

1.3 Unidad Ejecutora

DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA COMPUTACION

2. Lineas de investigación

DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA COMPUTACION	LPI-DICC-2023-01	Ingeniería de Software

DESCRIPCIÓN DEL PROYECTO

3. Presentación del Proyecto

3.1. Título del proyecto

Construcción de un artefacto para la detección de antipatrones en el ciclo de vida de desarrollo de software.

3.2. Resumen del proyecto

Los antipatrones en ingeniería de software representan soluciones recurrentes, pero ineficaces para resolver un problema, generando inconvenientes en las fases del ciclo de vida del desarrollo de software. Los antipatrones tienen un impacto significativo en la industria del software; su aparición puede provocar problemas en los proyectos relacionados cona la imposibilidad de refactorizar y de mantener el software involucrando un esfuerzo para entenderlo y, por ende, prefiriendo su reescritura. Aunque actualmente existen numerosos estudios relacionados con los antipatrones en el ciclo de vida del desarrollo de software, la mayoría se centra en analizar sus causas y consecuencias. Un número reducido de estudios se enfocan en la detección de code-smells que se producen en la fase de implementación de software utilizando mecanismos de IA. Sin embargo, existen antipatrones que se pueden producir en otras fases como análisis, diseño y pruebas. Este proyecto da continuidad al proyecto PIS-23-28 sobre la identificación de antipatrones, siendo que, el objetivo del presente proyecto es el de diseñar un artefacto para detección de antipatrones en las fases del ciclo de vida del desarrollo de software. De esta manera, se busca garantizar la calidad de un producto de software, al asegurar los procedimientos que conllevan el desarrollo de este.

3.3. ¿Este proyecto está relacionado con saberes ancestrales? NO

3.4. ¿La propuesta tiene potencial de transferencia tecnológica e innovación? SI

Sí, la construcción de un artefacto para la detección de antipatrones en el ciclo de vida de desarrollo de software puede considerarse un proyecto de transferencia tecnológica, debido a que cuenta con estos aspectos clave:
a) Innovación tecnológica. Hasta la actualidad no se cuenta con artefactos que permitan identificar antipatrones que mejoren significativamente los procesos existentes
b) Aplicación práctica. La transferencia tecnológica busca aplicar resultados de investigación a contextos prácticos. En este caso, el artefacto podría ser utilizado en empresas de desarrollo de software para mejorar la calidad de los sistemas y reducir costos asociados a errores o malas prácticas.
c) Colaboración entre academia e industria. Si el artefacto surge de un proyecto académico (tesis, investigación) con potencial aplicación industrial, sería un claro ejemplo de transferencia tecnológica. Especialmente si incluye formación de personal, documentación, y pruebas en entornos reales.

4. Palabras clave

Antipatrón, Artefacto, Industria de software, Detección, Ciclo de vida de desarrollo, Ingeniería de software

5. Objetivos del Proyecto

5.1. Objetivo General

Desarrollar un artefacto para la detección de antipatrones durante el ciclo de vida del desarrollo de software, con el fin de mejorar la calidad de los procesos y productos de software.

5.2. Objetivos Específicos

a. Identificar los problemas que generan los antipatrones en los proyectos de desarrollo de software.

b. Elaborar un procedimiento basado en evidencias que permita detectar antipatrones en las fases del ciclo de vida de desarrollo de software.

c. Construir un artefacto que permita la detección de antipatrones durante el ciclo de vida de desarrollo de software.

d. Evaluar el artefacto en base a un caso de estudio.

6. Hipótesis (opcional)

No aplica

7. Resultados Esperados

OBJETIVO	RESULTADO
Identificar los problemas que generan los antipatrones en los proyectos de desarrollo de software.	Clasificación de los antipatrones por fases del ciclo de vida del desarrollo de software
Elaborar un procedimiento basado en evidencias que permita detectar antipatrones en las fases del ciclo de vida de desarrollo de software.	Proceso estructurado para detectar antipatrones en cada fase del ciclo de vida del desarrollo de software

Construir un artefacto que permita la detección de antipatrones durante el ciclo de vida de desarrollo de software.	Artefacto que permita la identificación de antipatrones para cada fase del ciclo de vida del software
Evaluar el artefacto en base a un caso de estudio.	Resultados de la evaluación del artefacto

8. Impacto de la investigación

Objetivos de desarrollo sostenible

Objetivo 9: Industria, innovación e infraestructura
Objetivo 4: Educación de calidad

Justificación

Este estudio tiene como principal aporte el desarrollo de mecanismos para la detección y mitigación de antipatrones en el ciclo de vida del desarrollo de software. La propuesta busca mejorar la calidad y eficiencia de los procesos en la industria del software, reduciendo errores, sobrecostos y tiempos de entrega en cada fase del ciclo de desarrollo. De esta manera, se fortalece la infraestructura tecnológica del país, optimizando los recursos en proyectos de software y promoviendo la innovación en el sector. Al mejorar el proceso de desarrollo, se fomenta la creación de soluciones más robustas y competitivas, que pueden tener un impacto positivo en sectores clave como la salud, educación, transporte y manufactura, fundamentales para el desarrollo y modernización de Ecuador.

9. Estado del arte

ESTADO DEL ARTE

El desarrollo de software es un proceso complejo que requiere soluciones efectivas para enfrentar problemas recurrentes (Pressman, 2010). Estas soluciones suelen representarse en forma de patrones, que ofrecen estrategias reutilizables y comprobadas para crear sistemas funcionales y adaptativos. Sin embargo, su contraparte(Gamma et al., 1994), los antipatrones definidos por (Brown et al., 1998), surge cuando dichas estrategias se implementan de manera inadecuada o en contextos inapropiados, generando consecuencias negativas que afectan la calidad, la adaptabilidad y la evolución del software.

El proceso de desarrollo de software, conocido como el Ciclo de Vida de Desarrollo de Software (CVDS), se divide en una serie de fases: gestión de proyectos, análisis, diseño, implementación, pruebas y mantenimiento (Gamma et al., 1994). Durante estas etapas, la aparición de antipatrones es un riesgo latente, y su detección temprana resulta fundamental para garantizar la calidad y sostenibilidad del software. Cada etapa del Ciclo de Vida de Desarrollo de Software (CVDS) enfrenta desafíos particulares que propician la aparición de antipatrones. En el presente trabajo se analizará el estado actual de cada una de estas fases, identificando sus principales problemáticas y detallando las contribuciones específicas que se realizarán en el marco del proyecto.

La **fase de gestión de proyectos** es esencial en el desarrollo de software, ya que asegura la entrega exitosa del producto mediante la planificación, control de recursos y riesgos, y coordinación de equipos (Peters & Moreno, 2015). Sin embargo, muchos proyectos fracasan por la falta de conocimientos de los responsables o por la implementación incorrecta de prácticas de gestión (Moreno & Peters, 2016). La adopción de prácticas disfuncionales da lugar a antipatrones como el *micromanagement* o el liderazgo débil, que dificultan los procesos y generan problemas en el desarrollo (Brada & Picha, 2019).

Para identificar y gestionar los antipatrones, se han desarrollado herramientas como matrices estructurales de diseño y redes bayesianas, que ayudan a mapear las relaciones entre los elementos de estos problemas (D. L. Settas et al., 2009; D. Settas & Stamelos, 2008). Además, los antipatrones también están relacionados con las metodologías ágiles cuando estas se aplican de forma incorrecta (Flores et al., 2021). Factores humanos como las interacciones deficientes generan deuda social, lo que impacta negativamente en el proyecto y se vincula con la deuda técnica y los antipatrones (Tamburri et al., 2015).

Por ello, evitar los antipatrones es crucial para mejorar la gestión de proyectos, por lo que se han desarrollado modelos probabilísticos para detectarlos y prevenirlos. Un ejemplo es el modelo enfocado en el antipatrón *fire drill* (Picha et al., 2022). También se han estudiado marcos como *Scrum*, destacando la importancia de guiar al *Scrum Master* para que oriente al equipo y garantice el éxito del proyecto (Perkusich Mirko et al., 2013).

En el caso de la **fase de análisis de requerimientos**, un estudio demuestra que el 61% de usuarios prefieren expresar los requerimientos con lenguaje natural (Veizaga et al. 2024). De aquí nace la dificultad y necesidad de encontrar antipatrones en la especificación de requerimientos. En este mismo estudio se introduce el término *requirement smell* como problemas semánticos y sintácticos que afectan la calidad del lenguaje natural utilizado para los requisitos. Estos *requirement smells* pueden afectar el entendimiento de los requisitos escritos en lenguaje natural, en consecuencia, afectar la calidad del producto final.

Además, en estos últimos años se ha visto una gran cantidad de estudios que se enfocan en *community smells*. La definición de *community smells*, de acuerdo al estudio de (Tamburri et al., 2021) señala que estos *smells* son "…patrones subóptimos en toda la estructura organizacional y social en una comunidad de desarrollo de software que son precursores de tales desagradables eventos sociotécnicos".

Los antipatrones de análisis están relacionados con aspectos socio - técnicos Tamburri et al., 2021), por lo que se tienen indicadores como congruencia socio - técnica, que permite monitorear la salud de la comunidad de desarrollo y *stakeholders*.

Para continuar con el ciclo, la **fase de diseño de software** es un momento muy importante en el ciclo de vida del desarrollo, ya que en esta etapa se definen las bases y la estructura interna del sistema. Según (Sousa, Bigonha y Ferreira 2018), el uso inadecuado o apresurado de patrones de diseño puede llevar a la aparición de antipatrones, como *God Class* o *Duplicate Code*, cuando se concentran demasiadas funciones en un solo elemento o se repite el código de forma excesiva. Estos problemas de diseño suelen dificultar la evolución del software y aumentar los costos de mantenimiento con el paso del tiempo.

Por otra parte, la urgencia en la entrega y la falta de entendimiento profundo del problema pueden dar lugar a decisiones de diseño que, aunque parezcan correctas al inicio, generan estructuras frágiles y poco escalables. (Ouni et al. 2017) señalan que, en estas situaciones, el código tiende a presentar antipatrones como *Spaghetti Code* o *Long Parameter List*, los cuales reducen la claridad y dificultan agregar nuevas características más adelante. Así, a pesar de que al principio se busque velocidad, estas prácticas poco cuidadosas pueden tener un impacto negativo en la calidad general del proyecto. Se ha documentado ampliamente la relación entre la aparición de antipatrones en la fase de diseño y la reducción de la facilidad de mantenimiento del software. (Bán y Ferenc 2014) observan que, cuando hay demasiados antipatrones en un sistema, el esfuerzo para realizar refactorizaciones correctivas aumenta de manera notable. Estas refactorizaciones suelen ser necesarias para controlar la complejidad y adaptarse a nuevos requerimientos, pero también requieren más recursos y pueden retrasar la implementación de nuevas funcionalidades.

Adicionalmente en esta fase, (Khomh 2009) enfatiza que los componentes de software involucrados en antipatrones suelen sufrir cambios frecuentes y ser fuente de defectos repetitivos, sobre todo en casos como *Large Class* o *Spaghetti Code*. Esta dinámica de modificaciones constantes demuestra la importancia de planificar y estructurar adecuadamente el sistema desde etapas tempranas, evitando así la propagación de errores y protegiendo la calidad del software a lo largo de su ciclo de vida.

Para finalizar la **fase de implementación** en el desarrollo de software representa un punto crítico en el CVDS, ya que es donde las decisiones de diseño y análisis toman forma concreta. Según Sousa, Bigonha y Ferreira (2018), los antipatrones de implementación, como *Spaghetti Code* o *God Object*, emergen frecuentemente debido a prácticas apresuradas o una comprensión limitada de los principios de diseño. Así, la implementación no solo refleja las decisiones técnicas previas, sino que también se convierte en un escenario donde errores latentes se consolidan, incrementando la complejidad del mantenimiento posterior.

Uno de los problemas más recurrentes durante la implementación es la dificultad de identificar antipatrones y de manera eficiente y en tiempo real. Aunque herramientas como pruebas unitarias y análisis de calidad de código automatizados desempeñan un papel fundamental al proporcionar retroalimentación inmediata sobre posibles problemas (Ouni et al., 2017), estas técnicas suelen estar limitadas a la identificación de problemas sintácticos o superficiales. Esto deja a los desarrolladores en desventaja frente a antipatrones más abstractos y complejos, como el *Lava Flow* o el *Golden Hammer*.

Además, la investigación actual ha explorado métodos basados en heurísticas y métricas de calidad para detectar patrones disfuncionales. Estos enfoques ayudan a los desarrolladores a mejorar su código antes de que los problemas se propaguen. Sin embargo, carecen de estandarización universal, lo que dificulta la comparación de resultados entre herramientas y la adopción de soluciones consistentes en diferentes proyectos. En los últimos años, técnicas como la minería de repositorios de código y el uso de aprendizaje automático han mostrado su potencial para abordar estas limitaciones. Según Bán y Ferenc (2014), estas técnicas permiten identificar patrones disfuncionales en grandes bases de código, analizando tendencias y patrones recurrentes. Sin embargo, estas soluciones no están exentas de desafíos, como la necesidad de datos etiquetados y la interpretación de resultados en escenarios específicos, siendo así una tierra fértil para la innovación en la detección de los antipatrones en la fase de implementación.

A nivel global, se ha evidenciado que la detección de antipatrones es un aspecto crucial en todas las fases del desarrollo de software. Cada etapa del CVDS enfrenta desafíos específicos que, si

no se abordan a tiempo, pueden comprometer la calidad y sostenibilidad del sistema. Este proyecto, al enfocarse en la detección de antipatrones en distintas fases, tiene el potencial de aportar herramientas prácticas y conocimiento valioso que impacten tanto en el ámbito académico como en la industria del software, promoviendo prácticas más robustas y eficientes para minimizar riesgos y maximizar la calidad de los productos desarrollados.

A pesar de los numerosos estudios realizados en torno al ciclo de vida del desarrollo de software, la mayoría de ellos se centran de manera aislada en una sola fase o en actividades específicas, lo que limita su aplicabilidad en escenarios reales. Además, la falta de un enfoque estandarizado en estos estudios dificulta su validación práctica y su adopción por parte de los equipos de desarrollo. En muchos casos, las propuestas académicas se quedan en un estado teórico y no son actualizadas ni mejoradas con el paso del tiempo, volviéndose obsoletas y perdiendo relevancia ante los cambios constantes de la industria.

Por estas razones, este proyecto busca abordar de manera integral todo el ciclo de vida del desarrollo de software, prestando especial atención a la identificación de antipatrones que pueden surgir en cada una de sus fases. Para ello, se planteará un procedimiento estandarizado y claro que se pueda verificar su utilidad mediante casos de la vida real.

Este enfoque integral no solo busca llenar los vacíos que han dejado investigaciones anteriores, sino también proporcionar un prototipo que pueda evolucionar con las necesidades del sector. Al implementar un proceso estructurado para la identificación de antipatrones, se espera ofrecer una solución que sea práctica, adaptable y capaz de mantenerse vigente frente a los desafíos actuales del desarrollo de software.

Brada, P., & Picha, P. (2019, July 3). Software process anti-patterns catalogue. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3361149.3361178

Brown, W. J., Malveau, R. C., Mccormick, H. W., Thomas, I., Mowbray, J., Wiley, J., Brown, K., Malveau, C., Mccormick, K., & Mowbray, K. (1998). *AntiPatterns Refactoring Software, Architectures, and Projects in Crisis Executive Summary Overview* (T. Hudson, Ed.; 1st edition). Addison-Wesley.

Flores, G. C., Moreno, A. M., & Peters, L. (2021). Agile and Software Project Management Antipatterns: Clarifying the Partnership. *IEEE Software*, *38*(5), 39–47. https://doi.org/10.1109/MS.2020.3001030

Gamma, E., Helm, R., Ralph Johnson, & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*.

Moreno, A. M., & Peters, L. (2016). Software project management fallacies. *ICSOFT 2016 - Proceedings of the 11th International Joint Conference on Software Technologies*, *1*, 109–116. https://doi.org/10.5220/0005927001090116

Perkusich Mirko, Almeida Hyggo, & Perkusich Angelo. (2013). *A model to detect problems on scrum-based software development projects*. A.C.M.

Peters, L., & Moreno, A. M. (2015). Educating Software Engineering Managers - Revisited What Software Project Managers Need to Know Today. *Proceedings - International Conference on Software Engineering*, *2*, 353–359. https://doi.org/10.1109/ICSE.2015.168

Picha, P., Honel, S., Brada, P., Ericsson, M., Lowe, W., Wingkvist, A., & Danek, J. (2022, July 6). Process anti-pattern detection-a case study. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3551902.3551965

Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach* (P. Roig Vázquez, Ed.; V. Campos Olguín & J. Enríquez Brito, Trans.; Séptima).

Settas, D. L., Sowe, S. K., & Stamelos, I. G. (2009). Detecting similarities in antipattern ontologies using semantic social networks: Implications for software project management. *Knowledge Engineering Review*, *24*(3), 287–307. https://doi.org/10.1017/S0269888909990075

Settas, D., & Stamelos, I. (2008). *Resolving Complexity and Interdependence in Software Project Management Antipatterns Using the Dependency Structure Matrix*. http://www.problematics.com

Tamburri, D. A., Kruchten, P., Lago, P., & Vliet, H. van. (2015). Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications*, *6*(1). https://doi.org/10.1186/s13174-015-0024-6

Tamburri, D. A., Palomba, F., & Kazman, R. (2021). Exploring Community Smells in Open-Source: An Automated Approach. *IEEE Transactions on Software Engineering*, *47*(3), 630–652. https://doi.org/10.1109/TSE.2019.2901490

Veizaga, A., Shin, S. Y., & Briand, L. C. (2024). Automated Smell Detection and Recommendation in Natural Language Requirements. *IEEE Transactions on Software Engineering*, *50*(4), 695–720. https://doi.org/10.1109/TSE.2024.3361033

Bán, D., & Ferenc, R. (2014). Recognizing Antipatterns and Analyzing Their Effects on Software Maintainability. En ICCSA 2014 (pp. 337–352). LNCS 8583. Springer International Publishing. https://doi.org/10.1007/978-3-319-09156-3_25

Khomh, F. (2009). SQUAD: Software Quality Understanding through the Analysis of Design. En 2009 16th Working Conference on Reverse Engineering (WCRE) (pp. 303–306). Lille, France. https://doi.org/10.1109/WCRE.2009.22

Ouni, A., Kessentini, M., Ó Cinnéide, M., Sahraoui, H., Deb, K., & Inoue, K. (2017). MORE: A Multi-Objective Refactoring Recommendation Approach to Introducing Design Patterns and Fixing Code Smells. Journal of Software: Evolution and Process, 29(2), e1843. https://doi.org/10.1002/smr.1843

Sousa, B. L., Bigonha, M. A. S., & Ferreira, K. A. M. (2018). A Systematic Literature Mapping on the Relationship Between Design Patterns and Bad Smells. En Proceedings of the 33rd International Conference on Software Engineering (ICSE) (pp. 1528–1535). Honolulu, HI, USA. https://doi.org/10.1145/3167132.3167295

- *Describir el marco referencial relacionado al campo y área del tema del proyecto*
- *Máximo tres carillas (excluyendo referencia bibliográficas)*
- *Incluir referencias bibliográficas (Normas APA)*

A. II. DETALLES DEL PROYECTO

10. Descripción detallada de proyecto incluida su metodología

ESCUELA POLITÉCNICA NACIONAL
VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y VINCULACIÓN

1. DESCRIPCIÓN DETALLADA DEL PROYECTO, INCLUIDO METODOLOGÍA

Este proyecto tiene como objetivo principal el desarrollo de un artefacto para la detección de antipatrones a lo largo del ciclo de vida del desarrollo de software, el mismo que da continuidad al Proyecto Semilla PIS-23-28 que abordó la identificación y comprensión de antipatrones.

"Los antipatrones son soluciones comúnmente aplicadas a problemas recurrentes que, en lugar de resolverlos, generan consecuencias negativas en los procesos de desarrollo. Estos pueden surgir por falta de conocimiento, experiencia insuficiente o por la aplicación incorrecta de patrones válidos en contextos inapropiados" (Brown et al., 1998).

El proyecto se enfocará en estudiar los problemas recurrentes en el desarrollo de software que dan origen a los antipatrones, con el objetivo de comprender cómo surgen y en qué contextos aparecen. A partir de este análisis, se busca proponer un procedimiento que permita identificar de manera efectiva los antipatrones en cada fase del ciclo de vida del desarrollo de software, asegurando que dicha identificación se realice de forma sistemática y clara. Finalmente, el artefacto resultante será evaluado mediante un caso de estudio, con el fin de demostrar su utilidad práctica y validar cómo su aplicación puede contribuir significativamente a mejorar los procesos de desarrollo.

Metodología de investigación

El proyecto seguirá el enfoque de Design Science Research (DSR), dado que este método permite desarrollar soluciones prácticas mediante la creación y validación de artefactos que aborden problemas reales en contextos específicos (Dresch et al., 2015). En donde, la construcción y aplicación de un artefacto ayudará a resolver la problemática central, (Wieringa, 2014) indica que "El término artefacto es fundamental para la investigación científica del diseño y se utiliza para describir algo que es artificial o construido por humanos, en contraposición a algo que ocurre de forma natural" (p.106).

El DSR será implementado debido a su capacidad para combinar rigor científico con aplicabilidad práctica, facilitando la identificación de antipatrones en cada fase del ciclo de vida de desarrollo de software. La Figura 1 representa el proceso del DSR de una forma resumida.

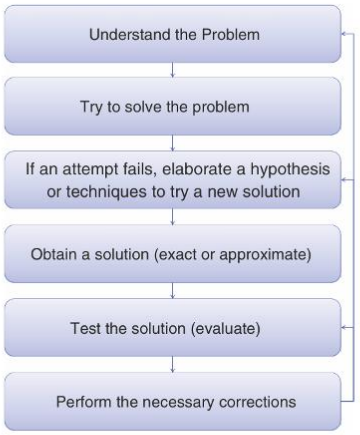

Figura 1. Pasos para realizar un proceso de investigación en DSR. Adaptado de (Dresch et al., 2015)

ESCUELA POLITÉCNICA NACIONAL
VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y VINCULACIÓN

A. Diagnóstico

Se identificará la problemática y se establecerá la necesidad de investigar antipatrones en las distintas fases del ciclo de vida de desarrollo. Esta fase es clave para delimitar el alcance del estudio y justificar su relevancia. Se empleará una revisión sistemática de literatura utilizando el marco de Kitchenham para garantizar una base sólida de conocimiento y asegurar la validez de los hallazgos (Kitchenham et al., 2015).

- **Definición del marco conceptual**

Se identificarán los conceptos clave relacionados con los antipatrones y se definirán las fases del ciclo de vida de desarrollo de software a analizar.

- **Formulación de preguntas de investigación**

Se definirán preguntas para dirigir los esfuerzos del estudio y encaminar el proyecto hacia los resultados deseados.

- **Revisión sistemática de literatura con el marco de Kitchenham**

Se empleará una revisión sistemática de literatura utilizando el marco de Kitchenham para garantizar una base sólida de conocimiento y asegurar la validez de los hallazgos (Kitchenham et al., 2015).

  - **Estrategia:** Se establecerán cadenas de búsqueda y bases de datos relevantes. Posteriormente, se definirán criterios de inclusión y exclusión.
  - **Ejecución:** Se buscarán y extraerán estudios relevantes. Posteriormente, se eliminarán duplicados y se seleccionarán los estudios relevantes. Por último, se obtendrá un conjunto final de estudios seleccionados.

B. Planificación

Se diseñará un plan de acción que permita llevar a cabo la investigación y garantizar que cada fase del ciclo de vida sea adecuadamente cubierta. Esto es necesario para asegurar la viabilidad y efectividad del proyecto.

- **Análisis de la información recopilada.**

El análisis de la información recopilada es una etapa crítica para garantizar que los fundamentos teóricos y prácticos del proyecto estén alineados con los objetivos establecidos. Esta actividad implica la revisión detallada de estudios previos, artículos académicos, marcos metodológicos y casos prácticos relacionados con el ciclo de vida del desarrollo de software. Mediante este análisis, se busca identificar antipatrones relevantes y recurrentes que puedan ser consideradas en el diseño del artefacto. El análisis también permitirá detectar áreas donde la información es insuficiente o inconsistente, facilitando la identificación de temas que requieren mayor profundización.

- **Determinación de brechas en la información recopilada.**

El análisis de brechas será una actividad fundamental dentro del desarrollo del proyecto, ya que permitirá evidenciar los diversos enfoques utilizados en investigaciones previas y las problemáticas asociadas a los mismos. Esto permitirá determinar si la falta de un enfoque estandarizado en dichas investigaciones dificulta su validación práctica, lo que podría ocasionar que muchas de las propuestas académicas permanezcan en un estado teórico sin ser actualizadas ni mejoradas con el paso del tiempo, perdiendo relevancia ante los constantes cambios en la industria del software. Este análisis contribuirá a identificar vacíos de información y áreas que requieren una mayor atención para garantizar que los resultados del proyecto sean aplicables y útiles en contextos reales.

- **Diseño del cronograma:**
  - **Recursos y tiempos:** Identificación de recursos y tiempos necesarios.

ESCUELA POLITÉCNICA NACIONAL
VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y VINCULACIÓN

- **Actividades e hitos para conseguir:** Planificación de actividades clave del proyecto y definición de hitos principales para evaluar avances.
- **Seguimiento de las actividades:** Implementación de revisiones periódicas para evaluar el progreso y realizar ajustes de forma óptima.

C. Construcción

Se desarrollará un artefacto que permitirá identificar los antipatrones en cada fase del ciclo de vida de desarrollo de software. Esta etapa es esencial para generar un instrumento práctico que facilite las aplicaciones del artefacto en fases posteriores. El proceso de construcción del artefacto incluye los siguientes pasos clave:

- **Definición del tipo de artefacto a construir**

En este paso se determina el tipo de artefacto que se va a desarrollar, definiendo su objetivo principal y el alcance que tendrá en la identificación de antipatrones. El artefacto puede tomar la forma de un modelo, una guía práctica, una herramienta automatizada o un marco conceptual, dependiendo de las necesidades del proyecto.

- **Desarrollo y documentación del artefacto**

Este paso implica la creación del artefacto en sí, asegurándose de que cumpla con los requisitos definidos previamente. Se utilizarán principios de diseño basados en el DSR para asegurar que el artefacto sea útil y efectivo en contextos reales.
La documentación del artefacto incluirá detalles sobre su estructura, propósito, funcionamiento y recomendaciones para su aplicación práctica.

D. Evaluación

Se evaluará la utilidad y aplicabilidad del artefacto en un caso de estudio. Esta fase es clave para asegurar la eficacia del artefacto en contextos prácticos.

- **Prueba del artefacto:** Se aplicará la guía y el artefacto en un caso de estudio.
- **Análisis de resultados:** Se compararán los resultados obtenidos con los objetivos propuestos.

E. Aprendizaje

Se extraerán lecciones aprendidas y se generará nuevo conocimiento sobre la identificación de antipatrones. Esta fase asegura que los hallazgos sean aprovechados para futuras investigaciones y aplicaciones.

- **Documentación de hallazgos:** Se registrarán los antipatrones detectados y las soluciones aplicadas.
- **Divulgación de resultados:** Se publicará un informe final y se compartirán los hallazgos con la comunidad académica y profesional.

Brown, W. J., Malveau, R. C., Mccormick, H. W., Thomas, I., Mowbray, J., Wiley, J., Brown, K., Malveau, C., Mccormick, K., & Mowbray, K. (1998). *AntiPatterns Refactoring Software, Architectures, and Projects in Crisis Executive Summary Overview*.

Dresch, A., Daniel, ·, Lacerda, P., Antônio, J., & Antunes, V. (2015). *Design Science Research A Method for Science and Technology Advancement*.

Kitchenham, B. A., Budgen, D., & Brereton, P. (2015). *Evidence-based software engineering and systematic reviews* (Vol. 4). CRC press.

ESCUELA POLITÉCNICA NACIONAL
VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y VINCULACIÓN

Wieringa, R. (2014). *Design Science Methodology: Principles and Practice Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE*. ACM Digital Library. https://doi.org/doi:10.1145/1810295.1810446

11. Descripción de equipos disponibles

EQUIPO	DEPARTAMENTO	UBICACIÓN
HP ProBook 450 G10	DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA COMPUTACION	Laboratorio ADN Software, edificio 20
HP ProBook 450 G10	DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA COMPUTACION	Laboratorio ADN Software, edificio 20
HP ProBook 450 G10	DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA COMPUTACION	Laboratorio ADN Software, edificio 20

12. Vinculación con Posgrados

No aplica.

13. Justificación del presupuesto para la ejecución del proyecto

Item presupuesto	Contribución	Justificación
Contratación de ayudantes de investigación	Contribuye directamente al objetivo1, objetivo2 y objetivo3	Los ayudantes de investigación son requeridos para poder realizar el diagnóstico de la situación actual además de la construcción del artefacto generado en este proyecto de tesis.
Ponencias en el exterior, capacitaciones y/o visitas técnicas	Se plantea presentar dos artículos que contemplen los objetivos del proyecto planteado.	Como parte de los entregables del proyecto se plantean dos artículos indexados en Scopus, por lo que se plantea la presentación en dos conferencias relacionadas al área de Software.
Pago de inscripciones	Se plantea presentar dos artículos que contemplen los objetivos del proyecto planteado.	Se plantea la presentación en dos conferencias relacionadas al área de Software, por lo que se estiman dos valores por pago de inscripción.

14. Fondos Adicionales

No aplica.

B. DATOS INFORMATIVOS DEL EQUIPO

1. Infomación del Director, Codirector, Colaboradores y Colaboradores Técnicos

Nombres y Apellidos	Cédula	HSS*	Unidad Académica	Rol	Cargo	Título de Mayor Nivel
PAMELA CATHERINE FLORES NARANJO	1716270838	6	DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA	Director	PROFESOR AGREGADO A TIEMPO COMPLETO (NIVEL 2, GRADO 4)	Doctora en Software y Sistemas
CARLOS EDUARDO ANCHUNDIA VALENCIA	1308669496	8	DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA	Codirector	PROFESOR AGREGADO A TIEMPO COMPLETO (NIVEL 1, GRADO 3)	Doctor en informática
EVELYN MARCELA MOSQUERA ESPINOSA	1724375892	3	DEPARTAMENTO DE INFORMATICA Y CIENCIAS DE LA	Colaborador	PROFESOR OCASIONAL A TIEMPO COMPLETO	MAGISTER EN SOFTWARE MENCIÓN EN CALIDAD

*HSS =Horas Semana Semestre: Es el número de horas que se dedica por semana a la investigación. Este número de horas se mantiene para todo el semestre

2. Investigaciones Previas del equipo

¿El Director tiene publicaciones en SCOPUS de los últimos 5 años? SI

¿El Codirector tiene publicaciones en SCOPUS de los últimos 5 años? SI

C. INFORMACIÓN ADICIONAL

1. Información de presentación previa de la propuesta

Tipo de proyecto	Año de postulación
No aplica	No aplica.

¿Este proyecto proviene de algún proyecto de vinculación existente? NO

2. Sugerencia de potenciales revisores

NOMBRES	APELLIDOS	INSTITUCION	EMAIL
Sonia	Pamplona	Universidad Complutense de Madrid	tesis.spamplona@gmail.com
Rigoberto	Fonseca	Universidad Yachay Tech	rfonseca@yachaytech.edu.ec

3. Exclusión de potenciales revisores

No aplica.

CRONOGRAMA DE ACTIVIDADES

NOMBRE OBJETIVO	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
AÑO 1												
Objetivo Específico: Identificar los problemas que generan los antipatrones en los proyectos de desarrollo de software.												
Actividad: Revisión sistemática de literatura con el	X	X										
Actividad: Clasificación de los problemas encontrados		X	X									
Actividad: Mapeo de los antipatrones hacia los			X									
Objetivo Específico: Elaborar un procedimiento basado en evidencias que permita detectar antipatrones en las fases del ciclo de vida de desarrollo de software.												
Actividad: Definición del marco teórico para la detección			X	X								
Actividad: Elaboración y documentación del				X	X	X						
Objetivo Específico: Construir un artefacto que permita la detección de antipatrones durante el ciclo de vida de desarrollo de software.												
Actividad: Definición del tipo de artefacto a construir						X						
Actividad: Desarrollo y documentación del artefacto						X	X	X	X	X	X	X
Objetivo Específico: Evaluar el artefacto en base a un caso de estudio.												
Actividad: Definición del caso de estudio												X
Actividad: Uso del artefacto para detectar a los												X
Actividad: Retroalimentación e informe de los resultados												
AÑO 2												
Objetivo Específico: Identificar los problemas que generan los antipatrones en los proyectos de desarrollo de software.												
Actividad: Revisión sistemática de literatura con el												
Actividad: Clasificación de los problemas encontrados												
Actividad: Mapeo de los antipatrones hacia los												
Objetivo Específico: Elaborar un procedimiento basado en evidencias que permita detectar antipatrones en las fases del ciclo de vida de desarrollo de software.												
Actividad: Definición del marco teórico para la detección												
Actividad: Elaboración y documentación del												
Objetivo Específico: Construir un artefacto que permita la detección de antipatrones durante el ciclo de vida de desarrollo de software.												
Actividad: Definición del tipo de artefacto a construir												
Actividad: Desarrollo y documentación del artefacto												
Objetivo Específico: Evaluar el artefacto en base a un caso de estudio.												
Actividad: Definición del caso de estudio												
Actividad: Uso del artefacto para detectar a los	X	X	X	X	X	X						
Actividad: Retroalimentación e informe de los resultados					X	X						

PRESUPUESTO

Lista de items	Cantidad	Unidad	Precio Unitario	Precio Total
SEMESTRE 1				
1. Contratación de servicios profesionales				
			$ 0.0	$ 0.0
Subtotal 1				$ 0.0
2. Contratación de ayudantes de investigación				
Ayudante de investigación 2	6	mes	$ 234.3	$ 1405.8
Ayudante de investigación 1	6	mes	$ 234.3	$ 1405.8
Subtotal 2				$ 2811.6
3. Maquinaria y equipo especializado				
			$ 0.0	$ 0.0
Subtotal 3				$ 0.0
4. Equipo informático				
			$ 0.0	$ 0.0
Subtotal 4				$ 0.0
5. Insumos y Reactivos				
			$ 0.0	$ 0.0
Subtotal 5				$ 0.0
6. Análisis de laboratorio				
			$ 0.0	$ 0.0
Subtotal 6				$ 0.0
7. Literatura especializada				
No se ha seleccionado un item	1	Libro	$ 230.0	$ 230.0
Subtotal 7				$ 230.0
8. Salidas de campo y de muestreo				
			$ 0.0	$ 0.0
Subtotal 8				$ 0.0
9. Ponencias nacionales, capacitaciones y/o visitas				
			$ 0.0	$ 0.0
Subtotal 9				$ 0.0
10. Ponencias en el exterior, capacitaciones y/o				
Viaticos al exterior	1	dólares	$ 1500.0	$ 1500.0
Pasajes al exterior	1	dólares	$ 1800.0	$ 1800.0
Subtotal 10				$ 3300.0
11. Pago de inscripciones				
Pago de inscripciones al exterior	1	dólares	$ 800.0	$ 800.0
Subtotal 11				$ 800.0
12. Atención a delegados / Investigadores				
			$ 0.0	$ 0.0
Subtotal 12				$ 0.0
13. Pago de publicaciones, subscripciones				

Lista de items	Cantidad	Unidad	Precio Unitario	Precio Total
			$ 0.0	$ 0.0
Subtotal 13				$ 0.0
TOTAL SEMESTRE 1				$ 7141.6
SEMESTRE 2				
1. Contratación de servicios profesionales				
			$ 0.0	$ 0.0
Subtotal 1				$ 0.0
2. Contratación de ayudantes de investigación				
Ayudante de investigación 1	6	mes	$ 234.3	$ 1405.8
Ayudante de investigación 2	6	mes	$ 234.3	$ 1405.8
Subtotal 2				$ 2811.6
3. Maquinaria y equipo especializado				
			$ 0.0	$ 0.0
Subtotal 3				$ 0.0
4. Equipo informático				
			$ 0.0	$ 0.0
Subtotal 4				$ 0.0
5. Insumos y Reactivos				
			$ 0.0	$ 0.0
Subtotal 5				$ 0.0
6. Análisis de laboratorio				
			$ 0.0	$ 0.0
Subtotal 6				$ 0.0
7. Literatura especializada				
			$ 0.0	$ 0.0
Subtotal 7				$ 0.0
8. Salidas de campo y de muestreo				
			$ 0.0	$ 0.0
Subtotal 8				$ 0.0
9. Ponencias nacionales, capacitaciones y/o visitas				
			$ 0.0	$ 0.0
Subtotal 9				$ 0.0
10. Ponencias en el exterior, capacitaciones y/o				
Viaticos al exterior	1	d	$ 1500.0	$ 1500.0
Pasajes al exterior	1	d	$ 1800.0	$ 1800.0
Subtotal 10				$ 3300.0
11. Pago de inscripciones				
Pago de inscripciones al exterior	1	d	$ 800.0	$ 800.0
Subtotal 11				$ 800.0
12. Atención a delegados / Investigadores				
			$ 0.0	$ 0.0

Lista de items	Cantidad	Unidad	Precio Unitario	Precio Total
Subtotal 12				$ 0.0
13. Pago de publicaciones, subscripciones				
			$ 0.0	$ 0.0
Subtotal 13				$ 0.0
TOTAL SEMESTRE 2				$ 6911.6
SEMESTRE 3				
1. Contratación de servicios profesionales				
			$ 0.0	$ 0.0
Subtotal 1				$ 0.0
2. Contratación de ayudantes de investigación				
Ayudante de investigación 1	4	mes	$ 234.3	$ 937.2
Subtotal 2				$ 937.2
3. Maquinaria y equipo especializado				
			$ 0.0	$ 0.0
Subtotal 3				$ 0.0
4. Equipo informático				
			$ 0.0	$ 0.0
Subtotal 4				$ 0.0
5. Insumos y Reactivos				
			$ 0.0	$ 0.0
Subtotal 5				$ 0.0
6. Análisis de laboratorio				
			$ 0.0	$ 0.0
Subtotal 6				$ 0.0
7. Literatura especializada				
			$ 0.0	$ 0.0
Subtotal 7				$ 0.0
8. Salidas de campo y de muestreo				
			$ 0.0	$ 0.0
Subtotal 8				$ 0.0
9. Ponencias nacionales, capacitaciones y/o visitas				
			$ 0.0	$ 0.0
Subtotal 9				$ 0.0
10. Ponencias en el exterior, capacitaciones y/o				
			$ 0.0	$ 0.0
Subtotal 10				$ 0.0
11. Pago de inscripciones				
			$ 0.0	$ 0.0
Subtotal 11				$ 0.0
12. Atención a delegados / Investigadores				
			$ 0.0	$ 0.0

Lista de items	Cantidad	Unidad	Precio Unitario	Precio Total
Subtotal 12				$ 0.0
13. Pago de publicaciones, subscripciones				
			$ 0.0	$ 0.0
Subtotal 13				$ 0.0
TOTAL SEMESTRE 3				$ 937.2
TOTAL PROYECTO				$ 14990.4

D. DECLARACIÓN FINAL

Declaración del codirector de proyecto

Declaro que conozco que la normativa de proyectos establece que, en caso de ausencia del Director del proyecto en ejecución, la Dirección de Investigación notificará el cambio de Director a favor del Codirector.

Yo, CARLOS EDUARDO ANCHUNDIA VALENCIA

Declaro que acepto los puntos descritos en esta sección.

Declaración del director de proyecto

El equipo de investigadores, representado por el Director del Proyecto declara lo siguiente:

Que el presente proyecto es una creación original de mi autoría y del equipo de investigadores, y por tanto asumimos la completa responsabilidad legal en caso de que un tercero alegue la titularidad de los derechos intelectuales del proyecto, exonerando a la EPN de cualquier acción legal que se derive por esta causa.

Que aceptamos conocer y cumplir con la normativa vigente para la gestión de proyectos y las Bases para la Convocatoria para la presentación de Proyectos de Investigación, Vinculación y Transferencia Tecnológica con financiamiento 2024.

Que el presente proyecto no ha sido ejecutado, ni está actualmente siendo evaluado en ninguna convocatoria de otra institución pública o privada. El incumplimiento será causal para que el proyecto no sea tomado en consideración.

Que si el proyecto genera algún producto o procedimiento susceptible de obtener derechos de propiedad intelectual, de los cuales se deriven beneficios, aceptamos que éstos serán compartidos entre los investigadores y la institución o las instituciones participantes en el proyecto, conforme a lo establecido en el COESC.

Que el equipo de investigadores y/o instituciones participantes se comprometen a mantener la confidencialidad de la información si ésta podría ser susceptible de protección por patentes, y solicitar la valoración de propiedad intelectual respectiva previa a cualquier publicación o difusión.

Que para el caso de derechos de autor otorgamos una licencia de uso exclusivo con fines académicos para la o las instituciones participantes en el proyecto.

Yo, PAMELA CATHERINE FLORES NARANJO

Declaro que acepto los puntos descritos en esta sección.

Firmado electrónicamente por:
PAMELA CATHERINE FLORES NARANJO