Ingeniería y
Competitividad
Vol 26 (1) 2024; doi: 10.25100/iyc.v26i1.12840

Vol 26 (1)

ISSN
0123-3033
e- 2027-8284

Artículo de investigación

Ingeniería inversa aplicada en función del contexto

Martín E. Monroy[1] Martin Pinzger[2] José L. Arciniegas[3]

[1]Universidad de Cartagena, Programa de Ingeniería de Sistemas, Cartagena, Colombia
[2]Alpen-Adria-Universität, Software Engineering Research Group, Klagenfurt, Austria,
[3]Universidad del Cauca, Departamento de Telemática, Popayán, Colombia.

Resumen

La ingeniería inversa se realiza en múltiples contextos. Cada contexto está definido por un grupo de participantes, un conjunto de recursos y situaciones que se encuentran dentro de un ámbito específico. Existen múltiples propuestas para realizar ingeniería inversa, sin embargo, todas asumen que se hace en el contexto de la producción de software. El objetivo de este trabajo es proponer un referente para recuperar el diseño de productos software, que pueda ser utilizado en diferentes contextos. Se hizo un análisis comparativo de los enfoques de ingeniería inversa utilizando la técnica de coincidencia de patrones. Para validar los resultados obtenidos se realizó un estudio de caso en dos contextos diferentes, el primero en un contexto de educación para apoyar un proceso de enseñanza aprendizaje y el segundo en un contexto de producción para recuperar el diseño de un producto software. Se definió un marco de referencia conformado por un sistema conceptual descriptivo y un conjunto de elementos instrumentales de tipo operativo, que guía el proceso de recuperación del diseño de productos software, ajustándose a las características del contexto en el que se realiza esta actividad. Se concluye que el marco de referencia definido, ofrece un nuevo enfoque para la recuperación del diseño de productos software, porque involucra el contexto en el que se realiza el proceso y oculta su complejidad a los participantes que no son expertos en ingeniería inversa.

Palabras clave: Educación, Ingeniería inversa, Marco de referencia, Producción de software, Recuperación del diseño.

¿Cómo citar?
Monroy, M.E., Pinzger, M., Arciniegas, J.L. Applied Reverse Engineering in Context. Ingeniería y Competitividad, 2024, 26(1); e-22112840.

https://doi.org/10.25100/iyc.v26i1.12840

Recibido: 2-03-23
Aceptado: 12-02-23

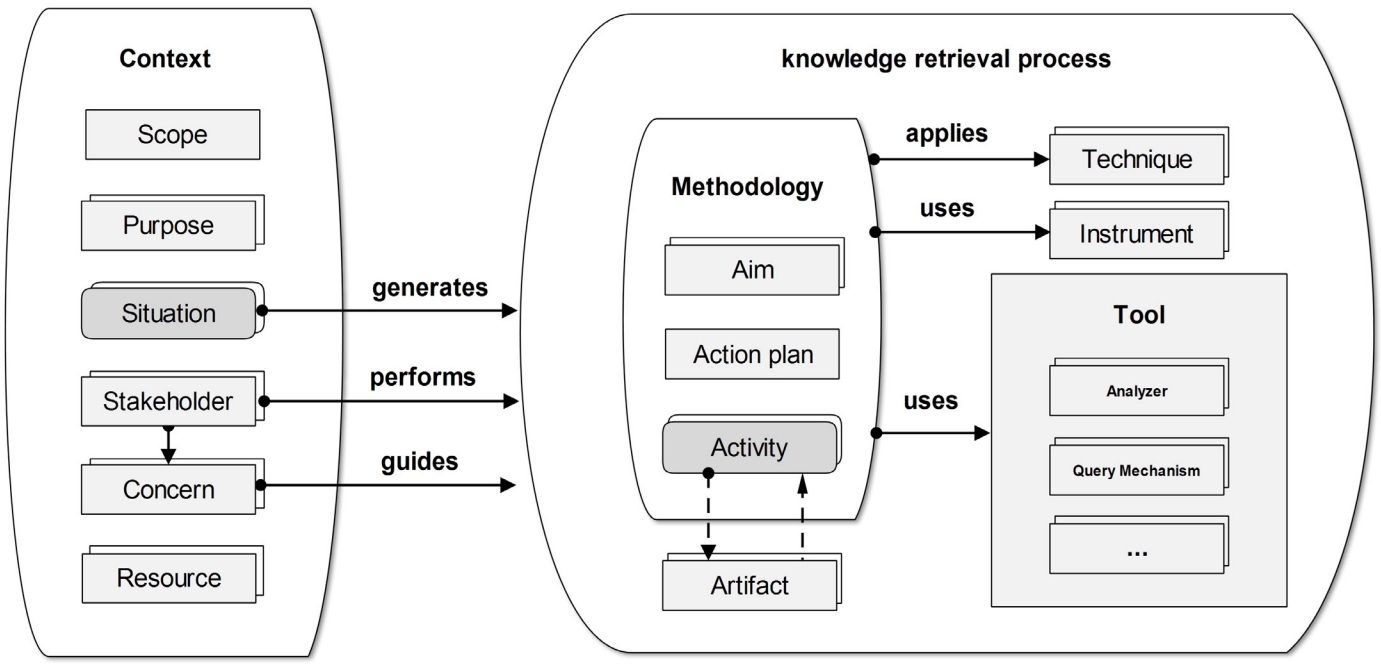
Correspondencia:
mmonroyr@unicartagena.edu.co

¿Por qué se llevó a cabo?
La ingeniería inversa se aplica en múltiples contextos. Cada contexto está definido por un grupo de actores, un conjunto de recursos y situaciones dentro de un ámbito específico. Existen diversos enfoques para la ingeniería inversa, sin embargo, todos asumen que se realiza en el contexto de la producción de software. El objetivo de este trabajo es definir un enfoque para recuperar el diseño de software de acuerdo con las situaciones del contexto y las preocupaciones de las partes interesadas.

¿Cuáles fueron los resultados más relevantes?
Se definió un marco, que incluye un sistema conceptual descriptivo y un conjunto de elementos instrumentales de tipo operativo, que sirve para guiar el proceso de recuperación del diseño del producto software, en función del contexto en el que se realiza esta actividad.

¿Qué aportan estos resultados?
Un modelo conceptual para recuperar el diseño y analizar la documentación recuperada, basado en los estándares ISO/IEC/IEEE 42010, ISO/IEC 19506 y UML.
Una metodología que guía el proceso de ingeniería inversa, según el contexto donde surge la necesidad, y que permite a los interesados obtener resultados relevantes para ellos.
La caracterización de los contextos de uso de la ingeniería inversa.
La inclusión del análisis de contexto en el proceso de ingeniería inversa, representado por el alcance, situaciones, recursos, stakeholders y sus preocupaciones.
El diseño y construcción de un prototipo de mecanismo de consulta para soportar el análisis de la documentación recuperada.

Graphical Abstract



doi: 10.25100/iyc.v26i1.12840

Introducción

Existen múltiples enfoques para recuperar el diseño de productos software, clasificados en: técnicas (1-4), herramientas (6-10), métodos (11-13), y marcos de referencia (5, 14-18). Las técnicas detallan la forma como se debe cumplir una actividad específica del proceso de ingeniería inversa. Los métodos se centran en el proceso y definen el orden lógico de las actividades requeridas para recuperar el diseño. Las herramientas implementan técnicas para automatizar el proceso y los marcos de referencia comprenden métodos, técnicas y herramientas. Los enfoques identificados aplican las actividades definidas por Tilley et al. (19): extracción de datos, organización del conocimiento y exploración de la información. Algunos se centran en el análisis de productos de software (11), otros en apoyar la evolución (12,20), en reconstruir el diseño (12-16) o en el re-diseño del producto (17). Todos están orientados al proceso de ingeniería de software, para realizar mantenimiento, control de calidad, re-diseño y reutilización de activos (21).

La ingeniería inversa es un campo de investigación que ha llamado la atención de los investigadores en los últimos años desde múltiples perspectivas (12,22,23). Su objetivo principal es reconstruir el conocimiento implícito en un sistema, representado en las vistas arquitectónicas de un producto de software (5). La arquitectura de software es la organización fundamental de un sistema representada en sus componentes, las relaciones que existen entre ellos y su entorno, y los principios que rigen su diseño y evolución (24). Una vista arquitectónica expresa la arquitectura del sistema a través de modelos, construidos desde la perspectiva de las restricciones definidas por el grupo de interesados (24). Hay varios enfoques que especifican las vistas arquitectónicas y los puntos de vista del software. Esto implica seleccionar un enfoque para representar las vistas del sistema, que se ajuste a la situación y el contexto, lo cual aumenta la complejidad del proceso de recuperación, porque obliga a tener en cuenta las situaciones, los recursos disponibles, las partes interesadas, sus limitaciones y propósitos.

El estudio de la ingeniería inversa se ha centrado en la definición de técnicas que se especializan en la reconstrucción del diseño del sistema (15,17), en la revisión (25), la evolución y el análisis del producto (11), o en lograr uno o más de estos propósitos al mismo tiempo (12). Estos enfoques se centran más en las actividades específicas de la ingeniería inversa, que en el proceso en sí. El proceso canónico definido por Tilley et al. (19) ha sido refinado por varias propuestas. Por ejemplo, Symphony (18) se enfoca en aspectos generales de la recuperación de arquitecturas y en cómo seleccionar las vistas a reconstruir. Cacophony (17) define un proceso genérico impulsado por meta-modelos. Kerdoudi et al. (13) definen un proceso de recuperación del diseño para líneas de productos. Tamburri y Kazman (12) establecen un método general para la recuperación de arquitecturas. Stormer (11) se centra en la recuperación de la arquitectura desde la perspectiva de los atributos de calidad. Bruneliere et al. (5) definen un marco basado en KDM (Knowledge Discovery Metamodel) orientado a la modernización del producto. Tamburri y Kazman. (12) establecen un proceso para extraer tres vistas del sistema para facilitar el mantenimiento; y Schmitt et al. (15) proponen un marco de referencia que incluye un conjunto de principios y procesos para recuperar arquitecturas. Cada uno de estos enfoques define un conjunto de actividades para recuperar el diseño y establece el orden lógico bajo el cual se deben realizar.

Por otra parte, Stormer (11) declara como contextos de la ingeniería inversa: la mejora de la comprensión de la arquitectura de sistemas software heredados, la mejora de la arquitectura en sí misma, la evaluación de las características de los atributos de calidad del sistema, la mejora del diseño del sistema. Todas estas actividades son típicas del

doi: 10.25100/iyc.v26i1.12840

proceso de construcción de software. Para Favre (17) e Ibrahim et al. (14) el contexto se refiere al entorno físico o situación de la aplicación, mientras que para Tamburri y Kazman (12) corresponde a las circunstancias de desarrollo del producto software. Solo Van Deursen et al. (18) contemplan aspectos relacionados con la disponibilidad de recursos y los intereses de los participantes a la hora de recuperar el diseño del producto software. Sin embargo, ninguna de las propuestas identificadas en la revisión de la literatura tiene en cuenta explícitamente las situaciones, las características de los participantes, el ámbito ni el propósito del contexto donde se cumple el proceso de ingeniería inversa. Esto crea una brecha que dificulta la recuperación del diseño en contextos diferentes a la producción de software, lo que se convierte en el principal aporte de este trabajo de investigación.

La ingeniería inversa es una actividad que también se realiza en otros contextos como la educación (26-28), la seguridad informática (25, 29-31) y la informática forense (32), además de la producción de software. Cada uno de estos contextos está determinado por un ámbito, situaciones, recursos, propósitos, y participantes que tienen intereses y características particulares (23). En este escenario surgen los siguientes interrogantes cuando se realiza ingeniería inversa en contextos distintos a la producción de software: ¿Cómo orientar el proceso de ingeniería inversa para que sea pertinente con las características de los participantes y sus intereses? ¿Cómo orientar el proceso de ingeniería inversa en función de las situaciones presentes en el contexto donde se realiza? ¿Cómo orientar el proceso de ingeniería inversa en función del ámbito y del propósito del contexto donde se realiza? y ¿Cómo ocultar la complejidad del proceso de ingeniería inversa para aquellos interesados que no son expertos?

En consecuencia, se propone un marco de referencia para la recuperación del diseño y el análisis de la documentación recuperada, que involucra a los participantes, sus intereses, las circunstancias relacionadas con la disponibilidad de los recursos, y las situaciones que se presentan al momento de realizar un proceso de ingeniería inversa. Las principales contribuciones de este trabajo son: un modelo conceptual para recuperar el diseño y analizar la documentación recuperada, basado en los estándares ISO/IEC/IEEE 42010, ISO/IEC 19506 y UML (Unified Modeling Language). Una metodología que orienta el proceso de ingeniería inversa, según el contexto donde surja la necesidad, y que permite a los participantes obtener resultados ajustados a sus intereses. La caracterización de los contextos de uso de la ingeniería inversa. La inclusión del análisis del contexto en el proceso de ingeniería inversa, representado por el ámbito, las situaciones, los recursos, los participantes y sus intereses. El diseño y construcción de un prototipo de un mecanismo de consulta para soportar el análisis de la documentación recuperada.

Metodología

Se realizó una investigación cualitativa que comprendió cuatro fases secuenciales. En la primera fase se revisó la literatura bajo la propuesta metodológica de Peters et al. (33), para identificar los enfoques de ingeniería inversa que se utilizan para recuperar el diseño de productos software. La búsqueda se hizo en las bases de datos IEEE, ACM y Scopus, con las cadenas (design OR architecture) AND (retrieval OR reconstruction OR recovery). Se incluyeron solo publicaciones a partir de 2018 de revistas indexadas y memorias de conferencias en inglés. Se excluyeron los documentos que no hacen referencia a la ingeniería inversa de productos software y los documentos duplicados.

En la segunda fase se caracterizaron las propuestas identificadas utilizando la técnica de análisis comparativo. Se definieron las siguientes características para establecer similitudes y diferencias: tipo de propuesta, propósito, participantes y contexto para el cual fue creada. En la tercera fase se definió la estructura conceptual del marco de

doi: 10.25100/iyc.v26i1.12840

referencia usando la técnica denominada modelado lógico para la representación del conocimiento, y se especificaron cada uno de los componentes del maro de referencia aplicando la técnica de coincidencia de patrones (34) sobre las propuestas encontradas en la revisión de la literatura. En la cuarta fase se evaluó el marco de referencia. La eficacia y la pertinencia del marco de referencia se evaluaron por medio de un estudio de caso, mientras que su aporte se determinó usando la estrategia que consiste en examinar explicaciones rivales posibles (34), para lo cual se usó el análisis comparativo entre el marco de referencia definido, y en calidad de rivales las propuestas similares identificadas en la revisión de la literatura. Como variables de comparación se usaron: las actividades realizadas para la recuperación del diseño, el modelo conceptual subyacente y los elementos que constituyen el contexto (propósito, ámbito, situaciones, recursos, participantes y sus intereses).

El estudio de caso se realizó en función de sus cinco componentes (34):

- Pregunta de investigación: ¿Cómo contribuye el marco de referencia propuesto en el proceso de recuperación del diseño de productos software?
- La principal proposición afirma que: el marco de referencia propuesto guía el proceso de recuperación del diseño de productos software en función del contexto en el que se aplica.
- Dos unidades de análisis. Una bajo el contexto de la educación y otra en el contexto de desarrollo de software.
- Para hacer inferencias lógicas se define la eficacia como la capacidad que tiene el marco de referencia para lograr el objetivo propuesto en cada unidad de análisis, y la pertinencia como el grado en que el marco de referencia oculta la complejidad y tiene en cuenta el contexto: el ámbito, el propósito, las situaciones, los objetivos, los recursos disponibles y los intereses de los participantes (Ver tabla 1.).
- Criterios para interpretar los resultados: se considera que el marco de referencia es eficaz si su valoración es superior al 75% en cada unidad de análisis del estudio de caso. La pertinencia se valora usando la escala de Lickert, con base en la percepción de los participantes en el caso de estudio.

En la Tabla 1 se presentan los instrumentos de recolección de datos, los recursos utilizados y se detallan las variables evaluadas con sus respectivas fórmulas y métricas.

doi: 10.25100/iyc.v26i1.12840

Tabla 1. Recursos metodológicos

Criterio		Contexto	
		Educación	Ingeniería de software
Instrumentos de recolección de información		Encuesta a estudiantes	Encuesta a los participantes
		Entrevista al profesor	Entrevista al Arquitecto de software
		Informes del laboratorio realizado por los estudiantes	Informe técnico del resultado presentado a la Empresa
		Informe del profesor sobre la actividad académica realizada	
		Artefactos recuperados	
Recursos	Producto objeto de estudio:	JHotDraw versión 7.0.6	Sistema CPL versión 1.0
	Enterprise Architect	Herramientas para recuperar y visualizar modelos	
	QModel-XMI	Herramientas para analizar modelos	
	Imagix 4D	Herramienta para hacer análisis de código	
Variables evaluadas		Ecuación (1) Eficacia = (RA*100)/RP RA: Resultado alcanzado, RP: Resultado previsto	
		RA: Grado de cumplimiento del objetivo de la actividad por parte de los estudiantes.	RA: Grado de cumplimiento del propósito de las vistas de diseño recuperadas.
		RP: Todos los estudiantes logran plenamente los objetivos de la actividad	RP: Todas las vistas de diseño recuperadas cumplen con su propósito
		Pertinencia: Se cuantificó la percepción de los participantes con respecto a los siguientes aspectos usando la escala de Lickert: OC Oculta la complejidad, tiene en cuenta el ámbito A, el propósito P, la situación S, los objetivos O, los recursos R, los intereses de los participantes I.	

Resultados y discusión

Inicialmente se presenta el marco de referencia conformado por un sistema conceptual y una parte instrumental, que surgen de la teoría sobre la naturaleza del proceso de ingeniería inversa y cómo abordarlo. Luego se presentan los resultados de la evaluación del marco de referencia y finalmente se abordan el análisis de los resultados.

Sistema conceptual

El sistema conceptual del marco de referencia está compuesto por un modelo conceptual, una base teórica y una base conceptual (ver Figura 1). Este último se entiende como una representación de conocimiento compartido científicamente, externo, preciso, completo y consistente, que facilita la enseñanza y comprensión del objeto de estudio. La base conceptual y la base teórica corresponden a la literatura disponible sobre ingeniería inversa. El modelo conceptual propuesto proporciona una mirada integral de la ingeniería inversa y detallada de la recuperación del diseño y análisis de la documentación recuperada, que facilita la comprensión y el estudio este campo del conocimiento.

doi: 10.25100/iyc.v26i1.12840

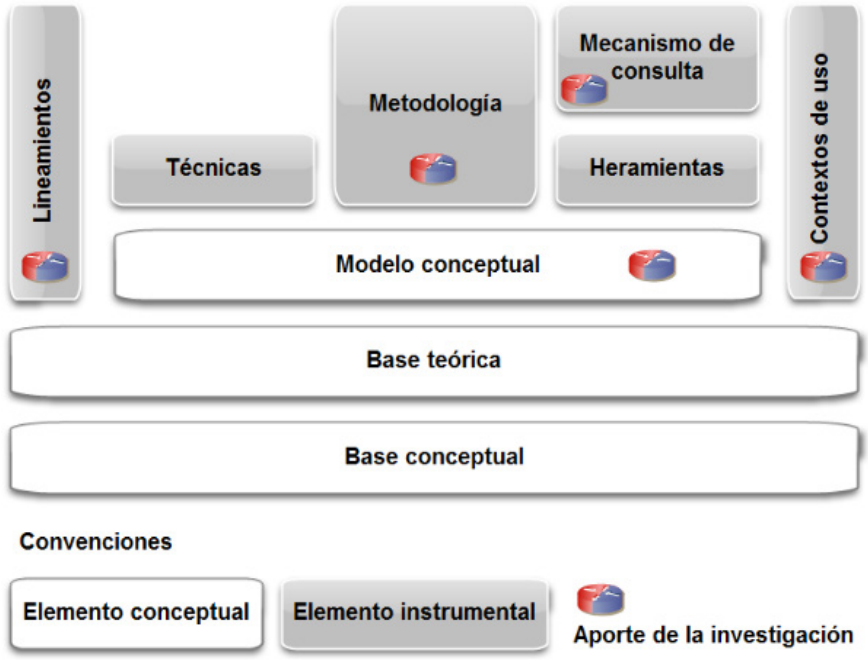


Figura 1. Marco de referencia.

El modelo conceptual comprende dos aspectos: la visión holística de la ingeniería inversa (ver Figura 2) y la recuperación del diseño. La ingeniería inversa aborda los métodos destinados a recuperar el conocimiento implícito en el software con el fin de apoyar la ejecución de actividades que requieren la comprensión de dicho sistema. Por lo tanto, se debe tener en cuenta el contexto donde surge la necesidad de recuperar el conocimiento, y el proceso que se realiza para lograr la recuperación del conocimiento. Por eso el modelo conceptual comprende estos dos elementos, representados en la Figura 2. El contexto está determinado por el ámbito, los propósitos, las situaciones que se presentan, los recursos disponibles y los participantes con sus intereses, como explica en detalle Monroy et al. (23). Por otra parte, el proceso de recuperación del conocimiento está determinado por una metodología que utiliza técnicas, instrumentos y herramientas para cumplir con los objetivos establecidos, realizando actividades que generan y requieren artefactos, de acuerdo al plan de acción.

El modelo conceptual se focaliza en la comprensión de cada actividad que aborda el proceso canónico de la ingeniería inversa (19) y los artefactos que se requieren o se generan en el desarrollo de estas actividades, como se observa en la Figura 3. El producto software y el conocimiento del experto se usan como insumos para identificar los artefactos que constituyen el sistema. Estos artefactos se descomponen aplicando técnicas de extracción de datos. Como resultado se obtiene el conocimiento implícito en el sistema, representado en KDM para facilitar la interoperabilidad entre herramientas. Los elementos explícitos en la sintaxis del producto objeto de estudio (paquetes fuente y código KDM) se obtienen aplicando reglas de mapeo sobre el lenguaje de programación. No pasa lo mismo con la capa de abstracción y la capa de recursos en tiempo de ejecución, porque los elementos están implícitos y corresponden a niveles superiores de abstracción, lo que exige un análisis incremental a partir de las representaciones KDM primitivas.

doi: 10.25100/iyc.v26i1.12840

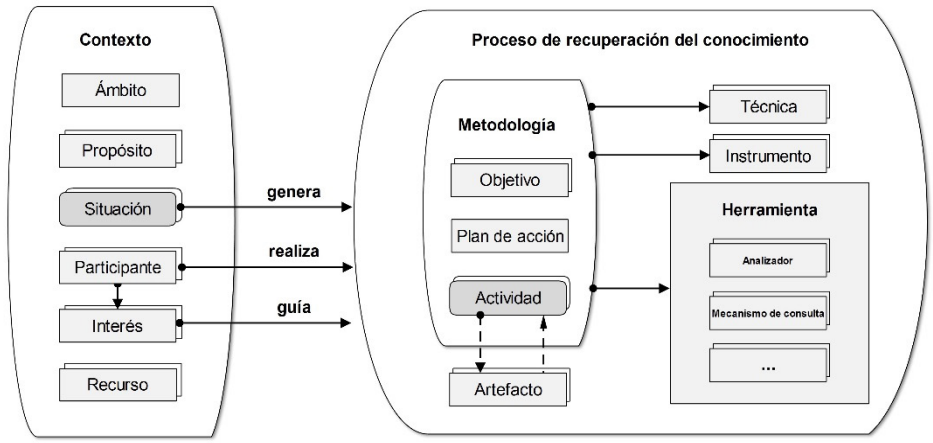


Figura 2. Visión holística del marco de referencia

Sobre los elementos identificados se aplican técnicas de organización del conocimiento para formar modelos UML del sistema, representados en XMI para garantizar la interoperabilidad. Teniendo en cuenta los objetivos establecidos para el proceso de ingeniería inversa se definen las vistas del sistema, que pueden estar compuestas por uno o por varios modelos. Para visualizar los modelos UML se utiliza cualquier herramienta que interprete código XMI.

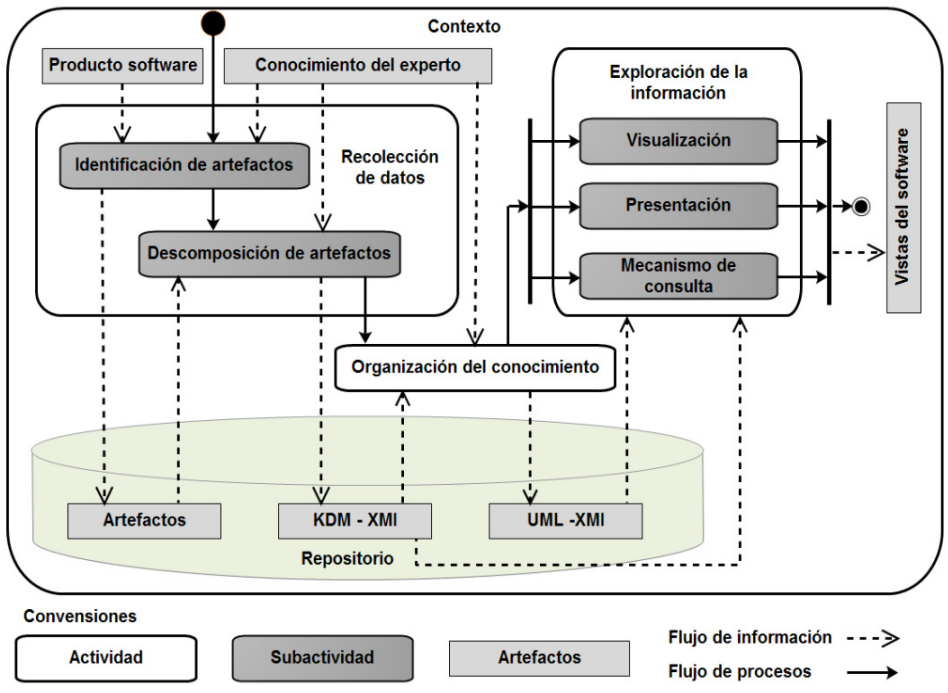


Figura 3. Modelo conceptual

Cada actividad cuenta con un objetivo, recibe insumos, emplea técnicas y produce salidas. El proceso de ingeniería inversa inicia con la extracción de datos (19). Su finalidad principal es recopilar los datos necesarios para obtener las vistas a recuperar del sistema. Esto exige como insumo el producto software que se va a analizar y el conocimiento del experto. La extracción de datos incluye dos tareas: en la primera se identifican los artefactos que conforman el sistema, aplicando técnicas de inspección manual. El resultado es el conjunto de repositorios que contienen: ejecutables, archivos de código fuente, archivos binarios, archivos de configuración, descriptores de recursos, imágenes, documentación, etc. Esto depende de la disponibilidad que se

doi: 10.25100/iyc.v26i1.12840

tenga de dichos artefactos en el contexto en el que se realiza el proceso de ingeniería inversa. En la segunda tarea se descomponen los artefactos identificados, utilizando técnicas como islas gramaticales, análisis sintáctico, análisis léxico, análisis difuso, análisis de flujo de datos, coincidencias sintácticas, perfilado e instrumentación de código. El resultado es un repositorio XMI de la representación KDM del producto.

El proceso de ingeniería inversa continúa con la actividad denominada organización del conocimiento, cuyo propósito es transformar los datos derivados de la extracción, para facilitar su almacenamiento, recuperación y análisis. El resultado son los modelos UML en XMI de la documentación recuperada. Esto se logra usando técnicas como: algoritmos de agrupación, consultas SQL, álgebra de Tarski, álgebra proposicional, álgebra relacional, modelo de reflexión y lenguajes de consulta ad-hoc. Finalmente, se realiza la exploración de información. El fin de esta actividad es ofrecer mecanismos para visualizar, presentar los resultados del proceso de ingeniería inversa y facilitar su análisis. Esto se logra con un mecanismo de consulta que extrae información no explícita en los modelos recuperados, facilitando la comprensión del producto software.

Parte instrumental

Está conformada por los lineamientos que establecen las directivas bajo las cuales se definió el marco de referencia y cómo debe ser utilizado; la metodología que guía el proceso de ingeniería inversa teniendo en cuenta los participantes, sus intereses y el contexto en el que se realiza; el mecanismo de consulta para apoyar el análisis del diseño recuperado; y la caracterización de los contextos en los que se realiza el proceso de ingeniería inversa.

Lineamientos

El marco de referencia se definió bajo los lineamientos relacionados en la Tabla 2. Para lograr un uso eficiente del marco de referencia se sugiere: conocer los elementos conceptuales que lo conforman, entender su modelo conceptual, usar la metodología propuesta aplicando las técnicas e instrumentos recomendados para cada actividad, la participación de un experto en el dominio de la aplicación y del problema, y una persona que conozca los instrumentos, herramientas y técnicas de la ingeniería inversa.

doi: 10.25100/iyc.v26i1.12840

Tabla 2. Lineamientos del Marco de referencia

Criterio	Lineamiento
Objetivo	Guiar el proceso de ingeniería inversa en función del contexto donde se realiza, para garantizar resultados pertinentes, precisos y eficientes, teniendo en cuenta que algunos participantes no tienen experiencia en ingeniería inversa.
Ámbito	Comprende la recuperación del diseño de sistemas software construidos bajo el paradigma de la POO, sin depender de las tecnologías usadas para el desarrollo y el despliegue del sistema.
Estándares de referencia	ISO/IEC/IEEE 42010:2022 para la descripción de la arquitectura. ISO/IEC 19506:2012 en calidad de meta-modelo para el descubrimiento del conocimiento. Especificación OMG: XMI 2.5.1:2015 y UML 2.5:2015 para que sea posible el intercambio de metadatos.
Referentes	Tecnológicos: el uso de paradigmas de programación, patrones arquitectónicos, tecnologías específicas, etc. Humano: debe ser acorde al conocimiento, las destrezas, las habilidades y a los intereses de los participantes en el proceso de ingeniería inversa. Recursos necesarios y disponibles para realizar el proceso de ingeniería inversa. El contexto en el que se realiza el proceso de ingeniería inversa.

Metodología para la recuperación del diseño

La propuesta se presenta como una metodología en coherencia con los cuatro axiomas de McGregor (35). El axioma epistemológico está incorporado en los elementos conceptuales del marco de referencia. El axioma axiológico establece las siguientes guías metodológicas: 1) La metodología corresponde al proceso para realizar ingeniería inversa. 2) Se integran las propuestas existentes para que los resultados sean adecuados al contexto y a la situación donde se lleva a cabo el proceso de ingeniería inversa. 3) El proceso de ingeniería inversa se estructura a partir de las actividades definidas por Tilley et al. (19). 4) El proceso define las metas que se proponen, las actividades y la secuencia lógica en que se ejecutan los recursos a nivel de entradas y los que genera (salidas) al igual que las técnicas que se aplican y los instrumentos que utiliza para alcanzar las metas planteadas. El axioma ontológico establece como objeto de estudio la ingeniería inversa bajo un contexto específico (23). Por último, el axioma lógico está constituido por la secuencia de acciones que hacen posible la realización del proceso de ingeniería inversa.

Además de los axiomas, la metodología define un proceso estructurado en fases: inicio, extracción de datos, organización del conocimiento y exploración de la información. Cada fase comprende actividades orientadas al cumplimiento de las metas y de los hitos definidos para el logro de los objetivos establecidos para el proceso, apoyándose en técnicas y el uso de diversos instrumentos y herramientas (ver Figura 4). El orden de las actividades se define en el plan de acción, que al igual que los objetivos del proceso se establece en función del contexto y la situación en la que se realiza el proceso de ingeniería inversa. Para cada actividad se define la motivación, los participantes y los artefactos (entradas y salidas). Si la actividad está estructurada en tareas, para cada tarea también se define estos aspectos.

doi: 10.25100/iyc.v26i1.12840

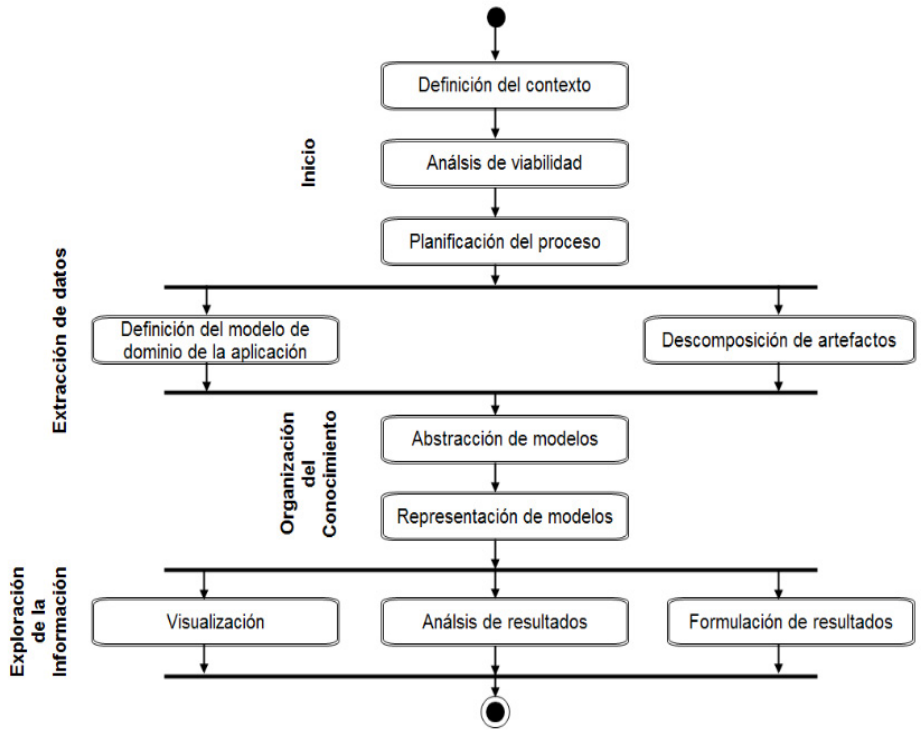


Figura 4. Metodología para la recuperación del diseño.

El propósito de la fase de inicio es entender el contexto del problema, para lo cual se identifica: el ámbito, los participantes y sus intereses, la situación que da origen al proceso de recuperación del diseño, los recursos disponibles y los objetivos que se pretenden lograr. Esta fase tiene tres actividades secuenciales con sus respectivos hitos: 1) Definición del contexto: definir el ámbito del problema, 2) Análisis de viabilidad: definir la viabilidad del proceso de ingeniería inversa, y 3) Planificación del proceso: establecer el plan de acción. Para especificar el ámbito de la situación del proceso de ingeniería inversa se establecen dos metas: 1) Definir el contexto determinado por el ámbito, los participantes y sus intereses, los objetivos del proceso y los recursos disponibles. 2) Describir la situación donde se lleva a cabo el proceso de ingeniería inversa, para lo cual se establecen las causas, se determinan las circunstancias del contexto y se definen las posibles consecuencias con el fin establecer el objetivo del proceso. El resultado de esta actividad es el documento con la descripción del contexto del problema.

Con el análisis de la viabilidad del proceso se determina si se tienen todos los recursos requeridos para lograr el objetivo establecido para el proceso de ingeniería inversa. La meta de esta actividad es: identificar la viabilidad del proyecto. Para lograrlo se recomienda: 1) Definir las vistas que se deben recuperar en función del contexto. 2) Para cada vista identificar los artefactos software requeridos 3). Comprobar la disponibilidad de éstos artefactos. Si no están disponibles el proceso termina. 4) Identificar las técnicas y herramientas requeridas para recuperar las vistas objetivo. 5) Comprobar la disponibilidad de las herramientas requeridas. Si no se no dispone de las herramientas el proceso termina. 6) Determinar si es necesario cualificar a los participantes, o contar con personal especializado y definir los costos respectivos. 7) Determinar el costo del proceso de ingeniería inversa y su viabilidad con base en la relación costo beneficio. La salida de esta actividad es la decisión sobre la viabilidad del proceso de ingeniería inversa, las vistas objetivo, los artefactos requeridos y disponibles, el costo del proceso, y las herramientas y técnicas a utilizar. Por último, en la fase de inicio se define la secuencialidad de las actividades del proceso de ingeniería

doi: 10.25100/iyc.v26i1.12840

inversa, en función de la experiencia de los participantes, asignando los responsables y los entregables. La salida es el plan de acción, los recursos y los responsables.

El propósito de la fase de extracción de datos es identificar los elementos estructurales y de comportamiento en bajo nivel del software, y las relaciones que tienen. Comprende dos actividades: 1) Definir el modelo de dominio del sistema aplicando técnicas de modelado conceptual, y 2) descomponer los artefactos del sistema para obtener las vistas objetivo, usando técnicas de transformación. Dependiendo del contexto donde se lleve a cabo el proceso de ingeniería inversa, es posible que la primera actividad se haga en forma iterativa y concurrente con actividades de la fase de inicio y de la fase de exploración de la información, porque sirve para identificar los artefactos que se tienen que recuperar, orienta a los participantes en la definición de la viabilidad del proceso, del plan de proceso y de la asignación de recursos, además, se puede usar para la abstracción de modelos. Las entradas de esta actividad son: el conocimiento del experto, la descripción del problema y los participantes. En la segunda actividad se aplican técnicas de análisis estático y dinámico sobre los artefactos que constituyen la implementación del software, para identificar los elementos que lo conforman y sus relaciones. El resultado de esta actividad es el modelo del sistema en bajo nivel representado en formatos como KDM, FAMIX, Rigi Standard Format, entre otros (5).

En la fase de organización del conocimiento el propósito es transformar los modelos en bajo nivel obtenidos en la fase anterior, en modelos de alto nivel, representándolos y almacenándolos en formatos que faciliten su recuperación y análisis. Las actividades son: 1) Abstracción de modelos: se usa el conocimiento del experto en el dominio del problema y se aplican reglas de mapeo para transformar los artefactos extraídos en elementos del sistema en alto nivel que constituyen las vistas objetivo. Para lograrlo se usan técnicas manuales, semiautomáticas y automáticas de organización del conocimiento (18). 2) Representación de modelos: tiene como meta la representación UML en XMI de los elementos y las relaciones obtenidos en la abstracción de modelos, con lo que se logra el hito de esta fase.

La fase de exploración de la información tiene como hito la preparación de un informe con el diseño recuperado y su análisis, argumentando el logro de los objetivos establecidos para el proceso de ingeniería inversa, por eso brinda mecanismos para presentar, visualizar, navegar y analizar los resultados del proceso de ingeniería inversa. Las actividades son: 1) Visualización: se presentan los resultados obtenidos en la fase anterior en forma gráfica, para que a los participantes del proceso les sea más fácil su interpretación y comprensión. Se puede hacer a nivel de arquitectura, de clases y/o de código fuente con herramientas de modelado y editores de código. Esta actividad aplica metáforas y técnicas de mapeo para presentar las vistas objetivo. Se complementa con estrategias de navegación usando hipervínculos para permitir la exploración de las vistas recuperadas. 2) Análisis de resultados: la meta es interpretar las vistas recuperadas. Se aplican técnicas de inferencia para hacer análisis de las vistas, con el apoyo del experto el dominio de la aplicación y del problema. El mecanismo de consulta propuesto puede ser usado como instrumento de apoyo en esta actividad. 3) Formulación de resultados: se organizan y presentan los resultados obtenidos en un informe final, con el fin de facilitar a los participantes en el proceso su interpretación. Dependiendo del estilo de trabajo de los participantes las actividades de esta fase se pueden realizar en forma secuencial o concurrente.

Mecanismo de consulta

Se diseñó e implementó un mecanismo para facilitar el análisis de las vistas recuperadas (36,37) en sus representaciones KDM y/o UML en XMI utilizando el

doi: 10.25100/iyc.v26i1.12840

lenguaje XQuery. El mecanismo utiliza una interfaz de texto que recibe consultas en lenguaje natural (español) y las aplica a las vistas recuperadas. Tiene dos grupos de elementos (ver Figura 5). En el primero están los artefactos que recibe como entrada o genera como salida, tales como: 1) La consulta original: es la pregunta que se formula en lenguaje natural; 2) la consulta procesada: es la sentencia expresada en XQuery; 3) el resultado: es la respuesta en lenguaje natural a la consulta original; 4) el modelo UML: es un archivo con las vistas UML recuperadas expresadas en XMI; y 5) la representación KDM: es un repositorio con los elementos del sistema, sus relaciones y entornos de operación expresados bajo la especificación KDM.

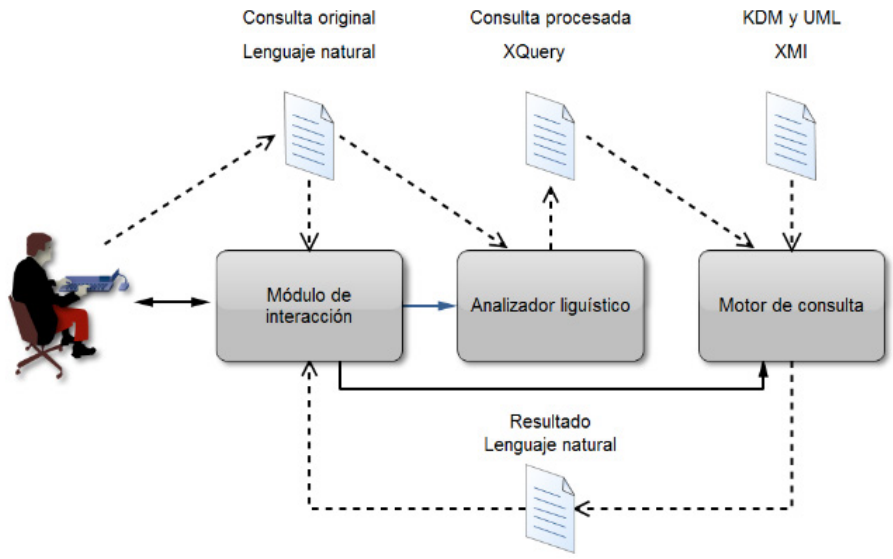


Figura 5. Mecanismo de consulta.

El segundo grupo corresponde a los componentes del mecanismo. Está conformado por: 1) El módulo de interacción: permite la entrada de la consulta original y la visualización de los resultados; 2) el analizador lingüístico: un autómata finito valida la sintaxis y la gramática de la consulta, si son correctas transforma la consulta original en consulta procesada. 3) El motor de consulta: genera el resultado ejecutando la consulta procesada sobre el repositorio que contiene los modelos. El mecanismo de consulta trabaja con el siguiente algoritmo: se escribe en lenguaje natural la consulta en el módulo de interacción, el analizador lingüístico valida la sintaxis y la gramática. Si son correctas se determina el tipo de consulta (consulta simple, consulta compuesta o cálculo de métricas,). A continuación, el motor de consulta ejecuta la consulta procesada dependiendo el tipo identificado y por último el módulo de interacción presenta los resultados.

Contextos de uso

Como resultado de la investigación se determinaron los siguientes contextos de uso de la ingeniería inversa (23): 1) Producción de software: en procesos del ciclo de vida del software como documentación, mantenimiento, reutilización de activos y verificación. 2) Seguridad informática: para definir estrategias para solucionar los riesgo e inconvenientes que ocasiona el código malicioso que pueda estar presente en el sistema (29-31), facilitando su análisis y entendimiento. Así mismo, se puede usar para identificar posibles vulnerabilidades de seguridad en sistemas software (25,29). 3) Computación forense: para construir evidencia que demuestren hechos y permitan formular hipótesis (32), facilitando la recuperación y presentación de los datos procesados electrónicamente y almacenados en medios computacionales. 4) Educación: se usa como herramienta didáctica para facilitar el aprendizaje basado

doi: 10.25100/iyc.v26i1.12840

en casos reales de éxitos y fallas, despertar la curiosidad, estimular el desarrollo de habilidades de diseño, de programación y de mantenimiento (26-28). Además, facilita la comprensión de conceptos en el campo de la ingeniería de software, como es el caso de la identificación de patrones aplicando técnicas de ingeniería inversa (22).

Evaluación del marco de referencia

El análisis de los resultados de la evaluación se realiza desde dos perspectivas. Inicialmente, se hace con base en los resultados del estudio de caso, y posteriormente, a partir de la comparación del marco de referencia con otros enfoques similares. En la Tabla 3 se presentan los resultados del estudio de caso destacando el cumplimiento de los hitos establecidos para cada fase de la metodología propuesta en el Marco de Referencia.

La eficacia del Marco de referencia se calculó con base en la ecuación (1). El resultado alcanzado corresponde al grado de cumplimiento de los objetivos establecidos para cada unidad de análisis. El en contexto educación el objetivo tiene tres componentes: el entendimiento del concepto polimorfismo, la identificación de su uso en el código fuente y la aplicación del mismo por parte del estudiante, como se observa en la tabla 4. En el contexto de producción de software la valoración del grado de cumplimiento del propósito de cada vista recupera la hizo el arquitecto, teniendo en cuenta la utilidad que le representó cada vista generada al momento de tomar decisiones para ampliar la funcionalidad del sistema, usando una escala de uno a diez, donde uno indica que no representó ninguna utilidad y 10 que fue completamente útil. En ambos casos la eficacia del marco de referencia fue mayor al 80%, por lo tanto, cumplió con este propósito. Figura 6 y 7.

doi: 10.25100/iyc.v26i1.12840

Tabla 3. Resultados del estudio de caso para cada unidad de análisis

Fase		Primera Unidad de Análisis	Segunda Unidad de análisis
	Contexto	Educación	Producción de software
	Ámbito	Enseñanza - Aprendizaje de la Programación Orientada a Objetos (POO)	Construcción de un software a la medida para una empresa del sector industrial de Cartagena
	Propósito	Desarrollar habilidades para programar aplicando el paradigma de la POO	Ampliar las funcionalidades del Sistema CPL versión 1.0
	Situación	Realización de un curso de POO en el programa de Ingeniería de Sistemas de la Universidad de Cartagena	El nuevo arquitecto de software no conoce el sistema y requiere agregar funcionalidades por solicitud de la gerencia de la Empresa. No se cuenta con documentación del sistema
	Objetivo	Comprender el polimorfismo como concepto y desarrollar habilidades para utilizarlo en forma correcta, realizando una práctica de laboratorio.	Recuperar la vista lógica y la vista de despliegue del sistema para extender su funcionalidades.
Inicio	Participantes e intereses	16 estudiantes. Quieren aprender el tema de estudio. Un Profesor. Quiere guiar el proceso de aprendizaje.	Arquitecto del producto. Debe garantizar la integridad arquitectónica del Sistema CPL 5 Programadores. Deben desarrollar las nuevas funcionalidades del sistema con base en las decisiones arquitectónicas tomadas Experto en el dominio del problema. Desea colaborar en la reconstrucción de la información del sistema Gerente de la Empresa. Desea ampliar la funcionalidad del sistema
	Recursos	El producto software objeto de estudio fue JHotDraw versión 7.0.6 Enterprise Architect para recuperar artefactos y visualizarlos Imagix 4D para calcular métricas del código fuente (10) QModel-XMI para analizar los modelos recuperados	Sistema CPL versión 1.0 Modelos de los procesos de negocio Diccionario de la Base de Datos Enterprise Architect para recuperar artefactos y visualizarlos Imagix 4D para calcular métricas del código fuente (10) QModel-XMI para analizar los modelos recuperados
	Artefactos a recuperar	Módulos (Componentes) del sistema Clases del sistema	Módulos (Componentes) del sistema Clases del sistema Vista de despliegue del sistema

doi: 10.25100/iyc.v26i1.12840

	Artefactos recuperados			
	Responsable	Artefacto	Responsable	Artefacto
Extracción de datos	Profesor	Dominio del sistema analizado: producto desarrollado en Java usado por múltiples aplicaciones para crear y editar gráficos.	Experto en el dominio del problema	Dominio del sistema denominado CPL: el control de la producción de envases plásticos. Permite un control del inventario de la producción y de la materia prima usada, generando indicadores de rendimiento que soportan decisiones de la gerencia.
		Usando Imagix 4D identificó que JHotDraw tiene 1.217 clases y 37 interfaces organizadas en 122 paquetes. En su totalidad tiene 32.054 líneas de código y 18.931 líneas de comentarios.	Arquitecto de software	Usando Imagix 4D identifió que el sistema es una aplicación Java, conformada por 1.302 clases y 3 interfaces distribuidas en 137 paquetes. En su totalidad tiene 66.095 líneas de código y 9.872 líneas de comentarios. Para la persistencia usa MARIADB junto con interfaces de SQL Server
	Estudiantes	Modelo de clases en formato XMI: Usaron Enterprise Architect para recuperar esta vista de JHotDraw.	Desarrolladores	Modelo de clases del sistema en formato XMI: usaron Enterprise Architect para recuperar el modelo de clases de cada paquete del sistema.
Organización del conocimiento	Profesor	Vista de componentes y conectores: Aplicó reglas de mapeo y usó técnicas semiautomáticas y manuales para obtener una representación en alto nivel del sistema.	Arquitecto de software	Vista de componentes y conectores: aplicó reglas de mapeo y usó técnicas semiautomáticas y manuales para obtener una representación en alto nivel del sistema.
				Vista de despliegue: Analizó la infraestructura del sistema en tiempo de ejecución.
Exploración de la información	Estudiantes	Informe del laboratorio: Usaron Enterprise Architect para visualizar las vistas recuperadas. Con el mecanismo de consulta interpretaron y analizaron las vistas recuperadas (ver Figura 6), lo que les permitió identificar la aplicación de polimorfismo.	Arquitecto de software	Con el mecanismo de consulta analizó los artefactos recuperados. Concluyó que el producto está estructurado en cuatro capas (Ver Figura 7). Evidenció el uso de buenas prácticas de desarrollo de software. Recomendó realizar un análisis de clonación de código para identificar activos reutilizables y depurar el sistema. Esta información se publica sin perjuicio del acuerdo de confidencialidad firmado con la Empresa.
	Profesor	Elaboró el informe de la actividad académica realizada.		

doi: 10.25100/iyc.v26i1.12840

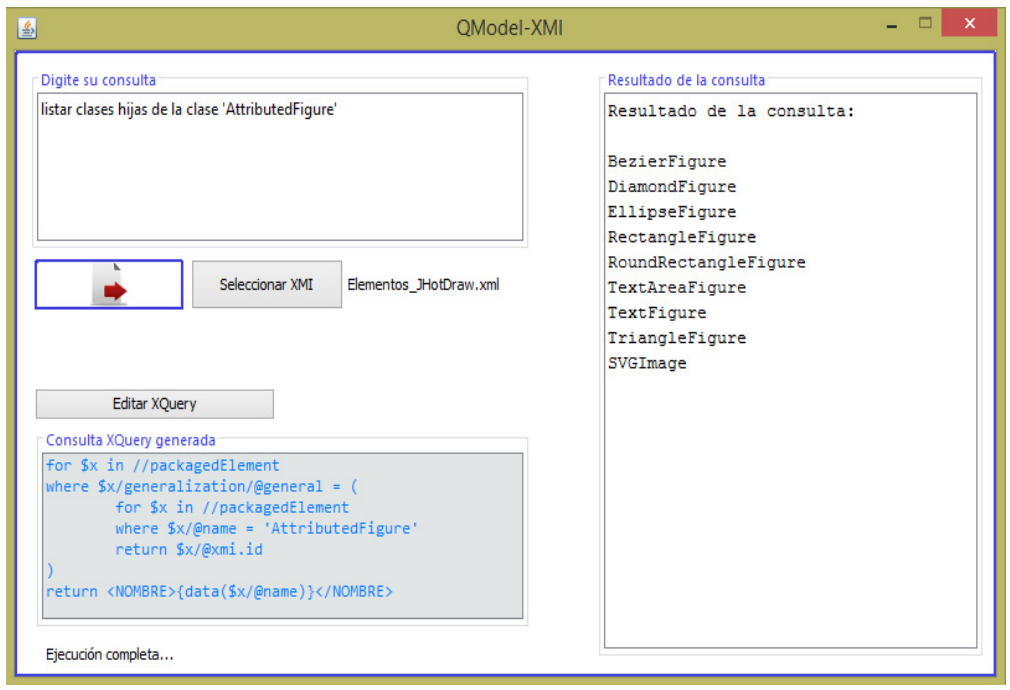


Figura 6. Uso del mecanismo de consulta.

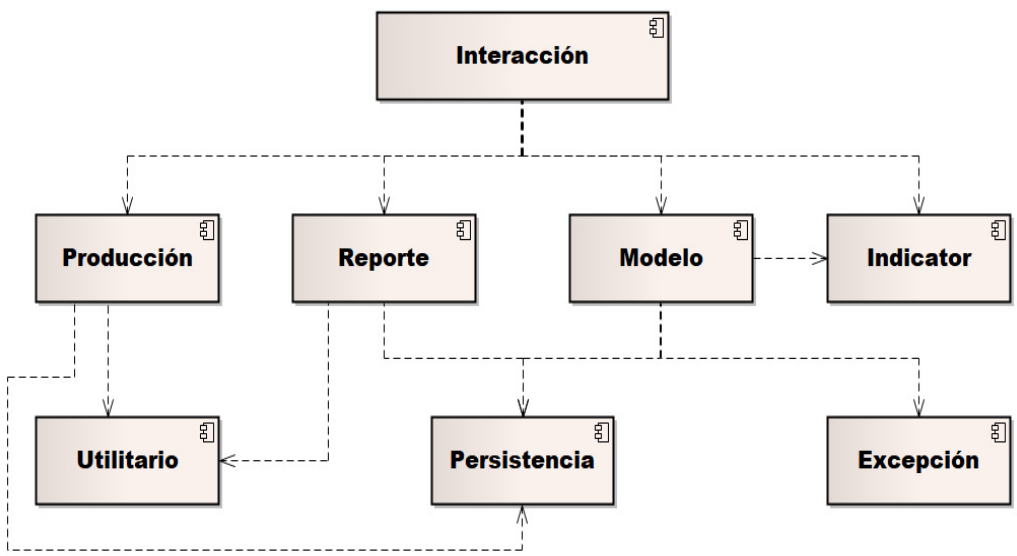


Figura 7. Vista de componentes y conectores del sistema CPL

doi: 10.25100/iyc.v26i1.12840

Tabla 4. Evaluación de la eficacia del Marco de Referencia

Contexto	Fuente	Criterio	RA	RP	Eficacia	
Educación	Informe del profesor sobre la actividad académica realizada	Entendieron el concepto de polimorfismo	15	16	94%	83%
		Identificaron dónde se aplica polimorfismo	13	16	81%	
		Aplicaron el concepto de polimorfismo	12	16	75%	
Producción de software	Entrevista al Arquitecto de software	Módulos (Componentes) del sistema	9	10	90%	87%
		Clases del sistema	9	10	90%	
		Vista de despliegue del sistema	8	10	80%	

Para evaluar la pertinencia se aplicó una encuesta a 23 participantes en el estudio de caso, 16 estudiantes, un profesor, un arquitecto de software y cinco desarrolladores para identificar su percepción usando la escala de Lickert (38), con respecto al grado en que el Marco de referencia oculta la complejidad (OC), tiene en cuenta el ámbito (A), el propósito (P), la situación (S), los objetivos (O), los recursos disponibles (R) y los intereses de los participantes (I). Como se observa en la tabla 5 cada uno de estos aspectos tiene una aprobación superior al 80%, en consecuencia, se puede afirmar que el Marco de referencia definido permite orientar el proceso de recuperación del diseño de productos software, en forma pertinente al contexto en el que se presenta la necesidad de realizarlo.

Tabla 5. Frecuencia en porcentaje para los valores de la escala de Lickert.

	Valor	OC		A		P		S		O		R		I	
		%	n	%	n	%	n	%	n	%	n	%	n	%	n
1	Totalmente de acuerdo	52,17	12	30,43	7	60,87	14	47,83	11	52,17	12	30,43	7	39,13	9
2	De acuerdo	39,13	9	65,22	15	30,43	7	47,83	11	39,13	9	52,17	12	52,17	12
3	Ni de acuerdo ni en desacuerdo	4,35	1	0,00	0	8,70	2	0,00	0	8,70	2	8,70	2	8,70	2
4	En desacuerdo	4,35	1	0,00	0	0,00	0	4,35	1	0,00	0	8,70	2	0,00	0
5	Totalmente en desacuerdo	0,00	0	4,35	1	0,00	0	0,00	0	0,00	0	0,00	0	0,00	0
Total			23		23		23		23		23		23		23

La revisión de la literatura permitió identificar 112 propuestas de las cuales 51 fueron seleccionadas y clasificadas (ver Tabla 6). Teniendo en cuenta que solo cinco definen un proceso genérico para la recuperación del diseño de productos software, en el análisis

doi: 10.25100/iyc.v26i1.12840

comparativo también se incluyeron Modisco (5), Cacophony (17) y Symphony (18) porque son referentes representativos en la ingeniería inversa. Todas las propuestas analizadas pertenecen al ámbito del desarrollo de software, y se utilizan en situaciones de mantenimiento (13-18), ingeniería inversa dirigida por modelos (5) y evolución o migración del producto (12). De igual forma, su propósito y los participantes son propios del contexto del desarrollo de software y solo (14) y (15) no presentan un modelo conceptual (MC) para facilitar su interpretación, según se observa en la Tabla 7.

Tabla 6. Clasificación de las propuestas identificadas en la revisión de la literatura.

Tipo de propuesta	Scopus	IEEE Xplore	ACM Digital Library	Total
Algoritmo	3		1	4
Técnicas de análisis del producto software	3	1		4
Evaluación de técnicas	3			3
Mantenimiento de software	1			1
Proceso de recuperación del diseño	5			5
Revisión sistemática	4	1		5
Técnica	23	2	4	29
Total	42	4	5	51

Por otra parte, todas las propuestas identificadas siguen el proceso canónico de la ingeniería inversa definido por Tilley et al (19), como se observa en la Tabla 8, por eso el marco de referencia definido las integra y las complementa al incluir aspectos determinantes del contexto, permitiendo que la recuperación y análisis del diseño se realice atendiendo los intereses de los participantes, los recursos disponibles y los propósitos que se presenten en diferentes situaciones del ámbito de la construcción del software, la educación, la seguridad informática y la computación forense. También incorpora un modelo conceptual que facilita la comprensión de la ingeniería inversa, que es útil para los participantes del proceso que no son expertos en el tema, como se observó en el estudio de caso realizado. Asimismo, el marco de referencia establece lineamientos y define una metodología que obedece al proceso canónico (19), incluyendo una fase de inicio para precisar el contexto del problema, definir la factibilidad de realizar el proceso y establecer el plan de acción. Además, define un mecanismo de consulta que se implementa en un prototipo, y que sirve para soportar el análisis de documentación representada en modelos UML. La principal ventaja del mecanismo de consulta radica en que permite hacer preguntas en lenguaje natural, facilitando a los participantes del proceso con poca experiencia hacer este tipo de análisis.

doi: 10.25100/iyc.v26i1.12840

Tabla 7. Comparativo entre propuestas existentes

Referencias	MC	Propósito	Participantes
(5)	Si	Facilitar la recuperación del diseño del sistema bajo el enfoque MDE	Ingenieros de software y desarrolladores
(12)	Si	Recuperar la arquitectura de sistemas de misión crítica	Arquitectos de software y desarrolladores
(13)	Si	Recuperar la arquitectura de líneas de productos	Equipo de desarrollo
(14)	No	Recuperar la arquitectura del producto con base en el conocimiento del contexto para el cual fue desarrollado	Ingenieros de software, desarrolladores, arquitectos de software y expertos en pruebas
(15)	No	Recuperar la arquitectura del producto y evaluar los cambios	Ingenieros de software y desarrolladores
(16)	Si	Evolución de la arquitectura del software	Ingenieros de software y desarrolladores
(17)	Si	Recuperar la arquitectura del sistema	Desarrolladores, integrador, arquitecto, gerente del producto y el arquitecto de pruebas
(18)	Si	Recuperar las vistas necesarias del producto software	Arquitectos de software, desarrolladores, responsables de la migración o personalización del producto, responsable del problema y representantes comerciales

El Marco de referencia definido no incorpora nuevas técnicas, no define nuevos algoritmos, ni modifica el proceso canónico de la ingeniería inversa, por lo tanto, no depende de tecnologías específicas como lenguajes de programación, tipo de aplicación, entre otros. Puede ser usado junto con otras propuestas si las circunstancias lo requieren. Para lograr los objetivos del proceso es indispensable definir en forma correcta el ámbito, los recursos, los participantes y sus intereses en cada situación que se presenta, así como también las vistas del sistema que se deben recuperar. Como trabajo futuro se propone definir o adaptar nuevas técnicas de recuperación del diseño, que sean intuitivas y que se integren al Marco de referencia para que puedan ser usadas por diferentes participantes, sin importar su nivel de experticia en ingeniería inversa. También se propone validar el Marco de referencia definido en contextos de seguridad informática y computación forense.

doi: 10.25100/iyc.v26i1.12840

Tabla 8. Actividades del proceso canónico de ingeniería inversa

Actividades	Referencias							
	(5)	(12)	(13)	(14)	(15)	(16)	(17)	(18)
Extracción de datos	Inyección	Micro análisis de código fuente, categorización de código	Ingeniería inversa de variantes	Hacer análisis de código	Recuperación		Análisis del dominio y de activos, análisis de requisitos.	Identificación del problema, determinación de conceptos, recopilación de datos
Organización del conocimiento		Reconstrucción de elementos arquitectónicos, Codificación teórica de relaciones y abstracción de alto nivel	Identificación de bloques y denominación de funciones, identificación de dependencias, construcción de múltiples vistas.	Identificar módulos	Detección de deterioros, mediciones	Análisis		Inferencia de conocimiento
Exploración de la información			Derivación de variantes de arquitectura	Representar la arquitectura recuperada	Visualización y predicción		Evolución	Interpretación de la información

Conclusiones

Los resultados del estudio de caso permiten concluir que el marco de referencia definido ofrece un nuevo enfoque, que orienta el proceso de recuperación y análisis del diseño de productos software, en forma pertinente con las características de los participantes y sus intereses, en función de las situaciones presentes en el contexto donde se realiza, teniendo en cuenta el ámbito, su propósito, y ocultando la complejidad de este proceso a aquellos participantes que no son expertos en ingeniería inversa. El marco de referencia integra los enfoques existentes (5, 12-18) y los complementa al incluir el contexto, un modelo conceptual, un conjunto de lineamientos, una metodología y un mecanismo de consulta. Se diferencia de estos enfoques porque puede ser usado en contextos diferentes al desarrollo de software. Además, en su aplicación se puede integrar con diferentes técnicas y herramientas.

Referencias bibliográficas

[1] B. Arasteh, R. Sadegi & K. Arasteh. Bölen: Software module clustering method using the combination of shuffled frog leaping and genetic algorithm. Data Technologies and Applications, vol. 55, no. 2,. pp. 251-279. 2021. https://doi.org/10.1108/DTA-08-2019-0138

[2] A. Shatnawi, A.D. Seriai, H. Sahraoui et al. Reverse engineering reusable software components from object-oriented APIs, J. Syst. Softw., 131, pp. 442–460, 2017

[3] F.A. Fontan, M.V. Mäntylä, M.Zanoni et al: Comparing and experimenting machine learning techniques for code smell detection, Empir. Softw. Eng., vol 21, no. 3, pp. 1143–1191 , 2016

[4] J. García, I. Ivkovic, N. Medvidovic. A comparative analysis of software architecture recovery techniques. 28th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE'13), Clayton, Australia, pp. 486–496, 2014

doi: 10.25100/iyc.v26i1.12840

[5] H. Bruneliere, J. Cabot, G. Dupé y F. Madiot, "Modisco: A model driven reverse engineering framework", Information and Software Technology, vol. 56, no. 8, pp. 1012-1032, 2014. https://doi.org/10.1016/j.infsof.2014.04.007
[6] M. Moser and J. Pichler, "eknows: Platform for Multi-Language Reverse Engineering and Documentation Generation," 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), Luxembourg, 2021, pp. 559-568, doi: 10.1109/ICSME52107.2021.00057.
[7] T. A. Ghaleb, K. Aljasser and M. A. Alturki, "Enhanced Visualization of Method Invocations by Extending Reverse-engineered Sequence Diagrams," 2020 Working Conference on Software Visualization (VISSOFT), Adelaide, SA, Australia, 2020, pp. 49-60, doi: 10.1109/VISSOFT51673.2020.00010.
[8] U. Sabir, F. Azam, S. U. Haq, M. W. Anwar, W. H. Butt and A. Amjad, "A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code," in IEEE Access, vol. 7, pp. 158931-158950, 2019, doi:10.1109/ACCESS.2019.2950884.
[9] Sparx Systems. Architect. User Guide Serieshttps://sparxsystems.com/resources/user-guides/15.2/model-domains/software-models.pdf. 2021
[10] Imagix Corp. Imagix 4D. Disponible en https://www.imagix.com/
[11] C. Stormer, "Software quality attribute analysis by architecture reconstruction (squa3re)", 11th European Conference on Software Maintenance and Reengineering (CSMR'07), IEEE, 2007, pp. 361-364. https://doi.org/10.1109/csmr.2007.43
[12] D. A. Tamburri y R. Kazman, "General methods for software architecture recovery: a potential approach and its evaluation". Empirical Software Engineering, vol. 23, no. 3, pp. 1457-1489. 2018. https://doi.org/10.1007/s10664-017-9543-z
[13] M. L. Kerdoudi, T. Ziadi, C. Tibermacine and S. Sadou, "Recovering Software Architecture Product Lines," 2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS), Guangzhou, China, 2019, pp. 226-235, doi: 10.1109/ICECCS.2019.00032.
[14] K. Ibrahim, H. Hassan, K.T. Wassif y S. Makady. Context-Aware Expert for Software Architecture Recovery (CAESAR): An automated approach for recovering software architectures. Journal of King Saud University-Computer and Information Sciences, vol. 35, no. 8, pp. 101-106, 2023.
[15] M. Schmitt Laser, N. Medvidovic, D.M. Le, & J. Garcia. ARCADE: an extensible workbench for architecture recovery, change, and decay evaluation. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering pp. 1546-1550, 2020.
[16] D. Guamán, D., J. Pérez, J. Diaz & C.E. Cuesta. Towards a reference process for software architecture reconstruction. IET Software, vol. 14, no. 6, pp. 592-606, 2020
[17] J. M. Favre, "Cacophony: Metamodel-driven software architecture reconstruction", 11th Working Conference on Reverse Engineering, IEEE, 2004, pp. 204-213. https://doi.org/10.1109/wcre.2004.15
[18] A. Van Deursen, C. Hofmeister, C, R. Koschke, L. Moonen y C. Riva, "Symphony: View-driven software architecture reconstruction". Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture, IEEE, 2004, pp. 122-132. https://doi.org/10.1109/wicsa.2004.1310696
[19] S. R. Tilley, P. Santanu y D. B. Smith, "Towards a framework for program understanding", WPC'96. 4th Workshop on Program Comprehension, IEEE, 1996, pp. 19-28. https://doi.org/10.1109/wpc.1996.501117

doi: 10.25100/iyc.v26i1.12840

[20] G. Granchelli, M. Cardarelli, P. Di Francesco, I. Malavolta, L. Iovino y A. Di Salle, "Towards recovering the software architecture of microservice-based systems". International Conference on Software Architecture Workshops (ICSAW), IEEE, 2017, pp. 46-53. https://doi.org/10.1109/icsaw.2017.48
[21] M. E. Monroy, J. L. Arciniegas y J. Rodríguez, "Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura", Información Tecnológica, vol. 27, no. 5, pp. 201-220, 2016. https://doi.org/10.4067/s0718-07642016000500022
[22] H. Zhang and J. Liu, "Research Review of Design Pattern Mining," 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2020, pp. 339-342, doi: 10.1109/ICSESS49938.2020.9237742.
[23] M. E. Monroy, J. L. Arciniegas y J. Rodríguez, "Caracterización de contextos de uso de la ingeniería inversa", Información Tecnológica, vol. 28, no. 4, pp. 75-84, 2017. https://doi.org/10.4067/s0718-07642017000400010
[24] IEEE/ISO/IEC International Standard for Software, systems and enterprise--Architecture description, International Organization for Standardization, Ginebra, Suiza, 2022
[25] A. Di Federico, P. Fezzardi and G. Agosta, "rev.ng: A Multi-Architecture Framework for Reverse Engineering and Vulnerability Discovery," 2018 International Carnahan Conference on Security Technology (ICCST), Montreal, QC, Canada, 2018, pp. 1-5, doi: 10.1109/CCST.2018.8585654.
[26] M. E. Monroy, G. E. Chanchí y M. A. Ospina, "Desarrollo de habilidades técnicas en ingeniería de software aplicando ingeniería inversa", Revista Boletín Redipe, vol. 11, no. 1, pp. 534-550, 2022. https://doi.org/10.36260/rbr.v11i1.1661
[27] E.J. López, M.A. Flores, G.L. Sandoval, B.L., Velázquez, J.J., Vázquez & L.A. Velásquez. Reverse engineering and straightforward design as tools to improve the teaching of mechanical engineering. Industry Integrated Engineering and Computing Education: Advances, Cases, Frameworks, and Toolkits for Implementation, pp. 93-118. 2019.
[28] "I. Verner & M. Greenholts. Teacher education to analyze and design systems through reverse engineering. In Educational Robotics in the Makers Era 1. Springer International Publishing, pp. 122-132, 2017.
[29] A. Sejfia, "A Pilot Study on Architecture and Vulnerabilities: Lessons Learned," 2019 IEEE/ACM 2nd International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE), Montreal, QC, Canada, 2019, pp. 42-47, doi: 10.1109/ECASE.2019.00015.
[30] A. P. David, Ghidra Software Reverse Engineering for Beginners: Analyze, identify, and avoid malicious code and potential threats in your networks and systems, Packt Publishing, 2021.
[31] M. F. Ismael and K. H. Thanoon, "Investigation Malware Analysis Depend on Reverse Engineering Using IDAPro," 2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM), Mosul, Iraq, 2022, pp. 227-231, doi: 10.1109/ICCITM56309.2022.10031698.
[32] K. Hausknecht and S. Gruičić, "Anti-computer forensics," 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2017, pp. 1233-1240, doi: 10.23919/MIPRO.2017.7973612.
[33] M. D. Peters, C. Marnie, A.C. Tricco, D. Pollock, Z. Munn, L. Alexander, L., ... & H. Khalil, (2020). Updated methodological guidance for the conduct of scoping reviews. JBI evidence synthesis, vol. 18, no. 10, pp. 2119-2126.
[34] R. K. Yin, "Case study research: Design and methods", Sage publications, 2013.
[35] S. L. McGregor y J. A. Murnane, "Paradigm, methodology and method: Intellectual integrity in consumer scholarship", International journal of consumer studies, vol. 34, no. 4, pp. 419-427, 2010. https://doi.org/10.1111/j.1470-6431.2010.00883.x

doi: 10.25100/iyc.v26i1.12840

[36] M. E. Monroy, J. L. Arciniegas y J. C. Rodríguez, "Mecanismo de Consulta para el Análisis de Arquitecturas Recuperadas". Información tecnológica, vol. 28, no. 5, pp. 87-100, 2017. http://dx.doi.org/10.4067/S0718-07642017000500011
[37] M. E. Monroy, J. C. Rodríguez y P. Puello, "QModel-XMI: un mecanismo de consulta para modelos XMI", Revista Espacios, vol. 41, no. 5, pp. 218-228, 2020. https://doi.org/10.48082/espacios-a20v41n45p17
[38] A.T. Jebb, V. Ng, & L. Tay. A review of key Likert scale development advances: 1995–2019. Frontiers in psychology, Vol. 4, no. 12, pp. 637547. 2021..

doi: 10.25100/iyc.v26i1.12840