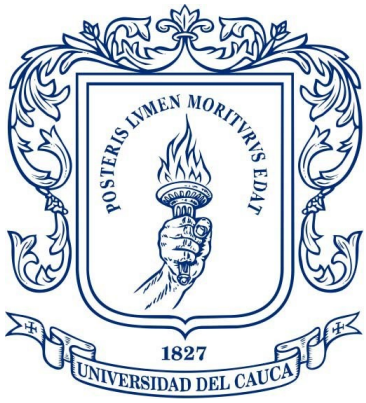
UNA LÍNEA DE PRODUCTOS DE SOFTWARE PARA ACELERAR LA PROGRAMACIÓN DE ROBOTS INDUSTRIALES CON MICROCONTROLADORES ARDUINO



Andrés Felipe Solis Pino

Tesis de la maestría en computación

Director:
Ph.D. Julio Ariel Hurtado Alegria
Codirector:
Ph.D. Pablo H. Ruiz

Universidad del Cauca
Facultad de ingeniería electrónica y telecomunicaciones
Grupo de investigación y desarrollo en ingeniería de software (IDIS)
Popayán, agosto 2022

Andrés Felipe Solis Pino

Maestría en computación

Trabajo de investigación presentado como requisito parcial para optar al título de Magíster en Computación.

Director:
Ph.D. Julio Ariel Hurtado Alegria
Codirector:
Ph.D. Pablo H. Ruiz

Popayán
2022

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Popayán, Julio 2022

DEDICATORIA

El primer agradecimiento es a Dios, la fuente inalterable de sabiduría que nos permite levantarnos cada mañana con un propósito, nos da la fuerza para continuar en los momentos más difíciles, y sobre todo nos da su paz y su amor cuando más lo necesitamos, a Él la gratitud eterna por todas las bendiciones recibidas.

A Alejandra, Guido, Camilo, Andrea y Luna por ser las personas más fieles, amorosas, compasivas y nobles que he conocido, dándome los momentos más felices de mi vida y estando siempre ahí para ayudarme en las peores situaciones. Ustedes son parte fundamental de mi vida. Lo único que tengo para decirles es que todos mis triunfos son de ustedes.

A mi familia, son parte fundamental de mi vida y me han dado apoyo y amor incondicional por esto y mucho más. Gracias.

Quiero dedicar este proyecto, especialmente a mi abuelo Reinaldo, que hoy ya no está con nosotros. Gracias por todo el amor, la sabiduría y la alegría que nos diste. Siempre estarás conmigo y hasta pronto.

A mis colegas y amigos que siempre me dieron la fuerza para seguir adelante, su amistad es invaluable para mí.

Andrés Felipe Solis Pino

AGRADECIMIENTOS

Agradecimientos sinceros al Dr. Julio Ariel Hurtado Alegria, director de este proyecto de investigación, colega y amigo que me brindó la maestría en computación. Su dedicación, apoyo y confianza hicieron posible este proyecto. Sus enseñanzas y su don de gente permanecerán para el resto de mi vida.

Un sincero agradecimiento al Dr. Pablo H. Ruiz, codirector de este proyecto, me ayudó desinteresadamente a planificar el proyecto y me enseñó sin conocerme. Muchas gracias. Sus orientaciones, enseñanzas y reflexiones fueron parte fundamental para la consecución y el éxito de este trabajo de grado.

Agradecimientos sinceros a todas las entidades que estuvieron involucradas en este proyecto, especialmente a la Corporación Universitaria Comfacauca por brindarnos sus instalaciones y equipos para realizar las pruebas de validación.

RESUMEN

En la actualidad, los sistemas robóticos industriales han tomado gran importancia en la sociedad, debido a que se utilizan en muchos dominios como la robótica de servicios, la industria de manufactura y las ciencias de la salud, entre otros. Sin embargo, hay un aumento en la complejidad del software requerido por parte de estos sistemas electromecánicos. Como respuesta, las universidades han diseñado programas de formación relacionados con esta área de conocimiento. En particular, la construcción de robots en el ámbito académico se ha centrado en la realización de prototipos, que permiten a los estudiantes comprender el dominio y sus principales bases teóricas y prácticas. Estos prototipos suelen utilizar microcontroladores (Arduino o Raspberry Pi) para dotar de inteligencia a los dispositivos electrónicos, lo que permite emular el desarrollo de software en la industria y cómo influye en el hardware subyacente. Aunque se han realizado esfuerzos para incorporar metodologías de reutilización de software en el dominio de Arduino, no se reportan muchas investigaciones que lo hagan en robots industriales que utilicen estos microcontroladores. Por lo tanto, se hace evidente la necesidad de fomentar y aplicar enfoques de reutilización que mejoren el desarrollo de software para robots industriales con Arduino, de manera que los desarrolladores (estudiantes) puedan beneficiarse de la reutilización planificada, además, de entender y familiarizarse con estos enfoques de ingeniería de software desde su formación académica, permitiendo así que la reutilización en la industria sea más factible en estos dispositivos cuando sea necesaria. Para resolver el problema planteado, se propuso como solución una línea de productos de software (IRArduino-SPL) enfocada en robots industriales con Arduino, desarrollada a través de dos iteraciones dentro de este enfoque de reutilización, la primera para observar la viabilidad de la propuesta en el dominio y la segunda para refinar la línea en base a la experiencia adquirida e incrementar el nivel de abstracción. Posteriormente, IRArduino-SPL demostró su viabilidad en el dominio mediante una prueba de concepto y su utilidad en términos de reutilización a través de un estudio de caso, logrando reutilizar aproximadamente entre el 38 y el 41% del total de las líneas de código necesarias para el funcionamiento de un robot industrial con Arduino.

ABSTRACT

Nowadays, industrial robotic systems have taken great importance in society because they are used in many domains such as service robotics, manufacturing industry, and health sciences, generating an increase in the software complexity these electromechanical systems require. In response, universities have designed training programs related to this area of knowledge. In particular, the construction of robots in academia has focused on realizing prototypes, which allow students to understand the domain and its main theoretical and practical bases. These prototypes typically use microcontrollers (Arduino or Raspberry Pi) to provide intelligence to the electronic devices, allowing emulating software development in industry and how it influences the underlying hardware. Although efforts have been made to incorporate software reuse methodologies in the Arduino domain, it reported no research on industrial robots using these microcontrollers. Therefore, the need to encourage and apply reuse approaches that improve software development for industrial robots with Arduino becomes transparent so that developers (students) can benefit from planned reuse. Besides understanding and becoming familiar with these software engineering approaches from their academic background, thus allowing reuse in the industry to be more workable in these devices when needed. To solve the problem posed, a software product line (IRArduino-SPL) focused on industrial robots with Arduino was proposed as a solution, in which two iterations of this reuse approach were developed, the first one to observe the feasibility of the proposal in the domain and the second one to refine the line based on the experience gained. Subsequently, IRArduino-SPL showed its feasibility in the domain through a proof of concept and its usefulness in reusability through a case study, reusing approximately 38-41% of the total lines of code required for the operation of an industrial robot with Arduino.

Tabla de Contenidos

Lista de tablas

Lista de figuras

Capítulo 1

Introducción

En este capítulo se hace una introducción a la investigación incluyendo la motivación, el planteamiento del problema, la hipótesis de investigación, el objetivo general y los objetivos específicos, así como se muestra la metodología utilizada para alcanzar los objetivos propuestos. Por último, se describe cómo se encuentra estructurado el presente documento.

1.1. Motivación

El desarrollo de los Sistemas Robóticos (SR) industriales siempre ha tenido un alto grado de complejidad por la multidisciplinariedad requerida para su construcción [1], esto incluye aspectos como el modelado de actuadores, la manipulación de protocolos de comunicación, la calibración de sensores, y el desarrollo de software especializado para estos dispositivos, entre otras actividades. Actualmente, en el desarrollo de aplicaciones de software para robots industriales, no existe un consenso sobre qué técnicas o metodologías son las idóneas para programar software en robots, además, no existe una aceptación general por parte de los desarrolladores sobre éstas [2]. El software para los sistemas robóticos industriales cumple un papel primordial, debido a que permite la integración de los distintos elementos que intervienen en el desarrollo de los robots, además de dotar de inteligencia a los dispositivos electrónicos [3].

Los SR industriales desempeñan un rol cada vez más importante en la industria y en la sociedad, por lo que, a medida que los robots se vuelven más complejos, se hace más evidente la necesidad de utilizar estrategias de reutilización de software, para evitar el retrabajo en el desarrollo de los activos de software existentes y centrarse en los nuevos desafíos [4]. Por ello, las universidades han desarrollado estudios de formación para enseñar las principales bases en la construcción de este tipo de dispositivos. En este sentido, carreras como la ingeniería mecatrónica o automática industrial, entre otras, han sido la respuesta, debido a que se enfocan en la

construcción del hardware y el software de los robots. No obstante, en este tipo de programas hacen poco énfasis en la ingeniería de software y en los enfoques de reutilización de software aplicados en los SR industriales, que han adquirido trascendencia tras agotar algunas metodologías actuales de construcción de software, debido a las exigencias de los usuarios y del dominio [5].

1.2. Planteamiento del problema

Los sistemas robóticos industriales son dispositivos manipuladores, funcionales y programables que permiten la operación de objetos según trayectorias programadas para realizar diversas tareas [6]. La construcción de este tipo de robots requiere una gran cantidad de conocimientos en varias disciplinas (mecánica, electrónica, informática, física, entre otras), lo que dificulta el desarrollo de dichos sistemas [7]. Entre estas disciplinas se encuentra el desarrollo de software, que es un componente esencial, debido a que, permite al robot realizar tareas de forma autónoma, unificar y direccionar los elementos del sistema, y dotar de inteligencia a los dispositivos electrónicos que lo componen [8]. En la actualidad, los SR industriales son cada vez más complejos, debido a que desempeñan un rol más importante en la sociedad. Diversas áreas como la manufacturera [9], la naval [10] y las ciencias de la salud [11] son testigos de los múltiples beneficios de la utilización de robots industriales en sus áreas, generando exigencias y expectativas cada vez más difíciles de cumplir. Asimismo, los enfoques tradicionales de construcción de SR están llegando a sus límites; las metodologías y herramientas que se utilizan actualmente no satisfacen las necesidades de hardware y de software [12], por lo que es ampliamente aceptado que se deben establecer nuevos enfoques y estrategias de reutilización de software para satisfacer las necesidades actuales en el dominio [13].

La reutilización de software es un proceso en el que se implementan o actualizan soluciones informáticas mediante la utilización de activos de software previamente creados. Es una alternativa al desarrollo de aplicaciones para un dominio específico, permitiendo que este proceso sea más eficiente, productivo y con una mejora en la calidad general del software. La idea principal es utilizar los elementos y componentes de software existentes en lugar de reconstruirlos desde cero, para utilizarlos en nuevas aplicaciones con características similares [14]. En este sentido, se han realizado acercamientos para adoptar estas estrategias de reutilización de software al dominio robótico [15]–[17]. Sin embargo, en el ámbito de los SR

industriales y en el dominio general de los robots, existe cierta resistencia al uso de los enfoques de reutilización de software, a pesar de haberse demostrado los buenos resultados que pueden tener en distintos dominios, lo que ha generado inconvenientes en el correcto desarrollo de este tipo de sistemas porque se realizan esfuerzos en la misma área de conocimiento y no se apropian estrategias probadas y exitosas [18].

El bajo nivel de adopción de enfoques de reutilización de software en la robótica industrial se debe a que la mayor parte de la cuota de mercado para la creación de software robótico son empresas, que realizan inversiones a largo plazo y exigen funcionalidad y fiabilidad, por lo que es difícil que los desarrolladores adopten nuevas técnicas de Ingeniería de Software (IS) que no están plenamente probadas y listas para la producción [19]. En segundo lugar, la heterogeneidad de los componentes de hardware y software dificulta la aplicación de estas técnicas [20], puesto que obstaculizan algunos procesos como la abstracción de modelos, la detección de patrones, la identificación del nivel de granularidad, entre otros, que son pasos esenciales para realizar una correcta reutilización del software [21]. Por último, la multidisciplinariedad de las áreas de conocimiento necesarias para la construcción de robots también es un factor que contribuye a la falta de consenso sobre las técnicas y enfoques idóneos de desarrollo de software para integrar este tipo de dispositivos [5].

Entre las alternativas (enfoques de desarrollo) más prometedoras para mejorar la reutilización de software en el dominio de la robótica industrial se encuentran la Ingeniería Dirigida por Modelos (MDE, por su denominación en inglés Model-driven engineering), la Ingeniería de Software Basada en Componentes (CBSE, por su denominación en inglés Component-based software engineering) y la Arquitectura Orientada a Servicios (SOA, por su denominación en inglés Service-oriented architecture), aunque no existe una aceptación general de las mismas [2]. Esto también se debe a que se hacen propuestas generales que abarcan muchas problemáticas del dominio, generando que no se solucionen inconvenientes reiterados de software en estos dispositivos [22]. Por lo tanto, las propuestas específicas que tratan de resolver problemáticas concretas pueden ser la solución a este tipo de dificultades, ya que se aplicarían estrategias de reutilización de software mejor planificadas, específicas y eficaces [23]. En este sentido, las Líneas de

Productos de Software (SPL, por su denominación en inglés Software Product Lines) pueden ser una solución potencial para resolver los problemas mencionados [24].

Debido a la importancia actual de los SR industriales en la sociedad, las universidades han diseñado programas de formación relacionados con esta área de conocimiento. Particularmente, la construcción de robots industriales en la academia se ha centrado en la realización de prototipos, que permiten a los estudiantes comprender el dominio y sus principales bases teóricas y prácticas [25]. Estos prototipos suelen utilizar microcontroladores (Arduino o Raspberry Pi) para dotar de inteligencia a los dispositivos electrónicos [26], lo que permite emular el desarrollo de software en la industria y cómo influye en el hardware subyacente. Ahora bien, la reutilización de software en Arduino se realiza de forma desordenada y no planificada, como es el caso de las librerías y drivers desarrollados para manejar sensores y actuadores, por lo que, no se ha adoptado plenamente los enfoques de reutilización de software en el dominio de Arduino, y mucho menos en el de los robots industriales que hacen uso de este tipo de microcontroladores, por lo tanto, es un área que puede ser abordada.

A partir del problema de la adopción de estrategias de reutilización en la robótica industrial con Arduino, y teniendo como principal preocupación mostrar una forma de materializar adecuadamente la reutilización en los SR industriales, en esta investigación se plantea la siguiente pregunta ***¿Cómo la ingeniería de línea de productos puede utilizarse para apoyar la reutilización de software en el dominio de los robots industriales en el contexto universitario?***

1.3. Hipótesis de investigación

A continuación, se presenta la oportunidad de evaluar si un enfoque sistemático puede ser parte de la respuesta.

1.3.1. Hipótesis alternativa H_1

Las líneas de productos de software, como una estrategia sistemática y centrada en el dominio, permitirá viabilizar y apoyar la reutilización de software en la robótica industrial, en un contexto universitario.

1.3.2. Hipótesis nula H_0

Las líneas de productos de software, como una estrategia sistemática y centrada en el dominio, no es suficiente para viabilizar ni apoyar la reutilización de software en la robótica industrial en el contexto universitario.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar una línea de productos de software que permita acelerar el desarrollo de aplicaciones informáticas para los sistemas robóticos industriales que hagan uso de microcontroladores Arduino, en el contexto universitario.

1.4.2. Objetivos específicos

- Caracterizar las distintas estrategias de reutilización de software que se están implementado en el dominio de los sistemas robóticos industriales a través de una revisión de la literatura.
- Integrar de forma práctica una estrategia de reutilización basada en SPL para el desarrollo de aplicaciones en la robótica industrial con microcontroladores Arduino en el contexto universitario.
- Evaluar la capacidad de la línea de productos planteada como estrategia de aceleración del desarrollo con calidad de sistemas robóticos industriales con Arduino, mediante un estudio de caso comparativo en el ámbito de un curso de robótica industrial.

1.5. Metodología

Para alcanzar los objetivos propuestos en el proyecto denominado “Una línea de productos de software para acelerar la programación de robots industriales con microcontroladores Arduino”, se utilizó una adaptación de las fases y actividades del método científico propuesto por Mario Bunge [27]. En la fase de construcción del modelo teórico, se siguieron actividades propuestas en Small-SPL [28], y algunas pautas de la Ingeniería de Dominio e Ingeniería de Aplicación del framework SPL del Software Engineering Institute (SEI) [29], además de sugerencias de PuLSE [30].

Para la fase de validación de la propuesta de reutilización se utilizó el método de estudio de casos propuesto por Runeson *et al.* en [31]. A continuación, se describen de una forma más detallada las fases y actividades seguidas en este proyecto de investigación.

1.5.1. Fase 1: Planteamiento del problema

Para esta fase se utilizan algunas directrices propuestas por Mary Shaw en [32] y Bunge [27]. Shaw plantea una estrategia para determinar la problemática a tratar en el dominio de la ingeniería de software. A continuación, se listan las principales actividades de esta fase.

- Reconocimiento de los hechos: Clasificación y selección de los acontecimientos relevantes que originaron la investigación.
- Descubrimiento del problema: Hallazgo de un vacío o faltante. Estudio de los referentes teóricos del proyecto sobre la reutilización de software en el contexto de los sistemas robóticos industriales: métodos, marcos de trabajo, lineamientos, estrategias de reutilización y experiencias referenciadas.
- Formulación del problema: Planteamiento de una pregunta, reducción del problema a su núcleo significativo.

Producto: Planteamiento del problema, pregunta problema y revisión de la literatura.

1.5.2. Fase 2: Construcción del modelo teórico

En esta fase se emplea la metodología PuLSE propuesta por Joachim Bayer en [30], donde se expresan las directrices principales para desarrollar una SPL en cualquier dominio de aplicación. Las actividades de esta fase se muestran a continuación.

- Selección de los factores pertinentes: Estudio de las diferentes estrategias de reutilización de software en el contexto de la robótica industrial.
- Identificación de estrategias: Seleccionar e identificar las estrategias de reutilización que puedan ser aplicadas en SR industriales.
- Desarrollo de la estrategia de la línea de productos de software: utilizar las estrategias seleccionadas para la construcción de las estrategias en el contexto SR industrial.

Producto: Reporte con la identificación de las estrategias de reutilización y desarrollo de la línea de productos enfocado en los robots industriales.

1.5.3. Fase 3: Validación de la estrategia de reutilización propuesta

En esta fase se siguieron las directrices reportadas por Runeson en [31], que guían el desarrollo del estudio de caso de forma sistemática y ordenada.

- Búsqueda de soportes: Deducción de consecuencias particulares que pueden haber sido verificadas en proyectos similares.
- Planteamiento de predicciones o resultados esperados al realizar la estrategia de reutilización.
- Selección y preparación de un entorno para la realización del estudio de caso.
- Diseño y planeación del estudio de caso: planteamiento de los objetivos, hipótesis y fuentes y medios para realizar el estudio de caso.
- Ejecución del estudio de caso: ejecutar la estrategia de reutilización de software.

Producto: Reporte técnico con las consecuencias verificables, además, de las predicciones de resultados y reporte técnico del estudio de caso ejecutado.

1.5.4. Fase 4: Introducción de las conclusiones en la teoría

- Confrontación de los resultados obtenidos con las predicciones.
- Reajustes a las estrategias de reutilización de software: Análisis de resultados y ajustes necesarios.

1.5.5. Fase 5: Documentación

- Escritura de artículos.
- Redacción del documento final de la tesis.

Producto: Documento final de trabajo de fin de grado y artículo científico.

Finalmente, en la siguiente figura se exponen las etapas y actividades más importantes que se siguieron en el proceso de investigación.

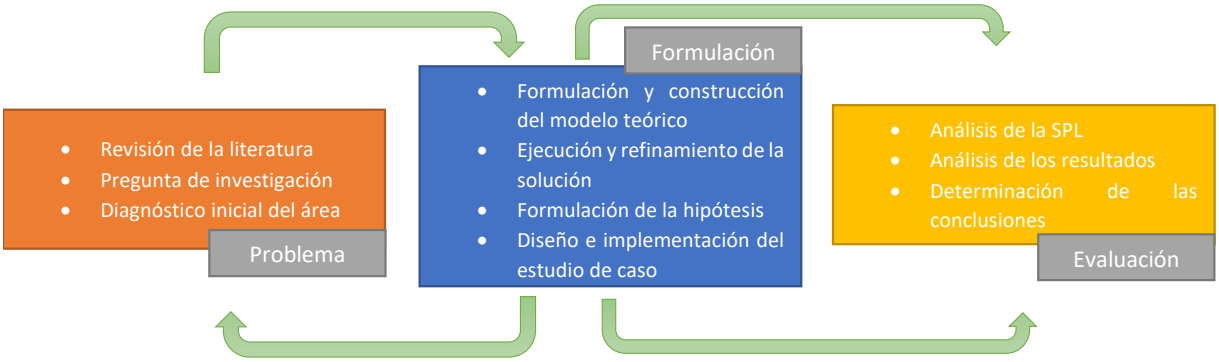


Figura 1. Metodología general del proyecto de investigación con sus fases y actividades más importantes.

1.6. Estructura del documento

El presente documento se encuentra estructurado de la siguiente forma:

Capítulo 1.
Se describen los principales elementos que sustentan la presente investigación definiendo la motivación, el planteamiento del problema, la hipótesis de investigación, los objetivos y la metodología.

Capítulo 2.
Se detallan los principales conceptos teóricos utilizados en el proyecto de investigación tales como sistemas robóticos industriales, reutilización de software, enfoques de reutilización de software, líneas de productos de software y plataforma Arduino.

Capítulo 3.
Se presenta una revisión de literatura y la caracterización de los enfoques de reutilización de software aplicados al dominio de los SR industriales con controladores y la plataforma de desarrollo Arduino.

Capítulo 4.
Se desarrolló una la construcción de la línea de productos de software para SR industriales con Arduino (IRArduino-SPL), empleando la Ingeniería de Dominio y la Ingeniería de Aplicación. Entre las actividades realizadas se concibió el modelado de dominio, la identificación de las características comunes y variables, el modelo de características de la SPL y la implementación de esta mediante una prueba de

concepto. Asimismo, se describe el refinamiento de la SPL para robots industriales con Arduino, siguiendo las sugerencias de los expertos en el dominio y definiendo el alcance mediante un enfoque colaborativo, asimismo, se describe el desarrollo de la infraestructura de la estrategia de reutilización, además de su gestión y mantenibilidad.

Capítulo 5.
Se desarrolla la segunda iteración de la SPL, retomando los diferentes modelos y refinándolos. El enfoque generativo de la primera iteración fue cambiado por un enfoque de framework para facilitar la reutilización. A partir de la estrategia implementada y refinada en el capítulo 4, se evalúa la SPL para robots industriales mediante una prueba de concepto y un estudio de caso en un contexto académico.

Capítulo 6.
Se presentan las conclusiones, las limitaciones y los trabajos futuros del proyecto de investigación.

Capítulo 2

Marco teórico y conceptual

En este capítulo se abordan los fundamentos teóricos, conceptos y argumentos relevantes que permiten comprender y sustentar el contexto y el dominio del problema en cuestión. Inicialmente, se especifican los conceptos fundamentales de la robótica industrial, sus principales componentes y las funciones que realizan este tipo de sistemas. Luego, se analiza en profundidad la reutilización de software y los principales enfoques de reutilización aplicados en el dominio. Posteriormente, se centra en las líneas de productos de software, su importancia y sus principales características. Finalmente, se abordan conceptos relacionados con la plataforma de desarrollo Arduino y la industria 4.0.

2.1. Sistemas robóticos industriales

En la actualidad, la definición más aceptada por los desarrolladores de los SR industriales es la del Robot Institute of America (RIA), que mencionan que un robot industrial es “un dispositivo manipulador multifuncional y reprogramable capaz de mover materiales, piezas, herramientas o dispositivos especiales según trayectorias variables, programadas para realizar diversas tareas” [33]. Otras definiciones establecen que para ser considerado un robot industrial debe tener al menos tres grados de libertad, dejando fuera de este concepto a dispositivos más simples y limitados. Los robots industriales son una diversificación de un concepto más amplio denominado robot, que son sistemas programables en los que convergen varias áreas de conocimiento como la mecánica, la electrónica, los sistemas y la física [34]. Son dispositivos electromecánicos que pueden realizar tareas de forma individual, repetitiva y con cierto nivel de peligrosidad; en algunos casos, pueden sustituir a los humanos para evitar cualquier riesgo. Los SR industriales cuentan con cinco elementos fundamentales que permiten el funcionamiento básico de este tipo de sistemas. Estos elementos son la estructura del dispositivo o sistema de locomoción, los elementos motrices o actuadores, el efector terminal o herramienta, los sensores

y el controlador. Aunque efectivamente estos elementos son los más utilizados en los robots, también existen características opcionales como los sistemas de visión artificial entre otros, los cuales permiten mejorar o añadir funcionalidades a estos dispositivos [35]. A continuación, se describen los principales elementos que conforman los sistemas robóticos industriales.

La **estructura del dispositivo** es el conjunto de elementos mecánicos que soportan el sistema robótico, además de proporcionar los medios para desplazarse según las trayectorias programadas. Generalmente, esta estructura se compone de elementos mecánicos como engranajes, poleas y objetos de transmisión de energía. Normalmente, el cuerpo de los SR se compone de una base fija, un cuerpo, un brazo y un antebrazo [36].

Los **elementos motrices** son los dispositivos encargados de brindar el movimiento a la estructura del robot mediante las articulaciones, generalmente, estos son motores que transmiten el movimiento a través de cables, poleas o engranajes, aunque, la tecnificación en la construcción de robots ha permitido el desarrollo de alternativas como los servomotores o los motores paso a paso [37].

El **efector terminal** es el dispositivo encargado de permitir al robot interactuar con el medio o con el objeto a manipular, sus aplicaciones son diversas y constituye un elemento fundamental en el SR industrial. Los tipos de efectores más utilizados son las pinzas para sujetar elementos, imanes magnéticos o ventosas para atrapar elementos sin daño, aunque últimamente se ha popularizado mucho el uso de láseres para realizar cortes de precisión [38].

Los **sensores** son todos los elementos que proporcionan información sobre el ambiente al sistema robótico industrial, generalmente, este tipo de detectores pueden ser mecánicos, eléctricos, magnéticos o térmicos. Además, permiten al sistema determinar en tiempo real cómo es el entorno de trabajo y cómo varía, utilizando señales lógicas al controlador [38].

El **Controlador** de un sistema robótico industrial funge como el “cerebro” del dispositivo es el encargado de procesar información que permite al robot entender cómo operar bajo un conjunto de instrucciones denominadas programa (software) robótico. Esta lógica de instrucciones envía la información a la CPU del robot donde se ejecutan todas las acciones correspondientes, sumado a esto, el controlador actúa como puente entre los demás elementos del robot, pues, recibe información de

los sensores y envía información a los actuadores para que realicen los movimientos dependiendo de las instrucciones previas, por último, una vez el dispositivo está en la posición deseada, el controlador envía una señal al actuador final para que se realice la tarea asignada [39]. El tipo de controlador más utilizado en la industria son los Controladores Lógicos Programables (PLC), mientras que, en la academia, se utilizan placas de desarrollo Arduino u ordenadores de placa reducida denominados Raspberry Pi, por su bajo precio y prestaciones [40] (Figura 2).

Figura 2. Robot industrial KUKA empleado en procesos de manufactura con controlador PLC (izquierda). Brazo robótico de carácter académico con controlador Arduino (derecha).
Fuente: [41].

Explorados los principales elementos que constituyen los sistemas robóticos industriales, a continuación, se exponen los principales conceptos sobre la reutilización de software.

2.2. Reutilización de software

La reutilización de software es el proceso por el cual se crean sistemas de software (programas informáticos) a partir de software existente, en lugar de construir programas informáticos desde cero. Es un proceso contemplado dentro de la Ingeniería de Software (IS) que busca mejorar múltiples aspectos del desarrollo de aplicaciones para un dominio particular [14]. Entre los aspectos más importantes que busca mejorar la reutilización de software están la reducción del tiempo de desarrollo, disminución de los costos, incremento de la productividad, la no reinvención de soluciones ya creadas y mejora en la fiabilidad del producto final. La reutilización no sólo se centra en usar nuevamente código previamente creado, sino

que también puede enfocarse en otros activos de software como las especificaciones de requerimientos, los diseños previamente definidos, el código funcional y depurado, los planes y casos de prueba, e incluso los paquetes de software de propósito general [42].

El concepto de reutilización de software surgió paralelo al de Ingeniería de Software en la conferencia de la OTAN (Organización del Tratado del Atlántico Norte) de 1968. Este coloquio se centró en la crisis del software y en cómo mejorar la construcción masiva y fiable del software en los sistemas informáticos. En concreto, Douglas McIlroy en este evento propuso un repositorio de componentes reutilizables y automatizables para adaptar programas informáticos con diferentes grados de personalización, precisión y robustez [43]. Está fue la primera ocasión en donde se planteó reutilizar componentes software creados para ser la base de nuevos programas y no realizar todo el trabajo desde cero. Específicamente, la reutilización de software en robots surgió formalmente en los años 80 [44], cuando se planteó realizar nuevos sistemas a partir de software existente, para ahorrar tiempo en la producción de estos complejos dispositivos. En un principio se realizaban aportes independientes que no eran reproducibles o simplemente quedaban en un estudio de caso, por lo tanto, era un enfoque de construcción poco aceptado por la comunidad de desarrolladores, hasta que en el año 2005 con la creación del Sistema Operativo Robótico (ROS, por sus siglas en inglés Robot Operating System) este framework se transforma en una de las principales propuestas para realizar la reutilización, su carácter de código abierto y su facilidad para acoger desarrollos de otros fue una de las principales razones para adoptar este marco de trabajo como el favorito de la comunidad [45]. Entre las principales características se encuentra la posibilidad de utilizar librerías compartidas y un núcleo central adaptable a los requerimientos, asimismo, en los últimos años han surgido alternativas como OROCOS, LEGO MINDSTORMS o el Robotic Framework del CERN [5].

La reutilización de software es una alternativa para desarrollar aplicaciones y sistemas informáticos de forma más eficaz, productiva y ágil [14]. Este proceso tiene varias ventajas como la reducción del tiempo de desarrollo, la disminución de costos, el incremento de la productividad y la mayor eficiencia [46]. Por otro lado, la reutilización de software tiene desventajas como la necesidad de invertir antes de obtener un resultado, ya que, el software se debe crear de una manera específica que permita el proceso de reutilización [14]. Otra problemática que aqueja la

reutilización es instruir al personal puesto que no todos están preparados para este proceso. Finalmente, la mayor desventaja de la reutilización de software es que es un proceso que debe ser institucionalizado (en una empresa por ejemplo), además de la falta de métodos o guías adecuadas para hacer el proceso de reutilización correctamente [47].

Entre los principales elementos clave que intervienen o hacen parte del proceso de reutilización se encuentran los siguientes conceptos.

2.2.1. Patrones de diseño

Los patrones de diseño son representaciones abstractas de buenas prácticas, diseños y encapsulación de la experiencia de otros desarrolladores, de forma que les permite resolver problemáticas con la lógica de las soluciones existentes. Un patrón de diseño no es una especificación detallada del problema, es simplemente una descripción del conocimiento y la experiencia acumulada que resuelve una necesidad. Tienen varias características como el nombre, la descripción, el área donde se aplica, las relaciones y la responsabilidad que cumple. Normalmente, los patrones se clasifican en tres dependiendo de su nivel de abstracción, por lo general se denominan patrones de arquitectura, patrones de diseño y patrones de dialecto. También, se clasifican por los tipos de patrones existentes, entre los más populares están los patrones de creación, los patrones estructurales y los patrones de comportamiento [48].

2.2.2. Marcos de trabajo o frameworks

La importancia de los patrones de diseño en la creación de arquitecturas de software está fuera de discusión, pero en ocasiones este concepto resulta abstracto y dificulta conectarlo directamente con las tecnologías que los implementan [49], por lo que aparece un elemento clave, los denominados frameworks o marcos de trabajo que son herramientas que permiten la reutilización de software suministrando funcionalidades genéricas enfocadas a resolver problemas recurrentes de un dominio. Normalmente, los marcos de trabajo están basados en patrones y tácticas de diseño que pueden estar compuestos por programas, bibliotecas, lenguajes de interpretación y otros tipos de herramientas. Los principales objetivos de los frameworks son acelerar el proceso de desarrollo, reutilizar el código existente y

promover buenas prácticas de desarrollo como el uso de patrones y tácticas de diseño [50].

Ahora bien, son muchas las aproximaciones, enfoques o consideraciones que la comunidad científica ha tenido para acercar la reutilización de software a la robótica industrial. En la siguiente sección se realiza una conceptualización de los principales enfoques de reutilización.

2.3. Enfoques de reutilización de software

Los enfoques de reutilización de software son la representación de cómo integrar la reutilización dentro del proceso de desarrollo de un dominio específico [51]. A continuación, se revisan los principales enfoques de reutilización de software en el dominio de los SR industriales.

2.3.1. Ingeniería dirigida por modelos

La Ingeniería Dirigida por Modelos es uno de los enfoques más populares para realizar la reutilización de software dentro de un dominio específico, consiste en encapsular la complejidad del dominio en modelos que permiten mejorar las interfaces y definir las arquitecturas. Puntualmente, un modelo se considera una pieza de software que representa de manera abstracta un sistema o fenómeno real. El Object Management Group (OMG) es el organismo encargado de crear y regular este enfoque de reutilización, estableciendo una serie de directrices para aplicarlo. La ingeniería dirigida por modelos se considera un paradigma con múltiples niveles, que son la representación de un fenómeno específico, concretamente existen cuatro niveles, siendo el nivel m0 la representación del fenómeno real o del sistema a incorporar, el modelo m1 (modelo) es el modelo de implementación específico donde se extrae el fenómeno real, el m2 (meta-modelo) es la representación de la abstracción realizada, finalmente, el m3 es el denominado meta-metamodelo y es el que representa todas las abstracciones y restricciones para modelar el conocimiento [22]. El principal objetivo que persigue la ingeniería dirigida por modelos es aumentar la productividad del desarrollo de software mediante el uso de sistemas compatibles (modelos estandarizados), utilizando patrones de diseño [52]. Entre las principales iniciativas del MDE se encuentran la arquitectura dirigida por modelos, el ecosistema de modelado y herramientas de eclipse. En cuanto al dominio de los SR industriales,

el MDE es el enfoque más utilizado por los desarrolladores independientes (no utilizan activos de ROS u otras plataformas para la reutilización) y actualmente, las herramientas más desarrolladas son los frameworks y las arquitecturas de desarrollo [5].

2.3.2. Ingeniería de software basada en componentes

La ingeniería de software basada en componentes es un paradigma de desarrollo que entiende que la separación de preocupaciones permite mejorar el desarrollo de software, buscando modificar la construcción tradicional hacia un enfoque de construcción modular, mezclando componentes estándar y protocolos de comunicación débilmente acoplados. Esto mejora el proceso de reutilización de software, ya que, si un componente se rompe, puede ser fácilmente reemplazado, mejorando la mantenibilidad de los sistemas. Un componente puede entenderse como una pieza de software con interfaces y dependencias definidas, que puede solicitar u ofrecer un conjunto de servicios o funcionalidades [53]. Específicamente, en el dominio de la robótica industrial este enfoque busca separar los principales elementos en la construcción de estos sistemas (sensores, actuadores, planificación de rutas, algoritmos) y encapsularlos en componentes que se convierten en elementos reutilizables y mantenibles, que pueden ser ensamblados para construir distintas configuraciones de sistemas robóticos o sus aplicaciones. La comunicación entre estos elementos se realiza mediante el protocolo de solicitud-respuesta y las conexiones mediante protocolos de transferencia establecidos [54]. En la actualidad, este paradigma se ha convertido en una alternativa para tener en cuenta en el dominio de los SR industriales a pesar de su antigüedad, ya que los desarrolladores entienden que el CBSE es una alternativa de reutilización probada en otros dominios que aporta múltiples ventajas sobre otros enfoques de reutilización.

2.3.3. Arquitectura Orientada a Servicios

La arquitectura orientada a servicios es un tipo de arquitectura en el contexto de las tecnologías de la información, que busca la interacción entre los componentes de un sistema de forma simple y eficiente [55]. Específicamente, en los SR intenta abstraer las principales funcionalidades del robot y permite la comunicación entre éstas, utilizando protocolos de comunicación estándar. Particularmente, este enfoque utiliza tres elementos fundamentales, el denominado corredor, publicador y suscriptor. El

proceso de interacción entre estos elementos es el siguiente, el publicador proporciona una representación del servicio a ofrecer, luego el suscriptor accede al corredor y recibe información de los servicios disponibles, finalizando, cuando el suscriptor se conecta con el publicador para solicitar los servicios requeridos. Un servicio se define como una representación lógica de la actividad del dominio a tratar, además, pueden comunicarse a través de una interfaz estándar estos elementos de software tienen un bajo acoplamiento con las tecnologías que lo contienen, permitiendo mejorar el proceso de reutilización, debido a que sus funcionalidades son independientes del hardware en el que se implementa. Por otro lado, se encontró que la utilización de este enfoque va de la mano con el uso de servicios web o computación en la nube [56].

2.4. Líneas de productos de software

Una línea de productos de software se define como “un conjunto de sistemas de software, que comparten características comunes (*core assets*), que satisfacen las necesidades específicas de un dominio o segmento particular, además, que se desarrollan a partir de un conjunto común de activos base de forma preestablecida”. Se trata de un enfoque que busca obtener resultados rentables consiguiendo productos adaptados a los mercados y clientes, utilizando componentes, modelos y servicios comunes de manera planificada [57]. La base de las SPL son los denominados activos núcleo (*core assets*), porque son las partes reutilizables que permiten desarrollar los productos, mientras que, las variables se adaptan a cada resultado que se quiere obtener. Entre los activos núcleo se incluyen arquitecturas, componentes reutilizables, modelos de dominio, requerimientos, documentación y planes de prueba [58].

El SEI menciona que existen tres actividades principales en el desarrollo de las líneas de productos de software (Figura 3), la primera es el desarrollo de los activos núcleo, la segunda es el desarrollo de los productos, mientras que la tercera es la gestión o administración de la SPL. Estas actividades están vinculadas entre sí y tienen una naturaleza altamente iterativa, lo que genera que una aplicación incorrecta de cualquiera de las actividades puede afectar directamente al proceso de producción de la línea. No existe un orden fijo de ejecución de estas actividades por lo que pueden realizarse en cualquier orden y retroalimentan a otras actividades para su realización. El desarrollo de los activos del núcleo se refiere a la planificación,

diseño y desarrollo de las partes reutilizables de la SPL, incluyendo el establecimiento de los planes de producción, que especifican la forma en que los productos son derivados a partir de los activos núcleo. El desarrollo de productos se ocupa de producir sistemas específicos a partir de un núcleo común. Los principales insumos para el desarrollo de productos son los activos núcleo, los procesos asociados a los bienes, los planes de producción y los requerimientos de variabilidad específicos de cada producto. Finalmente, la administración de la SPL se da en dos niveles, siendo el nivel técnico el que cubre la supervisión y desarrollo de los productos finales, mientras que el segundo nivel, denominado "organizativo" supervisa los esfuerzos para el correcto funcionamiento de la SPL [59].

Figura 3. Actividades esenciales en la ingeniería de líneas de productos de software.
Fuente: [29].

Generalmente, para la implementación de una línea de productos de software sobre un dominio específico, se tienen en cuenta dos actividades esenciales: la Ingeniería de Dominio y la Ingeniería de Aplicación. La Ingeniería de Dominio es la fase en la que se identifica la variabilidad y los aspectos comunes del sistema en el que se va a trabajar, y se desarrolla un conjunto de activos reutilizables básicos, como el modelo de diseño, el modelo de características y los requisitos del sistema. El objetivo principal de esta actividad esencial es definir la variabilidad que está intrínsecamente relacionada con la reutilización del sistema en cuestión. Posteriormente, la Ingeniería de Aplicación es el momento en el que se construye un producto de software basado en el modelo de características planteado en la fase anterior [60].

Las SPL son representadas mediante modelos abstractos y estos modelos se expresan mediante un modelo de características o *feature model* [61]. Básicamente, un modelo de características especifica el número de soluciones que puede proporcionar una línea de productos de software y juegan un papel central en su desarrollo, porque son la base para soportar el ciclo de creación para realizar los productos. Normalmente, una característica se considera como un rasgo o elemento distintivo que el usuario final puede percibir y sirve como medio para especificar elementos comunes y variables, siendo los aspectos comunes los que se reutilizan para otros productos [62].

2.5. Plataforma Arduino

Arduino es una plataforma para la creación y desarrollo de sistemas electrónicos de software y hardware abierto que permite la fácil integración de distintos dispositivos, permitiendo aumentar sus prestaciones. Esta plataforma consta de tres elementos que interactúan entre sí [63]. En primera instancia, es una placa de hardware libre que tiene como componente central un microcontrolador programable, con una serie de entradas y salidas que admiten conectar dispositivos que amplían sus prestaciones como sensores y actuadores (Figura 4). También, Arduino puede considerarse el entorno de desarrollo libre, gratuito y multiplataforma que permite escribir, compilar y programar en la memoria del microcontrolador, permitiendo así conectar la placa con el entorno de desarrollo generalmente por un cable USB. Por último, se puede considerar a Arduino como un lenguaje de programación que permite expresar instrucciones diseñadas para dar órdenes o instrucciones a este tipo de dispositivos [64]. Este lenguaje formal para dar instrucciones al microcontrolador está basado en C++ y tiene tres aspectos fundamentales: funciones, valores (variables y constantes) y una estructura de dos partes (inicialización y ciclo).

Por otra parte, Arduino ha desempeñado un papel fundamental en un nuevo proceso de accesibilidad a los datos y servicios a través de la internet, que se ha denominado internet de las cosas (Internet of things, abreviado IoT). Este concepto está íntimamente relacionado con la plataforma de hardware libre, ya que, la utilización de este tipo de microcontroladores ha permitido a los desarrolladores interconectar distintos dispositivos electrónicos de forma sencilla con el ciberespacio [65],

ofreciendo una serie de posibilidades para la sistematización de procesos a bajo costo.

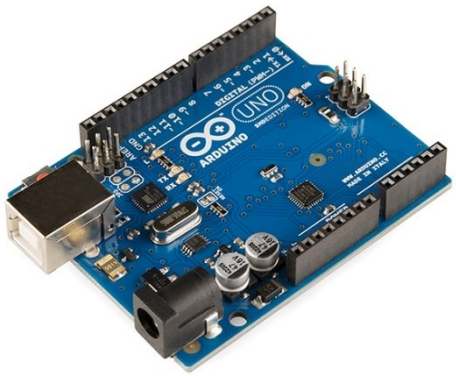


Figura 4. Ejemplo de una placa de desarrollo Arduino Uno.
Fuente: [66].

2.5.1. Industria 4.0, Internet de las cosas y Sistemas ciber-físicos

La Industria 4.0 es la cuarta revolución en los procesos de fabricación de los sistemas de producción en masa, buscando interconectar los sistemas físicos a redes centralizadas que permiten compartir datos en grandes cantidades, el uso de algoritmos para procesarlos y la interconexión de sistemas y dispositivos digitales. En la actualidad, la Industria 4.0 es un paradigma que no está totalmente adoptado y su desarrollo está en pleno proceso. Las principales tecnologías que sustentan esta revolución son el internet de las cosas, los sistemas ciber-físicos, la cultura maker y la fábrica 4.0 [67].

El término “internet de las cosas IoT” es un componente principal de la industria 4.0, a través de la cual diversos dispositivos electrónicos de diferentes tamaños y capacidades se conectan utilizando el protocolo de comunicación máquina a máquina (M2M), generalmente, utilizan una amplia gama de protocolos de red y aplicaciones que permiten el intercambio de datos, permitiendo sistematizar los entornos en los que están presentes [68]. La irrupción del internet de las cosas ha afectado a la robótica industrial, debido a que existe una necesidad constante de ampliar las posibilidades de estos sistemas [12], y es aceptado que las funcionalidades de los SR industriales dependen en gran medida de los sensores que utilizan, por lo que la interconexión de nuevos dispositivos aumentaría las funcionalidades de estos sistemas ciber-físicos, además, de aprovechar los terminales IoT que no están supeditados a conexiones físicas con el robot. Por este

motivo, ha surgido un concepto que combina el internet de las cosas y la robótica, que se denomina internet de las cosas robóticas (IoRT, por su denominación en inglés Internet of Robotic Things), que basa su funcionamiento en permitir que los dispositivos inteligentes monitoreen eventos mediante sensores de varias fuentes, y posteriormente actuar para controlar o manipular objetos del mundo real. Este concepto aprovecha la computación distribuida y en la nube para subsanar limitantes como el gasto de procesamiento en controladores y las distancias físicas entre elementos electrónicos. La principal diferencia entre IoT e IoRT es que este último toma acción en el mundo físico, según la información brindada por los sensores interconectados [69].

Capítulo 3

Caracterización de los enfoques de reutilización en el dominio de los robots industriales

En este capítulo se identifican, evalúan, describen y analizan los principales enfoques de reutilización implementados en el dominio de los sistemas robóticos industriales mediante un mapeo sistemático de la literatura. Particularmente, se presta especial atención a las principales investigaciones que combinan estas áreas de conocimiento, para proporcionar una visión general del estado actual del dominio. Finalmente, se identifican los principales desafíos, las tecnologías emergentes y los posibles vacíos en el conocimiento en relación con los SR industriales que hacen uso de enfoques de reutilización de software.

3.1. Revisión de la literatura

Para dar un panorama general de cómo se están aplicando las técnicas de reutilización de software en el dominio de los sistemas robóticos industriales, se planeó, ejecutó y reportó un mapeo sistemático de literatura, a través de una cuidadosa recopilación, confrontación y presentación de investigaciones disponibles en el área para clasificar y estructurar un campo alrededor de unas preguntas de investigación [70]. Este tipo de revisión bibliográfica se originó en los años 70 en el área de las ciencias de la salud, donde se buscaba un método eficaz para encontrar evidencias y pruebas sólidas de tratamientos para enfermedades, por lo que este tipo de trabajos fue la solución. Puntualmente, en la ingeniería, los mapeos sistemáticos han tenido un gran impacto, debido a la constante búsqueda de evidencias para poder dar continuidad a las investigaciones [71]. En el caso de la ingeniería de software, la investigadora Barbara Kitchenham fue pionera en portar este tipo de revisión a esta área de conocimiento, estableciendo bases y metodologías para permitir este tipo de estudios empíricos [5].

Un mapeo sistemático de literatura es un proceso mediante el cual se recopila información sobre un tema en específico de manera ordenada, metódica y replicable para proporcionar una visión general al lector [70].

A continuación, se exponen la planificación y presentación de los resultados del mapeo sistemático de literatura siguiendo la metodología propuesta por Petersen *et al.* [72].

3.1.1. Definición de las preguntas de investigación

Las preguntas de investigación planteadas pretenden establecer en qué contexto se encuentran las técnicas de reutilización de software en el dominio de los SR industriales, encontrar las carencias existentes en el área e investigar cómo se está realizando el proceso de abstracción e implementación de los enfoques de reutilización. En la Tabla 1 se presenta cada una de las preguntas y su motivación. A través de estas se seleccionó, analizó y categorizó la información encontrada en el área de estudio.

Tabla 1. Preguntas de investigación, motivantes y posibles respuestas utilizadas en el mapeo sistemático de literatura.

PREGUNTAS	MOTIVANTE	POSIBLES RESPUESTAS
¿Qué estudios existen que utilicen enfoques o técnicas de reutilización de software aplicadas al dominio de los SR industriales?	Determinar el número de publicaciones actuales y las tendencias de la reutilización de software en el dominio de los SR industriales.	• Component-based • Object-oriented • Domain-specific • Model-driven • Service-oriented
¿Cuál es la frecuencia de publicación de investigaciones que hagan reutilización de software para la construcción de los SR industriales?	Comprobar con qué frecuencia se están realizando estudios que impliquen la reutilización de software en el dominio de la robótica industrial, además de observar si la línea de tendencia es positiva y, por lo tanto, si es un tema actual.	Año de creación de la herramienta software
¿Qué tipo de herramientas se están desarrollando para implementar las técnicas de reutilización de software en los SR industriales?	Establecer qué tipo de herramientas (framework, DSL, aplicaciones) se están desarrollando para integrar la reutilización de software en el dominio de los SR industriales.	• API • Arquitectura • Marco de trabajo • Generador de código • Método o modelo • Middleware • Otra herramienta
¿Con qué frecuencia se relacionan las herramientas y los enfoques de reutilización de software en el dominio de los robots industriales en los estudios analizados?	Determinar cómo se están distribuyendo las herramientas creadas junto con los enfoques de reutilización de software para encontrar donde existen vacíos en el conocimiento o mayor evidencia empírica para sustentar hipótesis o desarrollar teorías.	Se realizará el cruce de información con las respuestas 1 y 3
¿En qué contexto se están haciendo los principales aportes al dominio de la robótica con técnicas de reutilización de software?	Encontrar en qué contextos se está desarrollando la reutilización de software en el dominio de la robótica industrial, para observar donde se realizan las mayores contribuciones y en donde existen vacíos en el área de conocimiento.	• Académico • Industrial • Ambos
¿Cómo se están implementado las estrategias de reutilización de software en SR industriales?	Determinar cómo se están implementando las estrategias de reutilización de software en el dominio de los robots industriales.	Se desarrollará al momento de responder la pregunta de investigación
¿Se han utilizado los enfoques de forma combinada o de manera aislada?	Cómo se están desarrollando las metodologías de forma separada o en combinación con otros métodos o metodologías.	Se desarrollará al momento de responder la pregunta de investigación
¿Las técnicas de reutilización de software en el dominio de la robótica industrial se están aplicando de manera ad-hoc o siguiendo un estándar o metodología ya probada?	Encontrar si se están realizando los esfuerzos de incluir la reutilización en el dominio robótico industrial de manera aislada o, al contrario, si los desarrolladores se están apropiando de tecnologías ya probadas como marcos de trabajo (framework), arquitecturas o API.	• Sistema operativo robótico • Otras tecnologías • Tecnología Ad-Hoc
¿Cuáles son los principales problemas y retos en el área de reutilización de software en el dominio de los sistemas robóticos industriales?	Determinar hacia qué dirección se dirige el área de reutilización de software en el dominio robótico industrial.	Se desarrollará al momento de responder la pregunta de investigación

3.1.2. Estrategia de búsqueda utilizada

Se utilizó una cadena de búsqueda para obtener sistemáticamente información de las bases de datos bibliográficas. Ésta se construyó usando los criterios PICOC, que son una excelente forma de estructurar los conceptos de manera ordenada, metodológica y abarcando todos los temas que se quieren encontrar [73]. Las bases de datos bibliográficas utilizadas fueron ACM Digital Library, EBSCO Information Services, IEEE Digital Library y ScienceDirect. Además, se añadió Scopus por ser considerada la mayor base de datos bibliográfica de artículos e investigaciones científicas. La cadena de búsqueda se aplicó al título, al resumen y a las palabras clave de todas las bases de datos mencionadas, excepto en ScienceDirect, donde la cadena de búsqueda tuvo que reducirse a ocho términos porque este sitio web acepta este número máximo de operadores lógicos. Además, se realizó una búsqueda manual en Google Académico y se seleccionaron los artículos más relevantes que relacionan la reutilización de software con los sistemas robóticos industriales (el buscador tiene en cuenta la pertinencia, el número de citaciones, los autores y la fecha de publicación para determinar la relevancia de los artículos), posteriormente, estas investigaciones se incluyeron entre los estudios a analizar. La búsqueda en las bases de datos bibliográficas comprende todos los estudios hasta abril de 2020, sin una ventana temporal restrictiva.

Tabla 2. Criterios PICOC aplicados en la cadena de búsqueda implementada.

CONCEPTO	TÉRMINOS UTILIZADOS
Población	Industrial robot*, articulated robot*, handling robot*, robotic arm*, robotic manipulators, robot* arm, robot* manipulator*
Intervención	software engineering, software development, software reuse, system development, programming
Comparación	Model-driven, Component-based, Service-oriented, framework, Domain-specific
Contexto	Academic, industry, peer-reviewed

Respecto a la cadena de búsqueda (Tabla 2), se utilizaron los criterios PICOC, que normalmente se utilizan para plantear preguntas de investigación, que son la base de los mapeos sistemáticos y que permiten definir lo que se busca con este, asimismo, es una excelente forma de estructurar las cadenas de búsqueda de forma ordenada y metodológica [74]. El término PICOC traduce Población, Intervención, Comparación, Resultado y Contexto. Estos conceptos se organizan de forma ordenada permitiendo construir la cadena de búsqueda (Tabla 3). A continuación, se

observa la cadena de búsqueda implementada en las diferentes bases de datos científicas.

Tabla 3. Cadena de búsqueda aplicada a cada una de las bases científicas.

CADENA DE BÚSQUEDA
("robot arm" **OR** "articulated robot" **OR** "articulated robots" **OR** "handling robot" **OR** "handling robots " **OR** "industrial robot" **OR** "industrial robots" **OR** "robotic arm" **OR** "robotic arms" **OR** "robotic manipulators" **OR** "robot manipulator" **OR** "robots arm" **OR** "robots manipulators") **AND** ("programming" **OR** "software development" **OR** "software engineering" **OR** "software reuse" **OR** "software reusable") **AND** ("Component-based" **OR** "CBD" **OR** "CBSE" **OR** "Domain-specific" **OR** "DSL" **OR** "framework" **OR** "Model-driven" **OR** "MDA" **OR** "MDD" **OR** "MDE" **OR** "Service-oriented" **OR** "SOA" **OR** "SOA" **OR** "code generation" **OR** "generative programming")

3.1.3. Metodología de selección de estudios primarios

El presente mapeo aplicó el protocolo guía de mapeos y revisiones sistemáticas propuesto por Petersen *et al.* [72]. Esta investigación se desarrolla en cinco etapas (Figura 5): (1) definir las preguntas de la investigación, (2) búsqueda de estudios primarios, (3) revisión de documentos basados en criterios de inclusión y exclusión, (4) clasificar los documentos y (5) extraer y mapear los datos recopilados. Cada etapa produce un resultado que se utiliza como insumo para las etapas siguientes.

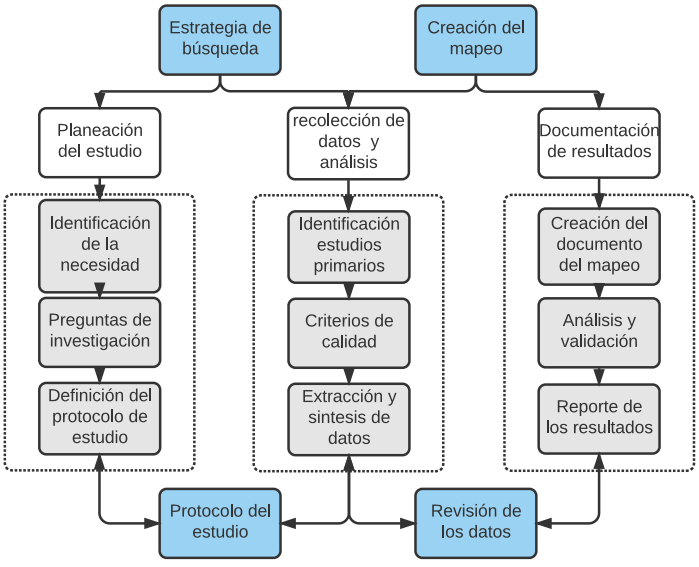


Figura 5. Metodología utilizada en el mapeo sistemático de literatura sugerido por Petersen
Fuente: Elaboración propia.

3.1.4. Criterios de selección de los estudios primarios

Siguiendo las directrices reportadas por Petersen *et al.* en [72], se utilizaron los criterios de inclusión y exclusión (Tabla 4) para determinar qué estudios son los más relevantes para la investigación y responder a las preguntas planteadas [75]. Un estudio se considera relevante si cumple todos los criterios de inclusión; por el contrario, si un estudio cumple con al menos un criterio de exclusión, no se tiene en cuenta.

Tabla 4. Criterios de inclusión y exclusión utilizados en la investigación

CRITERIOS DE INCLUSIÓN
Estudios presentados en inglés que tengan en cuenta la reutilización de software en los SR industriales.
La investigación presenta una solución concreta al área de reutilización de software en robótica industrial.
Artículos completos publicados en revistas, conferencias o congresos con revisión por pares.
CRITERIOS DE EXCLUSIÓN
El estudio primario no presenta una solución concreta al área de la robótica industrial.
Estudios no accesibles en texto completo.
Estudios que presentan material no revisado por pares.
Estudios secundarios o terciarios que reportan y analizan resultados de otras investigaciones.
Publicaciones duplicadas de los mismos autores

3.1.5. Estrategia de extracción de datos

La estrategia de extracción de datos se basó en recopilar información asociada a cada pregunta de investigación, utilizando una serie de posibles respuestas para garantizar los mismos criterios en todo el estudio (columna de respuestas en la Tabla 1). Posteriormente, se buscó el nivel de automatización de las herramientas desarrolladas para determinar en qué medida las nuevas propuestas de reutilización están ayudando a automatizar el desarrollo de aplicaciones para estos dispositivos. El nivel alto corresponde a los generadores de código, que permiten crear el software listo para personalizar a la medida del SR industrial, el nivel medio para los marcos de trabajo (framework) o lógicas de intercambio (middleware), que tienen que adaptar el código para cada plataforma. Finalmente, el nivel bajo para las técnicas de modelización o arquitecturas que se utilizan como guía para realizar el desarrollo de aplicaciones software. Adicionalmente, se cuestionó acerca de los lenguajes de

programación en los cuales están desarrolladas las herramientas finales para identificar cuáles son los más utilizados y si existe algún patrón de uso.

3.1.6. Etapa de ejecución del mapeo sistemático de literatura

En esta etapa de la investigación se ejecutaron los pasos establecidos en las etapas anteriores, por lo que se ha dividido en tres fases y se realizó con la ayuda de la herramienta en línea Parsifal [76]. En la fase 1, se aplicó la cadena de búsqueda a cada una de las cinco bases de datos bibliográficas; se obtuvieron un total de 1102 resultados. Posteriormente, se emplearon los criterios de inclusión y exclusión sobre el resumen, las palabras clave y el título de cada uno de los artículos, así como la eliminación de las investigaciones duplicadas, dando lugar a un total de 152 resultados, denominados “candidatos a estudios primarios”. En la fase 2, se realizó el mismo proceso de aplicar criterios de selección, pero en este caso sobre el artículo completo, dando como consecuencia un total de 73 artículos (Apéndice A), denominados “artículos primarios”. En la fase 3, se realizó la extracción de datos, la caracterización de los estudios y las respuestas a las preguntas de investigación propuestas. Por otra parte, en cuanto a los estudios duplicados se seleccionaron según el mayor índice de citación que tenían y aquellos de acceso completo a la publicación. En el caso de los estudios secundarios, no se tuvieron en cuenta para la extracción de información, pero sí para comparar los resultados de estos estudios con los de la presente investigación.

3.1.7. Resumen de resultados y análisis

En la Figura 6, se detalla la contribución de cada una de las bases de datos, los artículos primarios, duplicados y rechazados. Esto permite inferir que la mitad de los resultados sin analizar corresponden a la base Scopus, mientras que, IEEE Xplore Digital Library es la fuente que más artículos aceptados aportó.

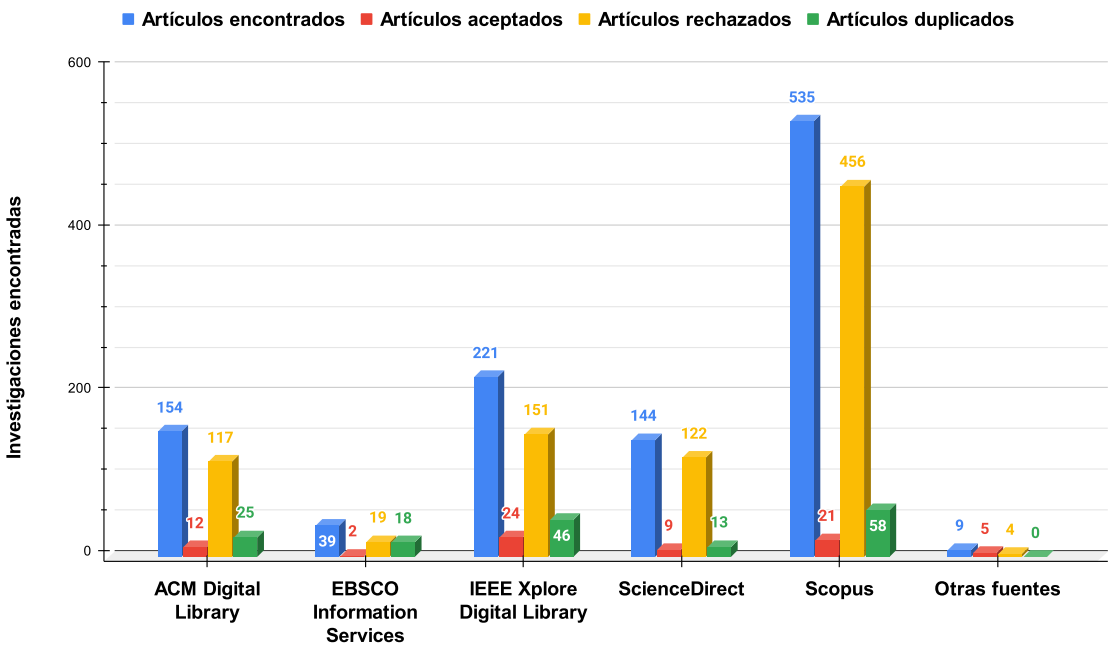


Figura 6. Aportes investigativos de cada base de datos y motores de búsqueda académicos antes de ser evaluados los artículos.
Fuente: Elaboración propia.

A continuación, se les da solución a las preguntas de investigación planteadas.

P1: ¿Qué estudios existen que utilicen enfoques o técnicas de reutilización de software aplicadas al dominio de los sistemas robóticos industriales?

En la actualidad, han surgido alternativas probadas que permiten incrementar la productividad en el desarrollo de software de los SR industriales, entre las técnicas más preponderantes se encuentran la ingeniería dirigida por modelos, la ingeniería de software basada en componentes, la arquitectura orientada a servicios y la arquitectura de software para un dominio específico (DSSA, siglas en inglés Domain-specific Software Architecture). Sin embargo, también existen enfoques menos tradicionales, como la Robótica Orientada a Objetos (OOR, siglas del inglés Object-Oriented Robotics) o la Programación Orientada a Tareas (TOP, siglas en inglés Task-Oriented Programming).

En la Figura 7 se observa que el enfoque de reutilización de software más utilizado es el MDE, con un 30.1%, que implica la creación de modelos abstractos de software y los transforma sistemáticamente en implementaciones concretas. Un modelo captura características importantes de la realidad y descuida los detalles irrelevantes [22]. Una herramienta habitual para realizar el modelado en este dominio es el

Eclipse Modeling Framework (EMF) [77], por su facilidad para definir metamodelos, transformaciones entre modelos, generación de código, construir herramientas y otras aplicaciones a partir de modelos (como el chequeo de validez de un modelo). Normalmente, las investigaciones que utilizan el MDE son realizadas en cuatro pasos aproximadamente. En primer lugar, se efectúa un análisis del dominio en el que se realiza la abstracción de este, luego, se plasma en un meta-modelo, eventualmente, se implementa el perfil de modelado en un lenguaje generalmente UML, finalmente, se crean los programas alrededor de los meta-modelos a través de algunos plugins EMF, Papyrus u otra herramienta. El objetivo del MDE es mejorar el proceso de generación de código a partir de un resumen de modelos que describen el dominio robótico.

Posteriormente, se encuentra el enfoque CBSE en un 24.7% de los estudios, su orientación cambia el énfasis de la construcción de la programación tradicional a la composición de sistemas modulares, a través de una mezcla de componentes estándar y a medida por lo tanto si una pieza de software está dañada no se tiene que crear desde el principio el programa, y sólo es necesario reemplazar dicho componente mejorando la mantenibilidad del software en los SR industriales [53]. Este enfoque pretende separar los módulos principales del dominio robótico en componentes, considerándose éstos como una pieza de software que efectúa una funcionalidad importante (por ejemplo, sensores, actuadores y trayectorias). Los componentes son elementos reutilizables y de fácil mantenimiento que pueden ser ensamblados para construir distintas configuraciones de aplicaciones [54]. La comunicación entre componentes se realiza mediante protocolos de solicitud-respuesta o publicador/subscriptor y las conexiones de comunicación sobre canales establecidos entre las partes [56]. El enfoque CBSE resulta muy atractivo para los sistemas robóticos distribuidos y permite encapsular funcionalidades de forma coherente y fácil de mantener, dando solución a problemas específicos encontrados en este tipo de dominios [78], [22].

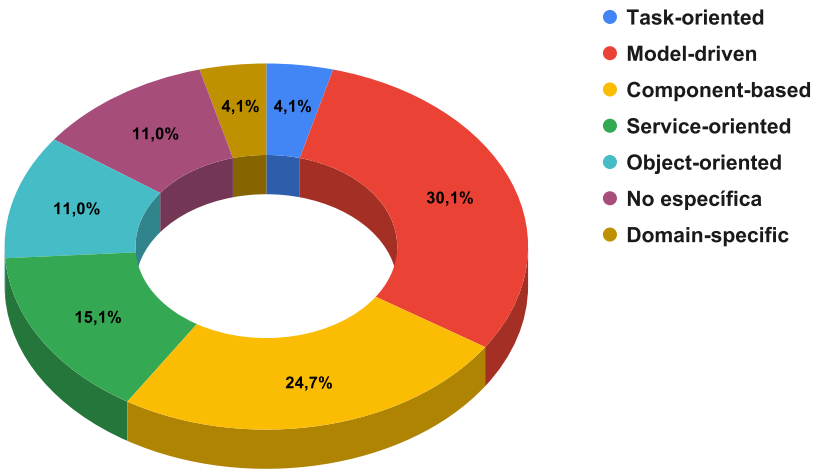


Figura 7. Distribución de metodologías de reutilización de software en el dominio de la robótica industrial.
Fuente: Elaboración propia.

El tercer enfoque más utilizado es SOA con un 15.1%, que considera el servicio como una representación lógica de una actividad del dominio que puede comunicarse a través de una interfaz estándar de intercambio de mensajes. Este enfoque se caracteriza por el bajo acoplamiento de los servicios a las tecnologías que los implementan, mejorando el proceso de reutilización, pues sus funcionalidades son independientes del hardware que los contiene. El uso del enfoque SOA en los sistemas robóticos industriales está directamente relacionado con la utilización de servicios web, que permiten ampliar sus posibilidades y eliminar las limitaciones de este tipo de dispositivos [79].

Los estudios que no especificaron el tipo de enfoque están relacionados con el desarrollo de interfaces gráficas (comunicación multimodal), que permiten mejorar la programación de los robots industriales sin utilizar ningún tipo de modelamiento o descomposición del sistema. La razón de incluirlos en el estudio está relacionada con la aceleración en el proceso de construcción de aplicaciones de software para estos dispositivos, debido a que utilizan la comunicación multimodal a través de un proceso de demostración para enseñar al robot lo que debe hacer, es decir, capturan y transforman las demostraciones en tareas que el robot ejecuta posteriormente, sin necesidad de programación adicional [80].

P2: ¿Cuál es la frecuencia de publicación de investigaciones que hagan reutilización de software para la construcción de sistemas robóticos industriales?

En la Figura 8, se aprecia la distribución anual de las publicaciones relacionadas con la reutilización de software y el dominio de los SR industriales. Es destacable indicar que los primeros 20 años, correspondientes al periodo de 1985 a 2005, tuvo un flujo de publicación relativamente bajo, con un total de 14 investigaciones correspondientes al 19.2% de los estudios totales, en cambio, el periodo comprendido de 2007 a 2020 se realizaron 59 estudios lo que equivale al 80.8% restante. Particularmente, en los últimos tres años (2020-2017) se ha publicado el 27.4% de las investigaciones, en tanto el 72.6% restante se produjo en los 31 años anteriores. También, hay que detallar que el año 2019, tuvo la mayor cantidad de publicaciones sobre este tema. Esto sugiere que es un área de investigación actual, con problemas relevantes y con acogida por la comunidad de investigadores.

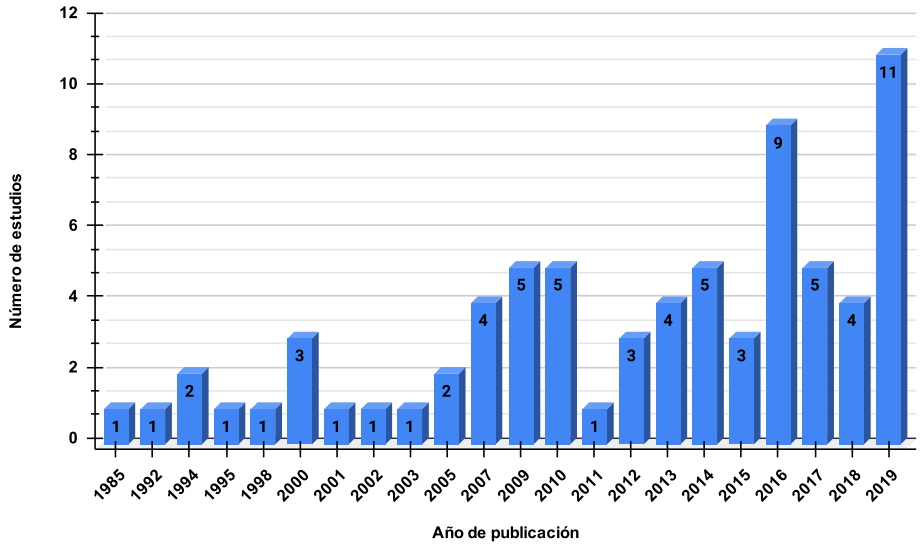


Figura 8. Distribución de investigaciones de reutilización de software en el dominio de los SR industriales.
Fuente: Elaboración propia.

El interés de la comunidad científica se acrecentó a partir del nacimiento del Sistema Operativo Robótico en el 2007, convirtiendo este marco de trabajo en punta de lanza para el desarrollo de sistemas robóticos a nivel de hardware y software [81], aumentando significativamente, a partir de ese año, el número de publicaciones en esta área del conocimiento. Específicamente, en el periodo comprendido entre 2017-

2020, ROS estuvo presente en el 60% de las investigaciones, mientras que, los desarrollos a medida (Ad-hoc) se realizaron en el 30%, dejando el 10% restante, a CERN Robotic Framework [82] y al software Lego Mindstorms, lo que permite intuir el aumento del uso de ROS como herramienta que soporta la reutilización de software.

P3: ¿Qué tipo de herramientas se están desarrollando para implementar las técnicas de reutilización de software en los SR industriales?

Las herramientas desarrolladas para apoyar los enfoques de reutilización van desde las arquitecturas de software, pasando por los marcos de trabajo hasta los generadores de código automático.

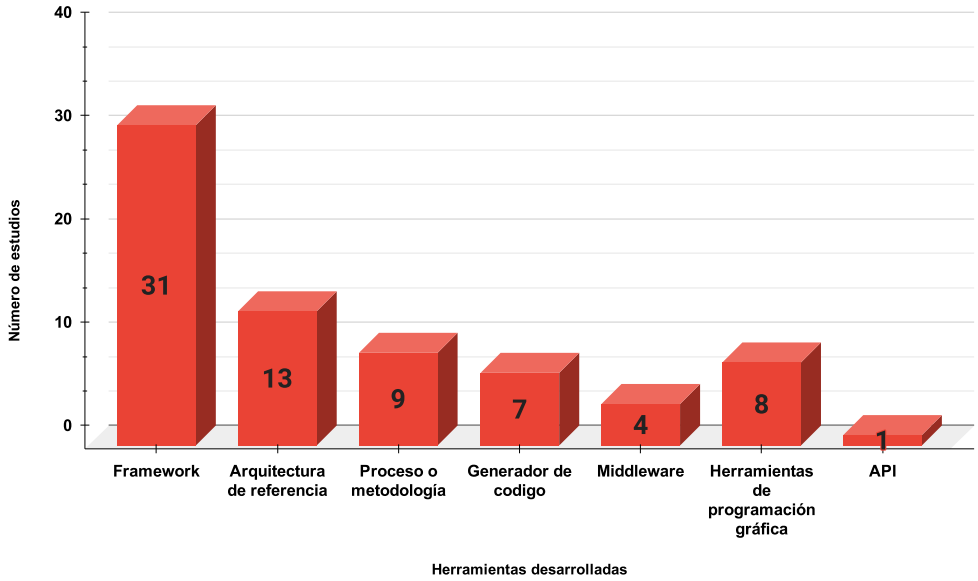


Figura 9. Tipos de herramientas desarrolladas en el dominio de los SR industriales.
Fuente: Elaboración propia.

En la Figura 9, se observa que las herramientas más desarrolladas son los marcos o entornos de trabajo, con un 42.5%, seguido por las arquitecturas de desarrollo, con un 17.8%, posteriormente los procesos o metodologías, con un 12.3% de los aportes totales a esta área. Una herramienta importante como los generadores de código se presenta en un 9.6% de los desarrollos totales. Esta distribución de herramientas sugiere que el área está realizando los pasos iniciales para establecer un conjunto estandarizado de conceptos, prácticas y criterios como base para integrar la reutilización de software en los SR industriales, pero se pueden considerar otras

estrategias de reutilización, como la generación automática de programas o las herramientas de programación gráfica.

P4: ¿Con qué frecuencia se relacionan las herramientas y los enfoques de reutilización de software en el dominio de los robots industriales en los estudios analizados?

En la Figura 10, se aprecia que los enfoques con mayor distribución de herramientas son el MDE y el CBSE, además el mayor número de contribuciones se concentra en estos dos continuamente. El enfoque SOA también tiene una gran distribución de herramientas de diferentes tipos, aunque las contribuciones se hacen con menor frecuencia que los enfoques mencionados.

La robótica orientada a objetos (OOR, por sus siglas en inglés Object-oriented Robotics), la arquitectura de software específica del dominio y la programación orientada a tareas son enfoques ortogonales a los demás, los cuales orientan conceptual o técnicamente la organización de la reutilización. Asimismo, los marcos de trabajo son las herramientas más transversales a todos los enfoques y donde la mayoría de las contribuciones se realizan de forma continua. Existe una mayor cantidad de trabajos que se concentran en la parte baja (Figura 10), que corresponde a las herramientas de bajo nivel, como las arquitecturas de software o los entornos de trabajo, surgiendo oportunidades de investigación en la parte superior, como los generadores de código. Existe una brecha en el enfoque CBSE con respecto a los generadores de código, debido a que, no existen estudios de este tipo y podría ayudar a mejorar la reutilización de software en el dominio, considerando aspectos de calidad como la escalabilidad, la portabilidad y el soporte del ciclo de vida de las aplicaciones [83].

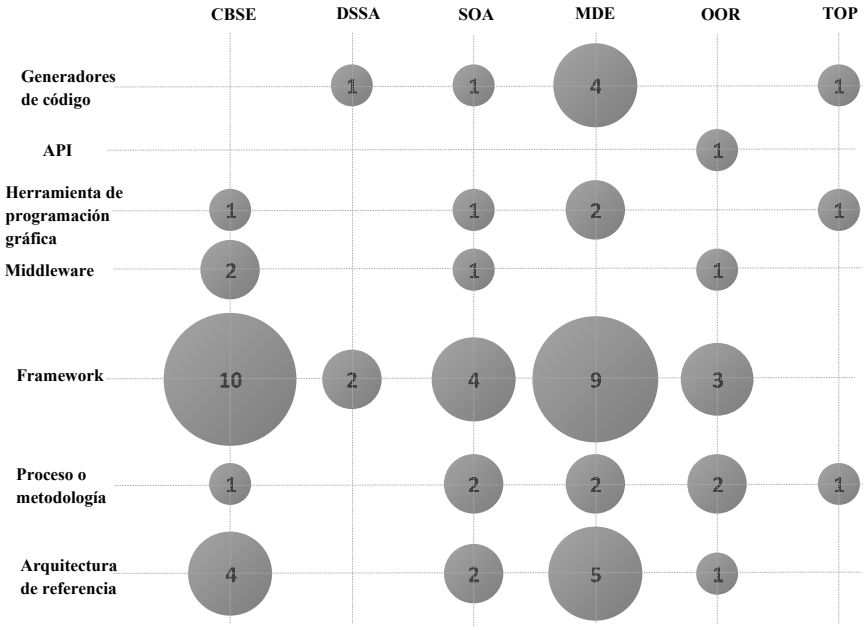


Figura 10. Distribución de herramientas según el enfoque de reutilización de software.
Fuente: Elaboración propia.

Algunas investigaciones que desarrollan herramientas de bajo nivel (marcos de trabajo, lógicas de intercambio o arquitecturas) y que se apoyan en instrumentos como ROS u OROCOS, intentan que sus desarrollos sean compatibles con el mayor número de plataformas de reutilización. Lo contrario ocurre con aquellas investigaciones que generan sus desarrollos de forma independiente porque buscan darle un mejor soporte al ciclo de desarrollo software. Este es el caso de la investigación de Estévez *et al.* en [84], que busca no sólo generar código de forma automática, sino que también apoya el adicionamiento de nuevas tareas que se requieran.

P5: ¿En qué contexto se están haciendo los principales aportes al dominio de la robótica con técnicas de reutilización de software?

Se están realizando continuas contribuciones al desarrollo de aplicaciones robóticas, principalmente desde dos lugares, la academia (investigación) y la industria (inversión) e incluso en casos concretos, estos dos estamentos trabajan de manera conjunta. La mayoría de los aportes provienen de la academia, en forma de nuevas metodologías, técnicas o aportaciones novedosas. Generalmente, estas nuevas contribuciones se ponen a prueba mediante un estudio de caso, y más adelante, si estas son validadas, las empresas adoptan estas aportaciones y las llevan al

contexto de la industria. Por otro lado, es difícil encontrar empresas que inviertan directamente en investigación, aunque existen casos concretos en los que las fábricas desarrollan una línea de productos de software de aplicaciones robóticas para brazos industriales [85]. La reutilización de software en el dominio de los robots industriales a nivel de empresa (industria) está dominado por los enfoques MDE y CBSE, siendo los enfoques más utilizados y con mayor evidencia de investigación empírica.

P6: ¿Cómo se están implementado las estrategias de reutilización de software en SR industriales?

A continuación, se describe cómo se están aplicando los principales enfoques en el dominio de los SR industriales.

Ingeniería dirigida por modelos (MDE)

Dentro de la reutilización de software en el dominio de la robótica industrial, el enfoque MDE ha surgido como una alternativa probada y funcional. Este enfoque consiste en encapsular la complejidad en modelos, mejorar las interfaces y definir arquitecturas para un dominio específico. Los modelos se consideran piezas de software que son representaciones abstractas de un sistema o fenómeno real, considerando las propiedades que son relevantes y descartando los detalles que no lo son [22]. Normalmente, el enfoque MDE se considera como una arquitectura con múltiples meta-niveles definidos por modelos (Mi), que permiten formar la pila de reflexión MDE. En el nivel más bajo se encuentra el modelo de objetos (M0), que es el código fuente ejecutándose sobre el nivel. Después, está el nivel del modelo de implementación específico de una aplicación en el dominio (M1). Posteriormente, la forma de representación para describir el dominio específico se define en el nivel conocido como meta-modelo (M2). Finalmente, en el nivel más alto de abstracción, se encuentra el modelo que representa las abstracciones y restricciones para modelar cualquier tipo de conocimiento del dominio, se le denomina meta-meta-modelo (M3) [86]. En la robótica industrial se ha adoptado este estándar de forma que el dominio se representa en tres o cuatro niveles dependiendo de la herramienta a desarrollar. Por ejemplo, en el documento de Bruyninckx en [87] el nivel más alto de la pila (M3) está representado por un meta-meta-modelo Ecore proporcionado por la herramienta de modelización EMF. El nivel (M2) está representado por un modelo abstracto (meta-modelo) independiente de la plataforma. En la capa (M1) se

representa el modelo de aplicación de los robots industriales. Por último, en este caso (M0) es una instancia del mundo real, representado por los marcos de trabajo OROCOS o ROS, dado que, la herramienta desarrollada busca estructurar y formalizar el desarrollo de aplicaciones robóticas, pero no llevarlo a la implementación como tal.

En el documento de Bubeck *et al.* en [88], buscan separar por roles de usuario el desarrollo de aplicaciones robóticas industriales, utilizando la capa (M3) para el meta-meta-modelo Ecore, luego, la capa (M2) que proporciona la cadena de herramientas para describir los meta-modelos, en este caso, un lenguaje específico del dominio para la herramienta, posteriormente, la capa (M1) brinda los mecanismos para describir el modelo. La capa (M0) es la implementación de la herramienta, permitiendo generar código para acoplarse con los nodos de ROS.

Cabe señalar que de la totalidad de las investigaciones que utilizan el enfoque MDE, solo el 18.9% sigue las capas de abstracción establecidas por la OMG, que se denominan M0 (instancias), M1 (modelo del sistema), M2 (modelo del modelo) y M3 (modelo de M2). Estos niveles permiten la adopción de la ingeniería dirigida por modelos de una manera más formal. Por otra parte, el 81.8% de los estudios restantes no siguen esta propuesta, pero sí adoptan el enfoque MDE de forma particular, en este caso, se realizan modelos donde cada uno representa de forma abstracta los conocimientos y las actividades del dominio robótico industrial.

Ingeniería de Software Basada en Componentes (CBSE)

El enfoque CBSE está siendo ampliamente utilizado para hacer la reutilización de software en los SR industriales, funciona mediante solicitudes entre el proveedor y el cliente. En general, los clientes solicitan funcionalidades que son proporcionadas por un proveedor, que suministra una colección de componentes reutilizables listos para usar, estos se almacenan en un depósito de componentes. Este enfoque utiliza los principios de modularidad y extensibilidad, además, suele ser requerido en el campo de la robótica para optimizar el desarrollo de estos sistemas y su interoperabilidad [56].

Específicamente, Wei *et al.* en [89] realiza la descomposición de un robot industrial en dos partes, el control del sistema robótico y la segunda el sistema controlado. El primero corresponde a la aplicación software del robot, mientras que el segundo a los componentes hardware, por lo que proponen una lógica de intercambio que permita

interactuar estas dos partes. Para ello, el sistema se descompone en cuatro componentes generales (alojados en un repositorio de componentes). El primero denominado “componente atómico” que interactúa con el hardware del dispositivo. Posteriormente, está el “Componente compuesto” que es la unión de varios componentes atómicos para proporcionar mayor granularidad al sistema, un ejemplo de esto es un chasis de un robot que comprende varios sensores y motores, el componente permite interactuar con estos dispositivos más allá de la variabilidad del número de elementos. Posteriormente, definen el “componente algorítmico”, que encapsula todos los algoritmos que necesita el robot, por ejemplo, la planificación de trayectorias, los filtros de Kalman o los modelos cinemáticos. Finalmente, se encuentra el “componente de aplicación” que es el lugar donde los componentes interactúan para hacer una aplicación concreta del sistema robótico. Cada uno de los cuatro componentes principales puede interactuar con los subcomponentes (por ejemplo, el componente algoritmo puede interactuar con un subcomponente que contiene las cinemáticas inversas del robot), y éstos deben obedecer.

En el documento de Beck *et al.* en [90], la reutilización permite que los recursos y las operaciones de fabricación común, se realicen entre las familias de productos. A diferencia de la investigación anterior, el proceso de reutilización por componentes se hace de la siguiente forma: el código probado y funcional del sistema robótico se sustenta en componentes básicos de la arquitectura. Posteriormente, el repositorio de componentes permite dotar a estos elementos de características específicas, para mejorar las prestaciones del sistema robótico. Finalmente, la arquitectura proporciona un entorno de reutilización para la definición de los componentes básicos, que permite el modelamiento de los recursos del sistema, los pasos y las recetas de las piezas. En resumen, el sistema está compuesto por una serie de componentes generales que proporcionan las funcionalidades básicas del robot (algoritmos, trayectorias, entre otros), pero permite adaptar otros componentes dependiendo de la aplicación (diferentes sensores o actuadores) requerida.

Arquitectura Orientada a Servicios (SOA)

La implementación de SOA en el dominio de los SR industriales sigue el mismo modelo que su homólogo en el desarrollo de software, donde existe un conjunto de servicios que se comunican entre sí utilizando protocolos de comunicación estándar [79]. En esta arquitectura se consideran tres elementos fundamentales: corredor,

publicador y suscriptor junto con sus protocolos de comunicación. El proceso de interacción entre los elementos es el siguiente, el publicador proporciona una representación del servicio al corredor, luego, el suscriptor accede al corredor y recibe información de los servicios disponibles, y por último, el suscriptor se conecta al publicador para solicitar los servicios requeridos [56].

En el documento de Rudorfer *et al.* en [91] se integra el enfoque SOA al dominio robótico, intentando mejorar el proceso de programación de estos dispositivos, mediante una interfaz intuitiva de arrastrar y soltar. La investigación propone encapsular las funcionalidades como servicios y hacer que se comuniquen a través de una interfaz común que permita la interoperabilidad, reutilización y escalabilidad. Dependiendo de la tarea que deba realizar el robot, las funcionalidades (servicios) pueden intercambiarse, permitiendo la adaptación de los sistemas a la medida de los requerimientos.

También, en el documento de Chen *et al.* en [92] combina la arquitectura orientada a servicios y la computación en la nube, acuñando el término “robot as service (RaaS)” que basa su funcionamiento en tres directrices principales, la unidad RaaS es un proveedor de servicios donde un desarrollador o cliente puede desplegar nuevos servicios o eliminarlos. La segunda directriz está relacionada con la unidad RaaS que contiene aplicaciones, el cliente puede acceder a las nuevas funcionalidades que están disponibles. Finalmente, la unidad RaaS es el publicador de servicios donde un cliente puede buscar los servicios y aplicaciones disponibles en la unidad directorio. Es decir, tiene las mismas características que el enfoque original, con la diferencia de que el editor está en la nube y es accesible desde un protocolo web remoto. Una variación de este enfoque se encuentra propuesta en el documento de Salman *et al.* en [93], el autor plantea utilizar la computación en la niebla en los SR industriales, que consiste en reemplazar las limitaciones de cómputo de los controladores de los robots, y trasladarlo a la capa borde. Aunque en este documento se plantea una arquitectura, no se implementa en un caso específico de estudio.

Por último, existe una tendencia creciente a integrar la nube con los sistemas robóticos permitiendo que accedan a recursos de almacenamiento y cómputo a distancia, además de mejorar las actividades y procesos colaborativos entre estos dispositivos [94]. Esto convierte al enfoque SOA en una alternativa a considerar para realizar la reutilización de software en el dominio.

Otros enfoques de reutilización de software utilizados

Otro enfoque importante encontrado en la revisión de la literatura fue la robótica orientada a objetos, aunque, actualmente no es el más elegido por los desarrolladores, fue uno de los primeros acercamientos de la ingeniería de software en el dominio robótico. La OOR utiliza tres conceptos principales: el componente central o núcleo, los componentes genéricos y los componentes específicos. Estos tienen propiedades de interconexión, herencias y usos. Por jerarquía, los componentes genéricos deben contener componentes específicos, mientras que los componentes específicos deben heredar características de los componentes genéricos. Además, el componente principal encapsula el control de bajo nivel, que son instancias de los controladores hardware y software que permiten la interacción entre estos, permitiendo reutilizar esta lógica para robots con características similares. Por lo tanto, los componentes genéricos usan el núcleo para proporcionar la funcionalidad general del sistema robótico, mientras que los componentes específicos heredan las funcionalidades de los componentes genéricos [95].

Un enfoque que se ha utilizado en menor medida es la OOR, que consiste en abstraer las características comunes de las tareas realizadas por el robot y descartar los elementos no comunes (sensores, actuadores y sistemas operativos), para que el desarrollador pueda centrarse en la tarea que debe realizar el dispositivo, y no preocuparse por la variabilidad del hardware. Generalmente, este enfoque se realiza a través de la comunicación multimodal y se utiliza cuando se requiere el trabajo colaborativo entre sistemas robóticos [96], [97].

Por último, también ha surgido una alternativa para mejorar el desarrollo de aplicaciones informáticas para robots que utilizan la comunicación multimodal. Esta consiste en demostrarle al robot mediante acciones (gestos o habla) las tareas que debe realizar. En este caso, el desarrollador ahorra tiempo porque no debe programar trayectorias o cinemáticas, ya que el robot imita las acciones aprendidas y esto permite que personas no expertas interactúen y programen el dispositivo, evitando tiempos de reprogramación o apagado [98]. Además, se intuye que existe una tendencia creciente a desarrollar herramientas de programación gráfica, más allá del proceso de reutilización que pueda existir.

P7: ¿Se han utilizado los enfoques de forma combinada o de manera aislada?

En la búsqueda, se encontró que pocos estudios combinan técnicas de reutilización de software en el ámbito de la robótica industrial. En el artículo de Doukas y Thramboulidis en [78] se encontró que el MDE y el CBSE se utilizan conjuntamente. La investigación propone dividir el sistema robótico en modelos y acoplarlos con la norma IEC 61499, que es un estándar para los procesos industriales y la medición, defiende que la mejor manera de construir procesos eficientes es a través de componentes. De esta forma la investigación plantea combinar estos dos paradigmas para mejorar la construcción de sistemas robóticos, permitiendo definir un modelo de aplicación (MDE) del robot usando conceptos abstraídos en componentes (CBSE). También, otra investigación que combina dos enfoques para la reutilización de software en el dominio, es la realizada por Wenger *et al.* en [99] donde describen la creación de una aplicación denominada ReApp que es un marco de trabajo basado en ROS capaz de proporcionar y ampliar las habilidades de los SR industriales más allá del lenguaje en el que está programado. La herramienta utiliza el MDE para abstraer las funciones del sistema robótico, y los desarrolladores pueden agregar nuevas habilidades en forma de componentes que se comunican naturalmente con otros y con el sistema robótico. Igualmente, utiliza la norma IEC 61499, lo que permite a ReApp seguir sus directrices para operar correcta y eficientemente especialmente en el comportamiento del sistema robótico industrial.

Por otra parte, los 71 estudios restantes no presentaron explícitamente la utilización de enfoques de reutilización de manera conjunta. Algunas investigaciones han combinado los enfoques MDE y CBSE en el dominio robótico general, que son casos exitosos, y pueden ser tomados como ejemplo para implementarse en el dominio robótico industrial [100] [101]. La mayor ventaja observada en este tipo de investigaciones es que aprovechan lo mejor de cada uno de los enfoques (combinación MDE y CBSE). Permitiendo proporcionar una forma intuitiva de crear componentes siguiendo el modelo de ejecución sincrónico, y a su vez permitir la comunicación entre esos componentes de forma natural y sin intermediarios.

P8: ¿Las técnicas de reutilización de software en el dominio de la robótica industrial se están aplicando de manera ad-hoc o siguiendo un estándar o metodología ya probada?

Se han desarrollado herramientas que permiten el proceso de reutilización en el dominio de una forma más sencilla, alternativas como ROS, OROCOS, o CORBA buscan mejorar y agilizar el proceso de desarrollo de software. En la Figura 11, se

observa la distribución de las investigaciones encontradas, separadas por las tecnologías en las que se basaron para hacer la reutilización. En este caso, el 49.3% del total de las investigaciones no están basadas en ningún estándar particular o en algún proyecto anterior (tecnologías Ad-hoc), en este tipo de trabajos se implementan las técnicas de construcción de software a medida. Por otra parte, el 30.1% utiliza ROS como base de sus investigaciones, que generalmente tienen que ver con la creación de nuevas librerías para el sistema operativo o nuevas funcionalidades. También, está OROCOS con un 4.1% de los estudios, que busca desarrollar un marco modular para el control de robots y máquinas de propósito general, de forma accesible para todos. En otras palabras, establece una infraestructura para el desarrollo de sistemas de tiempo real en el lenguaje C++ [102]. También se encontró que existen alternativas menos populares como CORBA y RTLINUX.

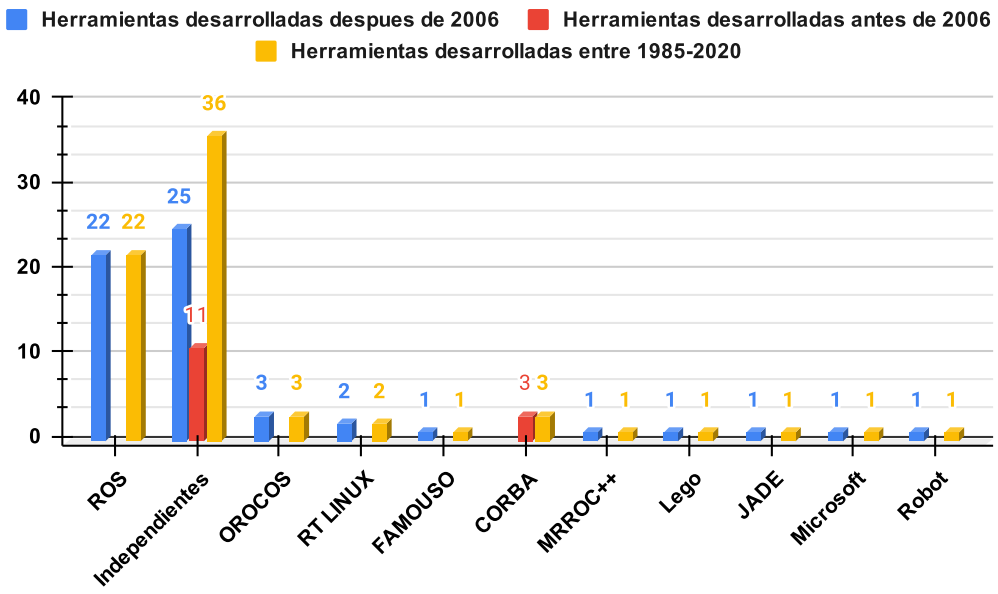


Figura 11. Distribución de tecnologías que apoyan la reutilización de software en los SR industriales.
Fuente: Elaboración propia.

Por otro lado, en el año 2007 se produjo un notable incremento en el flujo de trabajos e investigaciones que incluían enfoques de IS en sistemas robóticos, paralelamente, en ese año surgió ROS, que se ha convertido gradualmente en un referente para introducir metodologías y/o herramientas de reutilización de software en el dominio de la robótica [81]. Realizando un filtro de las investigaciones hechas desde 2007 (creación de ROS) hasta la fecha, se encuentra que los estudios que utilizan ROS

aumentan hasta el 37.3%, mientras que, las investigaciones con tecnologías Ad-Hoc decrecen a un 42.4%. Además, el uso de OROCOS crece en este periodo estableciéndose como una opción para tener en cuenta. Además, el estándar CORBA desaparece del gráfico, pues su utilización fue mayor entre 2000 y 2005, por lo que es una herramienta que actualmente no satisface las necesidades de los desarrolladores de aplicaciones robóticas.

Con los datos anteriores, se puede deducir que existe una tendencia creciente a utilizar ROS, y que podría establecer como referencia para introducir técnicas de reutilización de software en el dominio de la robótica industrial, debido a que ofrece funcionalidades valiosas como proporcionar una infraestructura de comunicación a través de la modalidad publicador y suscriptor, haciendo uso de nomenclaturas y registros que son proporcionados por un maestro. Además, cuenta con opciones específicas para cada tipo de robot, es decir, más allá de los componentes principales, ofrece funcionalidades secundarias como librerías de diseño de interfaces, herramientas de diagnóstico, mapeo, localización y navegación. Incluso permite trabajar con otros softwares como Gazebo, OpenCV, Qt y MOVEit, que actualmente son activos importantes que facilitan el desarrollo de sistemas robóticos [103]. ROS utiliza una licencia BSD, lo que ha permitido desarrollar numerosas bifurcaciones que permiten enfocarse en dispositivos específicos, este el caso de Micro-ROS orientado a llevar las funcionalidades del sistema operativo robótico a los microcontroladores o el caso de ROS-Industrial centrado a los brazos robóticos. Este aumento de la oferta de herramientas que apoyan la reutilización de software ha permitido a los desarrolladores ampliar sus investigaciones y no limitarlos a un nicho específico. Además, se espera que la adopción de este marco de trabajo por parte de los programadores aumente en el futuro, a medida que disminuya el desarrollo de nuevas tecnologías que funcionen de forma independiente, probablemente porque los desarrolladores han encontrado en ROS u otras posibilidades un soporte adecuado para llevar a cabo sus investigaciones.

P9: ¿Cuáles son los principales problemas y retos en el área de reutilización de software en el dominio de los sistemas robóticos industriales?

La implementación de enfoques de reutilización de software en los SR industriales se está llevando a cabo mediante el desarrollo de piezas de software de grano fino, que permiten adaptar los sistemas robóticos a las necesidades del cliente, por lo que se toman pequeñas funcionalidades de los sistemas robóticos y se personalizan según

los requerimientos. Este tipo de abstracción tiene ventajas permitiendo el desarrollo de una gran variedad de sistemas construidos a partir de múltiples piezas de software reutilizables. Aunque esto ha generado una problemática creciente en el dominio, que tiene que ver con la falta de apoyo a la reutilización, pues, no se genera soluciones efectivas a problemas reiterados de diseño arquitectónico [22], que pueden ser resueltos realizando procesos de abstracción de grano grueso, abarcando así problemas concurrentes en el diseño de los SR industriales. El documento de Gherardi y Brugali en [23] intenta resolver este problema desarrollando una arquitectura de referencia, que permite un mayor nivel de abstracción del dominio robótico, pero no ha tenido el impacto esperado, y los desarrolladores no han adoptado esta solución como propia.

Otro problema recurrente en este dominio está relacionado con la heterogeneidad de los componentes físicos de los robots, cualquier cambio en la configuración del sistema puede implicar una cadena de cambios en otros componentes, que pueden requerir herramientas específicas o tiempos de reprogramación largos, lo que genera incertidumbre en los desarrolladores, pues, estos dudan en realizar el proceso de reutilización si luego deben invertir grandes cantidades de tiempo en acondicionar el sistema a nuevos requerimientos o tareas. Otro problema surge con los sensores y actuadores ya que tienen diferentes funcionalidades, diferentes formas de interactuar con el entorno (pueden ser ultrasonidos, infrarrojos, u otros), y también tienen datos y protocolos de comunicación disímiles, por lo que las técnicas de reutilización tienen que adaptarse a esto para eliminar la variabilidad del software de este tipo de objetos. Aunque, una solución podría ser especificar estándares de interfaces software para este tipo de artefactos tecnológicos [104].

También, se pudo notar que numerosos investigadores que están realizando reutilización de software no se apropian de los resultados ya obtenidos por otros, debido a que la mayoría de las investigaciones se desarrollan mediante tecnologías ad-hoc, por lo tanto, se está realizando un retrabajo en el área que podría ser aprovechado en otros campos de igual importancia. Posiblemente, esto se debe a que las tecnologías desarrolladas no cumplen con las expectativas de los programadores y se debe mejorar la calidad de las herramientas finales.

Entre los principales desafíos del área, se encuentra la forma de elegir, integrar y alinear adecuadamente los componentes reutilizables que se desarrollan, y cumplir

con las funcionalidades requeridas teniendo en cuenta la particularidad del dominio. Entre estas particularidades que se deben sortear se encuentra la interacción física con el ambiente, la heterogeneidad de los elementos hardware, así como la alta confiabilidad que requieren este tipo de sistemas. Por lo tanto, se deben desarrollar estrategias de reutilización que cumplan con estos requerimientos del dominio [78], [22].

En el documento de Sun *et al.* en [105] se plantean dos desafíos que tiene la reutilización de software en los robots industriales, el primero tiene que ver con la complejidad de adaptar el software a los cambios que constantemente tienen los sistemas robóticos, pues, estos trabajan en distintos productos, acciones y tareas lo que hace que cambien frecuentemente las configuraciones e interacciones para abordar nuevas tareas. La consecución de este desafío implicaría la creación de sistemas robóticos más versátiles y una mejora en la velocidad de desarrollo software en este tipo de dispositivos. Mientras tanto, un segundo desafío plantea desarrollar software para múltiples plataformas (multiplataforma), es decir, realizar una correcta reutilización para evitar la reprogramación y poder utilizar piezas software para distintos tipos de fabricantes, esto evitaría que la obsolescencia de este tipo de sistemas no sea tan acelerada. Finalmente, un desafío importante en el área considerando las mejoras que produce la combinación de enfoques de reutilización de software en otros dominios, podría ir dirigido a desarrollar una solución que permita combinar enfoques de reutilización de manera intuitiva, lo que permitiría adoptar las mejores ventajas de cada uno de los enfoques utilizados.

Otros datos extraídos del mapeo sistemático de literatura

De los artículos encontrados se extrajo información que no tiene por qué estar relacionada con las preguntas de investigación planteadas, pero que es útil para el mapeo sistemático. Entre estas preguntas se encuentra el lenguaje de programación utilizado para el desarrollo de las herramientas finales y el nivel de automatización de las mismas. En la Figura 9, se observa que C++ y XML son los lenguajes predominantes (31% y 30% de uso respectivamente) para la construcción de aplicaciones de software para robots industriales, el uso de C++ tiene que ver con la utilización de ROS pues sus módulos están construidos en ese lenguaje de programación para una mejor interacción entre estos. En este sentido, XML también ocupa un lugar predominante y aunque no es un lenguaje de programación, se utiliza comúnmente para soluciones con el enfoque MDE. Por otra parte, JAVA fue utilizado

un 8.5% de las veces, sobre todo cuando se aplicó el enfoque OOR, aunque en la actualidad ha ido perdiendo fuerza debido a su auge en los años 90 cuando no existían alternativas a su portabilidad. También, C# se utilizó en algunos casos aislados, pero sin nada notable. Asimismo, algunos lenguajes han sido utilizados sólo en una investigación, como .NET, Esterel, Puma o Delphi. Finalmente, algunas herramientas se construyeron basándose en su lenguaje específico de dominio, lo que está relacionado con las investigaciones que utilizan el enfoque MDE.

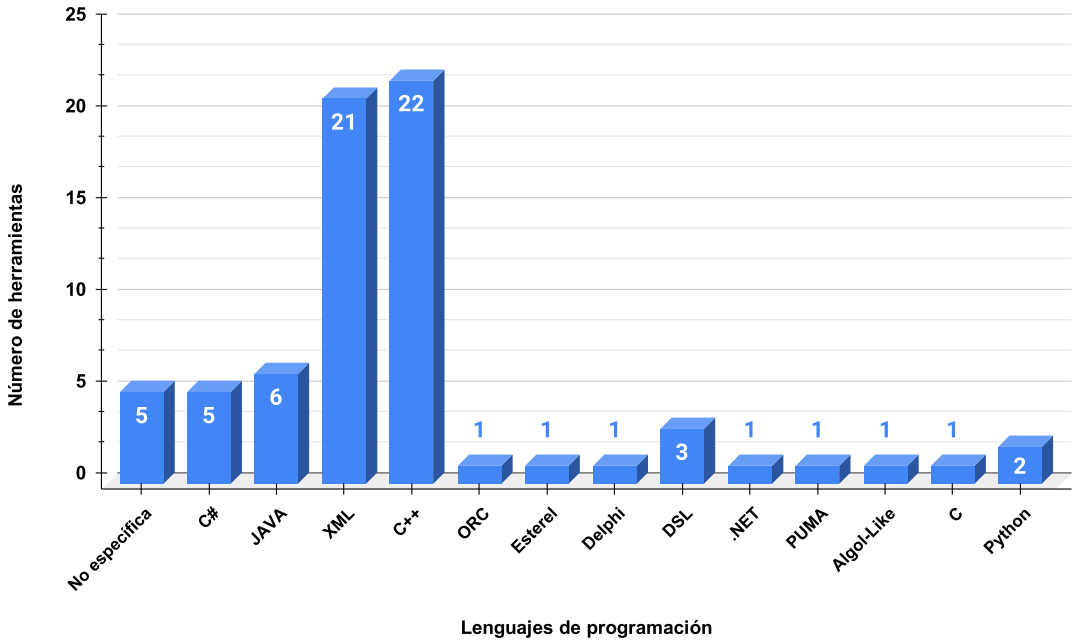


Figura 12. Distribución de los lenguajes de programación de las herramientas desarrolladas.
Fuente: Elaboración propia.

También se ha buscado encontrar el nivel de automatización que ofrecen las herramientas desarrolladas (se considera automatización el tiempo que se ahorra el programador en realizar o adaptar el software gracias a la herramienta desarrollada). El nivel más alto de automatización es para los generadores de código y el software de programación gráfica, porque el desarrollador no debe realizar programación manual o pocas modificaciones en el código final; este apartado corresponde al 20.5% de las herramientas desarrolladas. De igual manera, existen herramientas como los entornos de trabajo o las lógicas de intercambio que tienen un nivel de automatización medio, pues, el desarrollador de aplicaciones robóticas debe realizar un trabajo de acondicionamiento y generación del código sobre el que trabajar, pero con la ventaja de disponer de unas pautas predefinidas que agilizan el desarrollo del

software, este nivel corresponde al 49.4% de las herramientas analizadas. Finalmente, el nivel más bajo corresponde a las arquitecturas, metodologías e interfaces de Programación de Aplicaciones (API), debido a que el desarrollador debe generar el código desde cero, este nivel ocupa el 30.1% de las herramientas examinadas. Por lo tanto, la cuota más alta de automatización de herramientas corresponde al nivel medio, lo que puede estar relacionado con el hecho de que el área de estudio se encuentra en un momento de maduración tecnológica, y que seguramente en unos años se incrementara la creación de herramientas finales, como generadores de código o entornos de programación gráfica.

3.2. Trabajos destacados derivados de la revisión de literatura

La Ingeniería de Software y los Sistemas Robóticos Industriales son dos áreas que han tenido acercamientos puntuales en los últimos años, pero hasta hace poco no se ha empezado a popularizar el uso de técnicas como la reutilización de software (o Líneas de Productos de Software) en este ámbito, por lo que es un área relativamente nueva en la que se están desarrollando herramientas y/o recursos que permitan la adopción de manera generalizada de la reutilización. La conjunción de sistemas mecánicos, electrónicos, mecatrónicos y físicos en los robots industriales conlleva una serie de problemáticas de varias índoles y no sólo referente al software sino también a otras problemáticas como el control, la calibración de los sensores o el cálculo de la cinemáticas entre otros, por lo que es necesario desarrollar herramientas que ayuden a mejorar el desarrollo de sistemas software para robots industriales, para que los constructores se centren en otros desafíos de igual importancia. A continuación, se observa los principales aportes (investigaciones) al área de la reutilización de software en el dominio de los SR industriales, segmentados por el enfoque de reutilización que les antecede.

3.2.1. Trabajos relacionados entre la ingeniería dirigida por modelos y los SR industriales

Entre los estudios más destacados que involucran el MDE y la robótica industrial se encuentra el de Bruyninckx *et al.* en [87]. El autor sugiere que herramientas como ROS u OROCOS hacen reutilización de “grano grande”, esto significa que no se

preocupan por los problemas puntuales del área, por lo tanto, el trabajo proporciona a los desarrolladores un conjunto de pautas, metamodelos y herramientas que permitan estructurar el desarrollo de aplicaciones software para los SR industriales. Para esto, plantea seguir el enfoque de la OMG dividiendo el fenómeno en cuatro metaniveles, el más alto de la pila es M3 qué es representado por un metal-metamodelo Ecore proporcionado por la herramienta de modelización Eclipse Modelo Framework, el nivel M2 es la representación abstracta del modelo independiente de cualquier plataforma, en la capa M1 se representa el modelo abstraído del fenómeno, y por último, el modelo de M0 es una instancia sin definir, ya que, la herramienta desarrollada sólo busca a estructurar y formalizar el desarrollo de aplicaciones robóticas más no llevarlo a cabo o implementarlo, por lo que se puede aplicar ROS u otro framework.

Otro trabajo relevante que utiliza la MDE y se aplica en la robótica industrial es el realizado por Bubeck *et al.* en [16], en este se plantea que acoplar los robots industriales en las fábricas es un reto difícil, generalmente, este tipo de robots reemplazan a seres humanos con amplia experiencia, por lo tanto, estos dispositivos deben cumplir con la misma o más eficiencia. Para ello, se plantea realizar un brazo robótico doble que ayude en las tareas de ensamblaje de pequeñas piezas, utilizando el MDE como enfoque para lograr la reutilización del software de este tipo de robots. En primera instancia se plantea cuatro meta-niveles, siendo M3 el meta-metamodelo que es un archivo Ecore, posteriormente, M2 donde se describe un lenguaje específico del dominio para los frameworks donde se va a implementar la herramienta, luego M1 donde se representa el modelo abstraído de las herramientas y finalmente M0 que es la implementación de ROS. Los estudios encontrados permiten advertir que no necesariamente se utilizan las directrices expuestas de la OMG.

Otro trabajo es el de Lee *et al.* [106], donde se plantea que existe una problemática en el aumento de la complejidad del software requerido por los SR industriales, especialmente en la demanda de fiabilidad de las fábricas, además, exponen que existen acercamientos del MDE en el área, pero que estas soluciones no han sido probadas en ámbitos reales, por lo tanto, el autor propone desarrollar una nueva herramienta denominada UMIICA, la cual está basada en el MDE y permite la creación de aplicaciones de software de control industrial. Esta herramienta cubre todo el ciclo de desarrollo de software para robots, iniciando por el levantamiento de

requisitos, la generación de código fuente y las pruebas finales. También afirman que UMIICA ha demostrado generar garantías de rendimiento y está preparada para trabajar en procesos de producción reales en la industria. En este trabajo no se implementa ningún meta-nivel que lo relacione con la cadena de herramientas MDE, por lo tanto, esto permite inferir que, aunque se están realizando desarrollos con la ingeniería dirigida por modelos, no siempre se sigue el estándar sugerido por la OMG.

Estos estudios nos permiten deducir que el enfoque MDE es una de las estrategias de reutilización de software que más casos de éxito han tenido a lo largo del tiempo, por lo que la inserción de los modelos en los SR industriales ha permitido mejorar (acelerar) el proceso de desarrollo de software para estos sistemas, debido a que, se simplifica un dominio complejo, constituyéndose como una alternativa viable para implementar la reutilización de software en el dominio.

3.2.2. Trabajos relacionados entre la Ingeniería de software basada en componentes y los robots industriales

Uno de los trabajos más destacados en el área es el de Wei *et al.* en [89], donde realiza la descomposición de un SR industrial en dos componentes generales, el primero denominado “control del sistema robótico” (referente al sistema software) y el segundo el “sistema controlado” (hardware del dispositivo). El autor propone un middleware que permite la interacción de los dos componentes para mejorar el proceso de reutilización, para ello subdivide los dos componentes en elementos más pequeños. El primero nombrado “componente atómico”, que es el que interactúa con el hardware, luego se encuentra el “componente compuesto” que es la unión de varios componentes atómicos que permiten la reutilización de grano fino (por ejemplo, ayudar a manejar la variabilidad de los sensores o los actuadores). Posteriormente, se tiene el “componente algorítmico” que encapsula todos aquellos algoritmos necesarios para que el robot funcione adecuadamente (cinemática inversas y directas, programación de trayectorias, filtros de Kalman, entre otros elementos), por último, está el “componente aplicación” que es donde interactúan todos los componentes mencionados. Estos subcomponentes pueden relacionarse con otros, permitiendo la comunicación multidireccional entre los elementos del sistema, y se alojan en un repositorio de componentes accesible al programador para que pueda acceder a los recursos previamente creados. Finalmente, el autor

concluye que modularizar los SR industriales es un aporte valioso que permite mejorar la construcción de estos sistemas.

En el trabajo de Jung y Kazanzides en [107], se plantea que la reutilización de software en el dominio de los SR es una realidad, sin ser totalmente acogido por los desarrolladores. En el trabajo, se detectó una problemática centrada en la seguridad, debido a que, son sistemas cada vez más complejos que interactúan en espacios con los seres humanos, por lo tanto, la seguridad es un atributo para tener en cuenta. Para esto se plantea una arquitectura basada en componentes en el que la seguridad ya no es un atributo, sino que es un elemento visible, reutilizable y verificable. A partir de esta idea, desarrollan SAFECASS que propicia la reutilización de la experiencia y el conocimiento en la seguridad aplicándolo a los robots médicos que trabajan en áreas críticas. Luego, plantean modularizar el sistema definiendo un modelo de componentes genérico y abstracto, representando cualquier sistema sin profundidad en detalles técnicos, denominándose “componente genérico” que consta de elementos estructurales y semánticos. El autor alude que los elementos estructurales son un conjunto de componentes, interfaces y protocolos de comunicación que se comunican entre sí, mientras que, la semántica es la captura y representación sistemática del estado operativo de los componentes estructurales. La combinación de estos dos componentes permite definir el estado actual del robot y según las variables que entran a la herramienta se definen tres estados principales: normal, advertencia y error. Basados en estos tres estados se define una máquina de estados finitos que es impulsada por los eventos anteriormente descritos. Por último, aunque la propuesta se ha probado en un estudio de caso, no se ha probado en un entorno real.

3.2.3. Trabajos relacionados entre la arquitectura orientada a servicios y los SR industriales

Entre los principales trabajos que integran el enfoque SOA con la robótica industrial se encuentra la investigación de Rudorfer *et al.* en [91], donde se intenta mejorar el proceso de programación de estos dispositivos mediante una interfaz intuitiva de arrastrar y soltar, el autor encapsula las principales funcionalidades del SR industrial en servicios que se comunican a través de una interfaz previamente definida, la mejora es prominente debido a que si se quiere configurar una u otra funcionalidad del sistema robótico simplemente se debe arrastrar y soltar la funcionalidad

encapsulada, permitiendo la comunicación con otros servicios mejorando así la velocidad de desarrollo del software de los robots industriales. Otro aporte novedoso de esta herramienta es que el sistema de arrastrar y soltar se realiza mediante realidad aumentada, por lo que, si el usuario realiza el movimiento de agarrar una pieza y soltarla, el robot puede repetir esta acción, por tanto, esta herramienta combina dos enfoques novedosos, siendo la arquitectura orientada a servicios y la comunicación multimodal, que son tecnologías emergentes en la construcción de sistemas robóticos. Finalmente, el desarrollo es probado mediante un estudio de caso, pero el autor menciona que una de las principales problemáticas de la propuesta es la falta de precisión y grados de libertad para posicionar el objeto, por lo tanto, es un área en el que se debe trabajar.

Otro trabajo importante en el área es el de Chen *et al.* en [92], en este caso, el autor hace uso del enfoque SOA con una orientación hacia la computación en la nube, en este trabajo por primera vez se menciona el término “robot as service (RaaS)”, el cual está directamente relacionado con el encapsulamiento de servicios alojados externamente y que son accesibles mediante un protocolo web. Esta variación permite a los desarrolladores acceder, eliminar y/o configurar estos elementos de software en línea, también se diferencian en que la nube RaaS contiene aplicaciones por lo tanto cualquier persona puede acceder a nuevas funcionalidades que se encuentren disponibles. La unidad RaaS es el publicador de los servicios y el cliente puede buscar las aplicaciones disponibles en esta unidad de directorio. La herramienta desarrollada permite trabajar con frameworks como ROS u OROCOS, y múltiples lenguajes de programación. Los autores presentan un caso específico para hardware genérico basado en arquitectura Intel, que, gracias al alto nivel de abstracción de la herramienta, ésta puede trabajar con múltiples configuraciones.

3.2.4. Trabajos relacionados entre las líneas de productos de software y los robots industriales

Las líneas de productos de software en conjunción con los SR industriales es un campo poco explorado y con aportes limitados, esto se debe a que es una técnica menos extendida por la complejidad en su institucionalización en las organizaciones, por lo tanto, es un campo de estudio en el que se pueden hacer múltiples acercamientos y mejoras.

En el trabajo de Gherardi *et al.* en [108], plantean la creación de una SPL basada en la nube. El autor esboza que el gasto computacional que requieren los robots para su funcionamiento es alto, ya que, estos dispositivos son cada vez más complejos y exigentes, por lo tanto, plantean fusionar la robótica en la nube y las líneas de productos de software. La robótica en la nube es una solución novedosa que permite reducir el gasto computacional que exigen estos dispositivos, mientras que, las SPL admiten a los usuarios desplegar y configurar complejos sistemas robóticos sin tratar problemas de bajo nivel o tecnicismos que se implementan a la tecnología utilizada. Para ello, utilizan la herramienta Rapyuta's que es una plataforma robótica de código abierto que logra que los procesos computacionales se hagan en la nube utilizando el protocolo de comunicación WebSocket. Los autores proponen un depósito de conocimiento en el que se tiene información sobre modelos de objetos, mapas del medio, recetas de acción y experiencias de otros robots. De este modo, crean una arquitectura de referencia da soporte a la construcción de software para robots industriales. La variabilidad de la línea se modela bajo tres parámetros, el primer modelo define la variabilidad de la arquitectura de referencia en términos de elementos reutilizables y variables, incluyendo entornos informáticos, componentes y características. Estos elementos reutilizables se utilizan en aplicaciones parecidas, mientras que, los elementos variables son rasgos específicos de una configuración. El segundo modelo define la variabilidad de las funcionalidades por medio de un modelo de características. Y el tercer modelo describe el despliegue de una aplicación específica resolviendo la variabilidad de la arquitectura propuesta.

Brugali y Hochgeschwender en [85] expresan que los sistemas robóticos son cada día más importantes en la vida cotidiana, por lo que es fundamental desarrollar nuevos enfoques arquitectónicos para que el software de estos sistemas se cree de forma más rápida y eficiente. Los autores mencionan que las SPL han demostrado ser el enfoque más efectivo para enfrentar estos desafíos, ya que, cada nueva aplicación construida significa un activo de software para futuros desarrollos. También indican que este enfoque es una inversión estratégica para las organizaciones que quieren proporcionar valor al cliente, permitiendo diversificar productos a un coste mínimo. Con base en lo anterior, se proponen tres enfoques para las SPL, el enfoque proactivo que consiste en diseñar una SPL para soportar la totalidad del alcance de los productos, seguido por el enfoque extractivo que orienta la reingeniería de los sistemas preexistentes, y finalmente, el enfoque reactivo que

analiza el diseño de la infraestructura de software para desarrollar nuevos productos mediante un *feature model*. Estos productos son representados mediante un árbol estructurado donde hay elementos imprescindibles y variables. También, se desarrolla la herramienta Hiperflex que utiliza el MDE como base para abstraer los robots y apoya el ciclo de desarrollo de software para estos dispositivos. Finalmente, implementan la SPL en un caso realista donde se presentan buenos resultados, concluyendo que la estructuración del desarrollo de software para robots industriales es beneficioso para el dominio.

Algunos autores expresan que ROS es la principal alternativa para realizar la reutilización en este dominio, pero a raíz de esto, ha surgido una nueva problemática, ya que las exigencias arquitectónicas y de diseño software son nulas, generando procesos no estandarizados y por lo tanto menos reutilizables [109]. Por lo que han surgido múltiples alternativas, como CORBA Component Model (CCM) o la norma IEC 61499, que buscó especificar la estructuración de los recursos reales de un sistema, proporcionando y permitiendo el intercambio de datos. Estas son propuestas interesantes que no han sido adoptadas como se esperaba, por lo tanto, para resolver esta problemática se propone una SPL que estructura la construcción de los sistemas robóticos y amplía el ciclo de vida de estos. En esta propuesta, los autores proponen administrar la variabilidad de dos formas, la primera integrando y la segunda ortogonal. Además, sugieren que se sigan tres modelos para la implementación de la línea de productos. En primer lugar, el modelado de la SPL dónde se debe establecer la variabilidad y la arquitectura. En segundo lugar, el modelo de componentes donde se describe cada uno y el modelo de la arquitectura. Posteriormente, el modelo de aplicación donde se representa la implementación. Por último, la línea propuesta no se implementa y tan sólo es un modelo metodológico que se puede aplicar en marcos de trabajo como ROS.

Los estudios de la implementación de las SPL en los SR industriales permiten inferir que es un enfoque de reutilización que es aplicable al dominio, ya que, estas propuestas han intentado acelerar el desarrollo de software para estos dispositivos, permitiendo que una mayor aceptación para la adopción de la reutilización en el dominio de los SR industriales. Adicionalmente, este enfoque de reutilización tiene una ventaja sobre los demás y es que básicamente, las herramientas generadas tienen un alto nivel de automatización, ahorrando así a los desarrolladores más tiempo en la escritura de software informático. Asimismo, se detectó que las SPL se

están utilizando generalmente con el enfoque MDE, dando lugar a herramientas que adoptan lo mejor de ambos enfoques.

3.2.5. Investigaciones que relacionan la IS con Arduino en el dominio de los sistemas robóticos industriales

Específicamente, en el desarrollo de software de la plataforma Arduino, las metodologías de ingeniería de software son prácticamente inexistentes salvo algunas excepciones que no han dado los frutos esperados, generando que la reutilización se haga de manera desordenada, poco planificada y poco efectiva. La reutilización en esta plataforma se manifiesta mediante librerías y código fuente que a menudo trata de resolver problemáticas ya solucionadas, generando retrabajo en áreas donde no es necesario. Estas dificultades se deben a que existe una gran comunidad de desarrolladores que crean sus soluciones de forma independiente, sin buscar sistematizar el proceso de reutilización y mejoramiento de código para este ecosistema, además, la gran mayoría no tiene las competencias ni los conocimientos para realizar este proceso de mejoramiento en el desarrollo de software. Cabe destacar que, hasta donde los autores saben, no se evidenció investigaciones que hagan uso de la reutilización de software mediante SPL en el dominio de los robots industriales con Arduino, por lo que existe una oportunidad de investigación que pueda aportar mejoras en el área. Por esta razón, se realizó una búsqueda de mayor espectro en la que se intentó relacionar la reutilización de software en el dominio general de Arduino, logrando encontrar algunas investigaciones relevantes.

Este es el caso de la revisión de Geraldi *et al.* en [110], donde se presenta una visión general sobre cómo se están aplicando las SPL en dispositivos con microcontroladores para el internet de las cosas, donde las implementaciones se hacen en diferentes tipos de dispositivos como Raspberry Pi y Arduino. Los resultados evidencian la carencia en la especificación sistemática y detallada de las SPL que garantice la calidad de los productos derivados, así como directrices de adaptación para su uso y adopción. Asimismo, Bonfanti *et al.* en [111], plantean un generador de código con MDE, partiendo de las especificaciones del usuario utilizando máquinas de estado abstractas. Este método guía el desarrollo de software para sistemas de tiempo real, basándose en el principio de refinamiento, capturando así la semántica de los sistemas hasta el nivel deseado. Como parte de la propuesta, se desarrolló la herramienta Asm2C++ utilizando el MDE y la transformación de

modelo a texto (M2T) para generar código para microcontroladores. Además, también se realiza una prueba de concepto donde se obtuvo que la propuesta permite generar prototipos rápidos y diseñar sistemas empotrados fácilmente. Similarmente, Ataide *et al.* en [112], utilizan el enfoque MDE para transformar modelos a código fuente para diferentes microcontroladores, la propuesta consiste en un conjunto de herramientas y modelos de dominio que se dividen en submodelos, los cuales son entradas a una red de Petri para la generación de código. También, se realiza una prueba de concepto con dos ejemplos, el primero corresponde a un único dominio temporal, es decir, un solo microcontrolador, mientras que, el segundo ejemplo presenta tres dominios temporales, donde se expone cómo generar código para distintos microcontroladores.

Los trabajos anteriores exponen distintas representaciones de la reutilización de software en Arduino, con diferentes visiones, perspectivas y enfoques donde la reutilización se hace de forma desordenada, escasamente planificada y en algunos casos, poco efectiva. Además, estos no hacen uso específico de las SPL en los SR industriales con Arduino, siendo este el mayor aporte que pretende brindar esta investigación.

Tabla 5. Preguntas y posibles respuestas para diferenciar los trabajos encontrados.

ID	PREGUNTA	POSIBLES RESPUESTAS
P1.	¿Nivel de automatización de la herramienta desarrollada?	• Alto (A) • Medio (M) • Bajo (B)
P2.	¿Tipo de enfoque de reutilización de software en la investigación?	• Ingeniería dirigida por modelos (MDE) • Ingeniería de software basada en componentes (CBSE) • Arquitectura Orientada a Servicios (SOA) • Línea de productos de software (SPL)
P3.	¿Tipo de herramienta desarrollada para la investigación?	• Framework (FWK) • Herramienta de programación gráfica • Arquitectura de software (SA) • Plataforma Informática (PI) • Generador de código (GC)
P4.	¿La solución propuesta se centra en la plataforma Arduino?	• Si • No
P5.	¿La solución propuesta se centra en los SR industriales?	• Si • No
P6.	¿La herramienta creada es software de código abierto?	• Si • No
P7.	¿La solución propuesta se centra en la academia o en la industria?	• Academia (A) • Ambos rubros • Industria (I)

En la Tabla 5 se plantean una serie de preguntas y posibles respuestas para diferenciar los trabajos expuestos anteriormente.

Se le considera nivel de automatización al tiempo que se ahorra el programador en realizar o adaptar el software gracias a la herramienta desarrollada.

Tabla 6. Comparativa de los principales estudios analizados en la revisión de la literatura.

AUTORES	P1	P2	P3	P4	P5	P6	P7
Bruyninckx *et al.*	M	MDE	FWK	NO	SI	SI	Ambos
Bubeck *et al.*	M	MDE	FWK	NO	SI	NO	Ambos
Lee *et al.*	A	MDE	FWK	NO	SI	SI	I
Wei *et al.*	M	CBSE	FWK	NO	SI	NO	A
Jung *et al.*	A	CBSE	FWK	NO	SI	SI	Ambos
Rudorfer *et al.*	A	SOA	PG	NO	SI	NO	A
Chen *et al.*	B	SOA - CC	SA	NO	SI	SI	A
Gherardi *et al.*	A	MDE/ SPL	PI	NO	SI	SI	Ambos
Brugali *et al.*	A	MDE/ SPL	SA	NO	SI	NO	A
Heikkila *et al.*	A	MDE/ SPL	FWK	NO	NO	NO	-
Ataide *et al.*	A	MDE	GC	SI	NO	SI	Ambos
Bonfanti *et al.*	A	MDE	GC	SI	NO	NO	A
Solis *et al.*	A	MDE/ SPL	GC	SI	SI	SI	A

En la Tabla 6 se expone una comparación entre las características de las investigaciones examinadas y el presente trabajo, donde se discierne que la propuesta planteada en este documento es distinta a cualquiera de las analizadas, debido a que aborda la reutilización de software en el dominio de los SR industriales con Arduino, para que se pueda propiciar la utilización de estos enfoques de reutilización desde la academia y poder naturalizar el uso de estas técnicas en el dominio.

3.3. Conclusiones derivadas de la revisión de literatura

La reutilización de software se ha transformado en una parte importante de la construcción de los sistemas robóticos industriales [84], por lo que se están realizando grandes esfuerzos para agilizar y mejorar el desarrollo de software de este tipo de dispositivos. Los resultados permiten establecer que se trata de un campo de investigación en auge, con una tendencia positiva en la frecuencia de publicaciones anuales. Según lo encontrado en la investigación los enfoques de reutilización de software más utilizados en el dominio son MDE, CBSE y SOA.

Los productos de investigación más propuestos en el dominio son los marcos de trabajo y las soluciones arquitectónicas. Por lo tanto, se están desarrollando elementos de base que normalmente soportan enfoques y otros productos, lo que proporciona bases sólidas para continuar con las investigaciones en esta área.

El mapeo sistemático determinó que la computación en la nube y la comunicación multimodal son soluciones emergentes, y que desempeñarán un papel importante en el futuro para mejorar el proceso de desarrollo software para SR industriales. Entre las principales ventajas, la primera intenta eliminar las limitaciones de hardware y ayudar al trabajo colaborativo entre dispositivos robóticos [113], mientras que, la segunda permite la programación y reprogramación de estos sistemas de manera rápida y por personal no experto [114].

Entre los principales desafíos de investigación en el área se encontraron la combinación de diferentes enfoques de reutilización de software, para obtener las ventajas de cada uno y optimizar el proceso de desarrollo [115]. Esto está relacionado con la abstracción e integración de las piezas de software reutilizables, para responder a los nuevos requisitos de los sistemas robóticos.

Los enfoques de reutilización más utilizados están relacionados con la abstracción del sistema y sus funcionalidades para lograr la reutilización de software. El proceso de composición también ha sido clave para mejorar y adaptar la reutilización en este dominio, un claro ejemplo es el enfoque CBSE que se está convirtiendo en una forma natural de representar los sistemas robóticos y sus componentes, mientras que, el enfoque SOA proporciona un mecanismo de comunicación para los componentes a un nivel de abstracción más cercano a la implementación, logrando cierta independencia del hardware y los lenguajes de programación. El uso de la descomposición y la abstracción no son procesos que deban realizarse por separado y, de hecho, puedan mejorarse mutuamente como se ha demostrado en apartados anteriores. El MDE es justamente el enfoque que busca una separación de preocupaciones a través de modelos en diferentes niveles de abstracción. Así, se puede dar una propuesta novedosa de reutilización combinando enfoques, permitiendo que los servicios empaqueten componentes, y simultáneamente, los componentes, los servicios y sus interacciones puedan ser abstraídos como modelos para que el desarrollador trabaje con elementos abstractos, y genere soluciones

concretas sin tener que lidiar con detalles técnicos específicos de las tecnologías subyacentes.

Los resultados de este mapeo sistemático permiten establecer los múltiples acercamientos entre las técnicas de reutilización de software y el dominio de los SR industriales. ROS puede considerarse una alternativa a tener en cuenta como punto de referencia para que los desarrolladores integren diferentes enfoques de ingeniería de software en los sistemas robóticos industriales [116]. La mejor forma de adoptar y establecer estas buenas prácticas de programación podría ser incluir a ROS en la oferta académica de las instituciones educativas para que los programadores se acondicionen a la herramienta desde las primeras experiencias. También, es importante indicar que se debe establecer una curva de aprendizaje menos pronunciada para ROS o para cualquier otro desarrollo, que busque ser punta de lanza para implementar completamente estos enfoques [87], de manera que los desarrolladores puedan adoptar estas herramientas de forma natural y abstraer aspectos técnicos y específicos de las tecnologías de bajo nivel.

Finalmente, hasta donde el autor sabe, la reutilización de software en el dominio de los SR industriales con Arduino es limitada y poco eficaz, por lo que existe un vacío en el conocimiento para estos dispositivos específicos. El desconocimiento en esta área es una problemática importante, debido a que, los robots industriales con Arduino se utilizan en la academia para entender las bases teóricas y prácticas en la construcción de estos dispositivos electromecánicos. Por tanto, se hace evidente la necesidad de incentivar y aplicar enfoques de reutilización que permitan mejorar el proceso de desarrollo de software para robots industriales con microcontroladores, de manera que los desarrolladores (estudiantes) puedan beneficiarse de la reutilización planificada, además entiendan y se familiaricen con estos enfoques de IS desde su formación académica, permitiendo así que la reutilización en la industria sea más factible en estos dispositivos cuando sea necesario.

3.4. Resumen de los resultados obtenidos en el capítulo

En este capítulo se expone una revisión de la literatura sobre los enfoques de reutilización de software aplicados a los robots industriales, así como una revisión

detallada de las investigaciones más influyentes en el dominio. Los principales productos obtenidos en el presente capítulo son:

- Artículo de investigación **publicado** denominado “Reutilización de software en la robótica industrial: un mapeo sistemático”, publicado en la Revista Iberoamericana de Automática e Informática industrial (RIAI), indexada por Scopus.
- Participación como ponente en el simposio nacional de maestrías y doctorados 2020 por la sociedad colombiana de computación.
- Caracterización de los enfoques de reutilización en los SR industriales, donde se encontró que las SPL podrían ser una alternativa factible para propiciar la reutilización de software en el dominio de los robots industriales con Arduino, para naturalizar este proceso cuando sea el caso en la industria.
- Identificación de ROS como alternativa principal de reutilización de software en los SR industriales con controladores, siendo este marco de trabajo la herramienta más difundida en el dominio y, por lo tanto, se sugiere la enseñanza de éste en la academia para propiciar la reutilización desde etapas tempranas.
- Se encontró que los enfoques de reutilización de software en robots industriales con controladores han tenido problemas para resolver las dificultades de diseño arquitectónico en el dominio, por lo que se propone combinar dos o más enfoques para obtener las ventajas de cada uno.

Capítulo 4

IRArduino-SPL: Línea de productos de software para robots industriales con Arduino

En este capítulo se aborda la planificación, desarrollo, construcción y refinamiento de una SPL para microcontroladores Arduino en el dominio de la robótica industrial con un enfoque académico (IRArduino-SPL). Para ello, se siguieron los procesos de Ingeniería de Dominio e Ingeniería de Aplicación con un enfoque generativo, donde se estudiaron y analizaron las estructuras de programación empleadas habitualmente por los robots industriales con Arduino (modelamiento de dominio). Luego, se identifican los puntos variables y comunes del dominio y se propone un modelo de características. Posteriormente, se realiza un refinamiento de la propuesta de reutilización donde se recogen las experiencias e insumos de la primera experiencia IRArduino-SPL, añadiendo elementos a la Ingeniería de Dominio y a la Ingeniería de Aplicación. Además, se replantea el alcance de la propuesta de reutilización en base a las opiniones de expertos en el dominio de la robótica utilizando una adaptación del método CoMeS-SPL [133] así como una encuesta en línea para desarrolladores expertos en robótica. Posteriormente, se presenta el desarrollo e implementación de las actividades fundamentales de una SPL, como el desarrollo de activos núcleo, el desarrollo de productos (utilizando un enfoque basado en activos núcleo), y la administración de la estrategia de reutilización, acogiendo algunas pautas y directrices de las propuestas metodológicas de PuLSE [30], Small-SPL [28], y el framework del SEI [29].

4.1. Introducción al capítulo

La Ingeniería de Líneas de Productos de Software para dominios específicos normalmente se enmarca bajo dos grandes actividades, el primero es denominado Ingeniería de Dominio, mientras que el segundo es la Ingeniería de Aplicación. El correcto desarrollo de estas dos actividades esenciales permite la creación de

productos de software a partir de un conjunto compartido de activos núcleo que utilizan un medio de producción común [117].

Específicamente, la investigación siguió las directrices propuestas en Small-SPL [28], así como, algunas pautas sugeridas por el framework SPL del SEI [29] (Figura 13), para realizar la Ingeniería de Dominio y la Ingeniería de Aplicación, al igual que las actividades esenciales en este tipo de estrategias de reutilización de software. Small-SPL es un modelo orientado a facilitar la adopción de las SPL en pequeños grupos de trabajo mediante la reutilización de activos de proceso. Por otra parte, las pautas sugeridas por el SEI son muy populares por la comunidad de desarrolladores que acoge este enfoque de reutilización, porque describe las acciones y habilidades necesarias a desarrollar para beneficiarse de las SPL. Por las razones anteriores, las directrices mencionadas se utilizan para lograr la consecución de una línea de productos de software para robots industriales con Arduino.

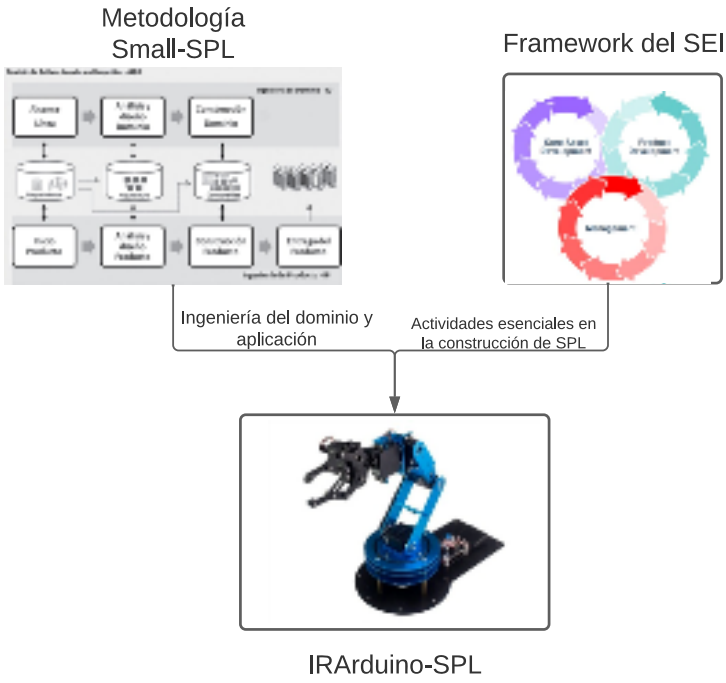


Figura 13. Aportes de los referentes metodológicos en la construcción de IRArduino-SPL.
Fuente: Elaboración propia.

4.2. Ingeniería del dominio

El objetivo de esta investigación es mejorar el proceso de construcción de software de los SR industriales con Arduino, por lo que un proceso fundamental para

implementar este tipo de estrategia de reutilización radica en analizar y determinar el dominio de aplicación, establecer las abstracciones relevantes y la variabilidad en el dominio, entre otras tareas. Por esta razón, se realizó un modelamiento del dominio, que es una representación formal de un sistema con conceptos, roles, individuos y reglas típicamente basados en una lógica de descripción, es decir, es un modelo que se centra en las abstracciones relevantes en un dominio específico [118].

4.2.1. Modelamiento del dominio

Teniendo en cuenta lo anterior, se determinó cómo se estructura la programación de los sistemas robóticos industriales con Arduino, para encontrar elementos de software relevantes en la construcción de este tipo de sistemas, es decir, se buscaron los principales modismos de programación[1] (*Idioms*, en inglés) en la plataforma. Con esto, se consultó en la literatura metodologías para la determinación y análisis de *idioms* de programación. Esta búsqueda fue infructuosa porque no se encontraron técnicas o métodos establecidos para este fin, aunque se han desarrollado herramientas que apoyan la determinación de los modismos de programación. Este es el caso de Haggis [119], que es un sistema de minería de código que determina los principales modismos para varios lenguajes de programación utilizando técnicas estadísticas avanzadas. Asimismo, existe code2vec [120], que es un modelo neuronal que segmenta la semántica del código en vectores para entender cómo se estructura la programación de estos segmentos informáticos. Desafortunadamente, las investigaciones encontradas no soportan el lenguaje de programación Arduino o no son accesibles al público en general, por lo tanto, se decidió crear un algoritmo informático que funja como un analizador lexicográfico para encontrar cómo se estructura la programación en los SR industriales en la plataforma.

Analizador lexicográfico y análisis semántico del dominio

Los analizadores lexicográficos son algoritmos que reciben como entrada el código fuente de un programa (algoritmos de SR industriales con Arduino) y producen una salida compuesta por tokens o símbolos (componentes léxicos) según reglas o

[1] Un **modismo de programación** (*idioms*) es un grupo de fragmentos de código que comparten una función semántica equivalente y que se repite con frecuencia en los proyectos de software [119].

patrones predeterminados. Estos tokens permiten un análisis más eficiente del código fuente ingresado, ya que, los símbolos se agrupan en categorías léxicas recurrentes en los algoritmos analizados [121]. Normalmente, los analizadores lexicográficos se utilizan en los compiladores (también en otras áreas) para especificar y determinar las instrucciones (palabras reservadas del lenguaje) que siguen ciertos patrones dentro de una cadena de caracteres a reconocer [122].

En esta investigación, el desarrollo del analizador lexicográfico (Apéndice B) se realizó en el lenguaje de programación Python utilizando el módulo “re” de su biblioteca estándar, para facilitar las operaciones con expresiones regulares en los códigos analizados. El script recibe el código fuente de un archivo de texto (código fuente SR industriales), y luego el algoritmo encuentra, determina y especifica las palabras reservadas del lenguaje de programación Arduino, para establecer cuáles son las instrucciones más recurrentes utilizadas en estos dispositivos. Puntualmente, el analizador lexicográfico toma las cadenas de caracteres del archivo de texto y las descompone en una unidad más pequeña denominada categoría léxica, en este punto el analizador establece qué tipo de símbolos (pueden ser identificadores, variables, funciones, entre otros) están ingresando desde las distintas cadenas de caracteres, además, el algoritmo utiliza la función “*re.escape()*” para separar y desestimar cualquier expresión con operadores o caracteres alfanuméricos especiales. Una vez realizado este procedimiento, se clasifica cada una de las palabras del código fuente según lo expresado en [123], donde se plantea una guía de referencia de Arduino por parte de sus desarrolladores, en la que se indica que la sintaxis para estos microcontroladores está dividida en tres grandes grupos: estructuras, valores (variables y constantes) y funciones, que contemplan todas las palabras reservadas del lenguaje de programación. Esta operación se realiza con la función “*re.compile()*”, que busca y determina las coincidencias entre un patrón dado (palabras reservadas) y las cadenas de caracteres del código fuente. Posteriormente, el software contabiliza las coincidencias encontradas y las presenta segmentadas según los grupos mencionados, utilizando la función “*re.findall()*”. Por último, en la siguiente figura se puede apreciar el diagrama de flujo del algoritmo descrito.

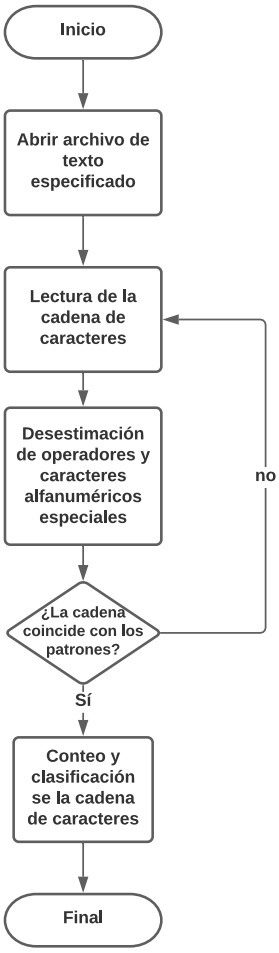


Figura 14. Diagrama de flujo del analizador lexicográfico desarrollado en Python.
Fuente: Elaboración propia.

En esta línea, y para una mayor precisión en la forma en que se está estructurando el código fuente en los SR industriales con Arduino, se utilizaron dos herramientas de terceros como son Code Comparer [124] y SemanticMerge [125] para efectuar un análisis semántico sobre la programación de estos dispositivos, buscando bloques de instrucciones comunes en la plataforma. Para ello, las citadas herramientas comparan las estructuras e instrucciones de programación no sólo por su posición, sino también por su significado en el código fuente de los robots, determinando bloques de instrucciones similares entre los distintos algoritmos analizados.

Modelado del dominio de los SR industriales con Arduino

Posteriormente, se recopilaron 10 códigos fuente de SR industriales con Arduino accesibles a través de la web y se analizaron utilizando las herramientas descritas

(Figura 15). El análisis a nivel general expuso que un acercamiento a la reutilización de software en Arduino es la plantilla inicial denominada *BareMinimum*, que representa el código fuente mínimo necesario para compilar un programa en la plataforma. Esta plantilla utiliza abstracciones, llamadas a funciones y librerías junto con una Capa de Abstracción de Hardware (en inglés, Hardware Abstraction Layer o HAL) que brinda una base para que los desarrolladores trabajen. Lo anterior es un acercamiento a la reutilización de software en la plataforma, además, Arduino permite la definición de *templates* desarrollados por la comunidad para su uso en los programas, que también son mecanismos de reutilización.

Teniendo en cuenta lo anterior, se puede inferir que, aunque la reutilización de software en Arduino no está generalizada, si existen las herramientas para realizar esta actividad. Sin embargo, desafortunadamente para el dominio de los SR industriales con Arduino *BareMinimum* es insuficiente, porque omite elementos de software relevantes (cinemáticas o trayectorias) o no brinda la asistencia necesaria (interfaces de usuario) para la construcción de software en robots industriales, además, no es adaptable al dominio donde se implementa el código. Por lo tanto, se podrían utilizar *templates* centrados en estos dispositivos para facilitar la reutilización en el dominio.

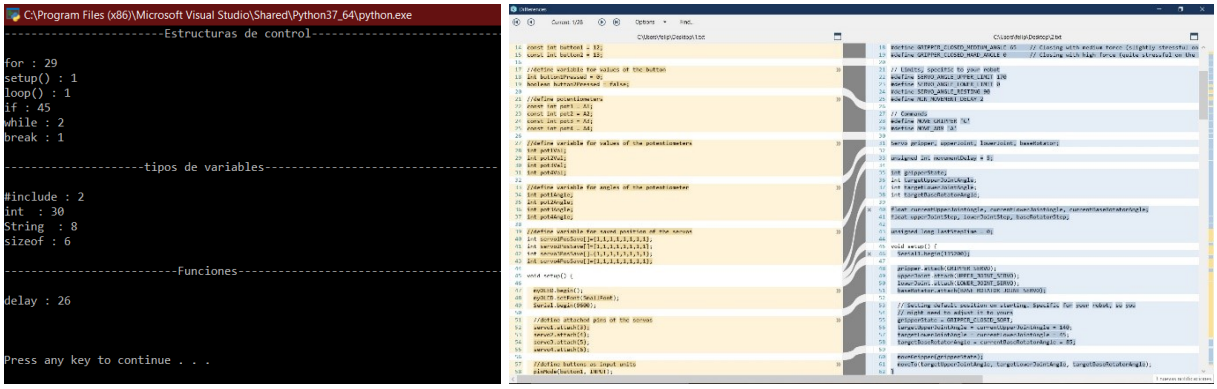


Figura 15. (a) Análisis lexicográfico realizado sobre el código fuente de un SR industrial con Arduino. **(b)** Resultados de un análisis semántico realizado sobre dos códigos fuente de SR industriales. **Fuente**: Elaboración propia.

Otro resultado que se obtuvo mediante el análisis con el script desarrollado (analizador lexicográfico) y las herramientas Code Compare y SemanticMerge, fue que se determinó que existen tres bloques de sentencias frecuentes en los códigos analizados, que están en sintonía con lo propuesto por la sintaxis de Arduino. En concreto, se puede hablar de un primer bloque donde se ubican las instrucciones que

declaran variables, realizan la inclusión de librerías y el llamado de cabeceras; la posición de estas sentencias se encuentra generalmente antes del *void setup()*. En el segundo bloque de sentencias, se encontraron todas aquellas relacionadas con la inicialización de variables y puertos, así como la declaración de objetos de librerías; este bloque de código se encuentra generalmente dentro del *void setup*(). El tercer bloque está relacionado con la lógica adyacente al funcionamiento del robot en Arduino; normalmente, el código de esta sección determina las trayectorias, los movimientos y el funcionamiento de los sensores y actuadores; su posición en el código fuente está dentro del *void loop()*. También, se determinó que los códigos de los SR industriales hacen referencia a dos importantes bloques de sentencias de programación, el primero relacionado con los algoritmos que gobiernan el robot (denominado, bloque de algoritmos), y el otro referido al control de los dispositivos hardware (denominado, bloque de hardware).

El bloque de algoritmos contiene las sentencias relacionadas con la lógica adyacente al funcionamiento del dispositivo, generalmente expresada a través de un patrón de acción y reacción, es decir, para cada estímulo percibido por los sensores existe una reacción por parte de los actuadores. Este bloque carece de elementos de suma importancia para la construcción de software en brazos robóticos, como la utilización de cinemáticas para la determinación de trayectorias, o la inclusión de una interfaz de usuario (GUI), esto refleja un vacío en la construcción de este tipo de robots con Arduino. El segundo bloque de sentencias pertenece a los controladores de hardware, que son el código implementado para el correcto funcionamiento de los dispositivos físicos, en este sentido, los desarrolladores se esfuerzan para que los sensores y actuadores funcionen acorde y en sincronía con el bloque de algoritmos del dispositivo. Las referencias al hardware en los códigos analizados se supeditan a dos tipos de dispositivos electrónicos, los sensores, y los actuadores, que son elementos fundamentales en el funcionamiento de este tipo de robots.

Los elementos electrónicos más usados en los SR industriales son los sensores y los actuadores. Estos permiten percibir el entorno (sensores) y actuar (actuadores) en consecuencia, además, dictan su funcionamiento en base a premisas de software, por lo que se realizó un proceso de abstracción específico sobre estos dispositivos electrónicos, para limitar la variabilidad en la programación de estos elementos y abstraer mejor el dominio general. Para ello se analizaron también los códigos fuente de diez tipos de sensores y cinco tipos de actuadores (se realizó un análisis

lexicográfico de 3 códigos por cada tipo de sensor y actuador), lo que constituye el estudio de casi 40 códigos para determinar qué elementos de software comunes se pueden modelar sobre estos elementos electrónicos. Según el análisis se estableció que la mayoría de los sensores tienen cuatro bloques de código principales y comunes en su programación. El primer bloque de sentencias (denominado *header*) está relacionado con la declaración de variables, constantes y librerías. El segundo bloque (nombrado *body*) contiene todas las acciones relacionadas con la inicialización de los módulos internos y externos de Arduino (por ejemplo, puerto serial = *Serial.begin()*), la inicialización de las variables, la determinación de los pines de entrada y salida. El tercer bloque (llamado *footer*) identificado, tiene las sentencias que envían las señales de los sensores para determinar el estado actual del robot, generalmente funciones como *analogRead* o específicas de las librerías utilizadas. Por último, el cuarto bloque (nombrado *logic*) contiene todas aquellas decisiones y acciones que el sensor realiza tras recibir la información del entorno. Generalmente, éstas tienen que ver con la lógica propuesta por el desarrollador del robot en forma de operaciones de los actuadores o de órdenes a los módulos internos para continuar el funcionamiento normal del sistema.

También, se reconoció que existen sensores activos y pasivos utilizados en los SR industriales con Arduino. Estos dispositivos se diferencian en que los sensores pasivos no necesitan alimentación externa para funcionar, como es el caso de los interruptores o los finales de carrera, mientras que los sensores activos usan una fuente eléctrica externa como alimentación, por ejemplo, los sensores de ultrasonido e infrarrojos. La segmentación de los sensores en IRArduino-SPL obedece a que los elementos pasivos poseen menos atributos y operaciones respecto a los sensores activos. En cuanto a los actuadores, estos emplean sentencias diferentes porque son dispositivos de distinta naturaleza, pero tienen las mismas abstracciones de los sensores en el *header*, *body* y *logic* ya que ambos tipos de dispositivos basan sus instrucciones de funcionamiento en las mismas estructuras de programación. La principal diferencia radica en *footer*, porque en los sensores esta abstracción alberga las sentencias relacionadas con el envío de señales para percibir el ambiente, en este escenario los actuadores no realizan esta función, sino que ejecutan acciones que suelen ir precedidas de un cambio en el entorno del SR industrial, por lo que aquí existe una diferencia en la forma de programar actuadores y sensores. Finalmente, es importante indicar que la determinación de los atributos de los sensores y actuadores en el diagrama de clases se basó en lo propuesto por Adafruit

Industries, que estructura el desarrollo de software de sus sensores en [126]. Finalmente, en la Figura 16 se expone el diagrama de clases conceptuales propuesto, como resultado del modelo de dominio abstraído y planteado.

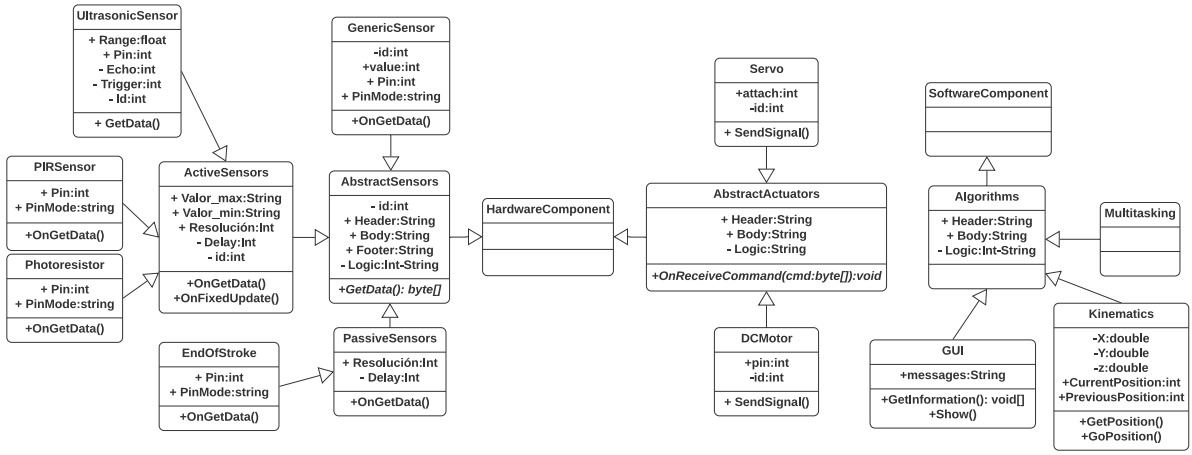


Figura 16. Modelo conceptual propuesto en un diagrama de clases para un SR industrial con Arduino.
Fuente: Elaboración propia.

4.2.2. Identificación de la variabilidad del dominio

Un aspecto diferencial en la SPL es la gestión de la variabilidad. Este enfoque de reutilización implica administrar los puntos de variación entre los diferentes productos derivados de la línea. Con este fin, se identifican los aspectos comunes (*commonalities*) y los variables (*variabilities*) del dominio en cuestión. Cabe destacar que las características variables y las comunes entre los productos tienen la misma importancia en el modelado de la línea de productos [60]. En la investigación se caracterizaron los aspectos comunes y variables según lo propuesto en [127].

Identificación de las características comunes del dominio

Una característica común es un elemento que todos los productos derivados de la SPL deben considerar. En este caso, la plataforma de funcionamiento para todos los derivados de la SPL es Arduino, por lo que los elementos y productos que componen la propuesta tienen una arquitectura de software similar y operan bajo la misma plataforma de desarrollo, lo que facilita el desarrollo de software para los *core assets* y las derivaciones. Otro punto en común es que todos los SR industriales derivados de la SPL utilizan sensores, actuadores, efectores finales y realizan tareas semejantes, lo que permite que el software de estos dispositivos sea parecido porque

tienen lógicas de funcionamiento similar. Asimismo, es importante indicar que los dispositivos que basan su funcionamiento en premisas de software, se abstrajo su programación mediante los modismos de programación, por lo que tienen en común que están estructurados bajo tres bloques principales de código fuente.

Ahora bien, los productos que se generarán de la SPL tienen diferentes características y elementos configurables (sensores, actuadores u otras funcionalidades), pero dentro de estos elementos se encontraron varias características comunes en el software que pueden abstraerse y facilitar la creación de código fuente. En concreto, se puede mencionar que en los sensores existen cuatro secciones de código, que se denominaron *header*, *body*, *footer* y *logic*¸ mientras que en los actuadores se determinaron tres bloques de código *header*, *body* y *logic*. El establecimiento de estas abstracciones para sensores y actuadores facilita la construcción del software en la SPL ya que estructura y simplifica la transformación de la abstracción a texto para cada dispositivo independientemente de su naturaleza.

Identificación de las características variables del dominio

Los elementos que pueden variar en el dominio son numerosos porque los SR industriales son sistemas complejos que están conformados por múltiples partes y configuraciones. Estos se componen de diferentes tipos de actuadores, sensores y efectores finales, lo que habla de múltiples configuraciones posibles para estos elementos. Además, se sabe que existen distintos tipos de dispositivos electrónicos para la plataforma Arduino, debido a que, por su carácter libre, permite acoplar fácilmente nuevas tecnologías y elementos desarrollados. Lo anterior se traduce en un gran número de configuraciones posibles para la construcción de robots industriales, por lo que es necesario limitar la variabilidad en este apartado.

Otro elemento variante en el dominio tiene que ver con los algoritmos que gobiernan el robot, dado que, la utilización o no de cinemáticas directas e inversas son decisiones que afectan al desarrollo y funcionamiento de estos sistemas. Puntualmente, se encontró que en el dominio los desarrolladores de SR industriales con Arduino no hacen uso de este tipo de ecuaciones o algoritmos, que permiten controlar los movimientos del robot, lo que constituye una variación en la construcción del software de los robots industriales, ya que, las cinemáticas tienen un alto nivel de complejidad en la implementación de estos algoritmos. Las

funcionalidades que pueden cumplir este tipo de sistemas son variables, debido a que, los trabajos a los que se pueden adecuar son diversos, haciendo casi imposible abstraer de forma real y convincente las tareas de los robots [128]. Esto también se refleja a nivel de programación de los dispositivos electrónicos (sensores, actuadores y efectores finales), ya que, una vez que cumplen su función, es complicado determinar qué paso seguir con la información recolectada en el caso de los sensores, o cuál es la acción posterior de un efector final una vez cumplida su tarea. Lo anterior no es exclusivo de la estrategia propuesta, debido a que es un problema recurrente en todas las SPL para robots, por lo que han surgido alternativas como OpenRTM [129], Proteus Design Suite [130] y Smartsoft [131] para subsanar esta problemática.

4.2.3. Modelo de características y definición del alcance de la línea de productos de software

La definición del alcance (*scoping*) de una SPL es una actividad fundamental en la construcción de este tipo de estrategia de reutilización porque permite establecer qué productos derivados están dentro y cuáles fuera de la línea de productos [132]. Un alcance demasiado reducido podría dar lugar a productos con un número limitado de características, por lo que serían poco personalizables y no abarcaría la totalidad de los requisitos del usuario. Por el contrario, un alcance demasiado amplio podría implicar un esfuerzo de construcción de productos demasiado elevado, además de mantener los activos principales de la línea [60].

Una de las técnicas más utilizadas para definir el alcance de las líneas de productos de software son los modelos de características, que son particularidades visibles de los productos y representan una abstracción de los requisitos [61]. En particular, los rasgos son características distintivas entre los productos de una línea y, por tanto, determinan y definen sus funcionalidades comunes y variables [133].

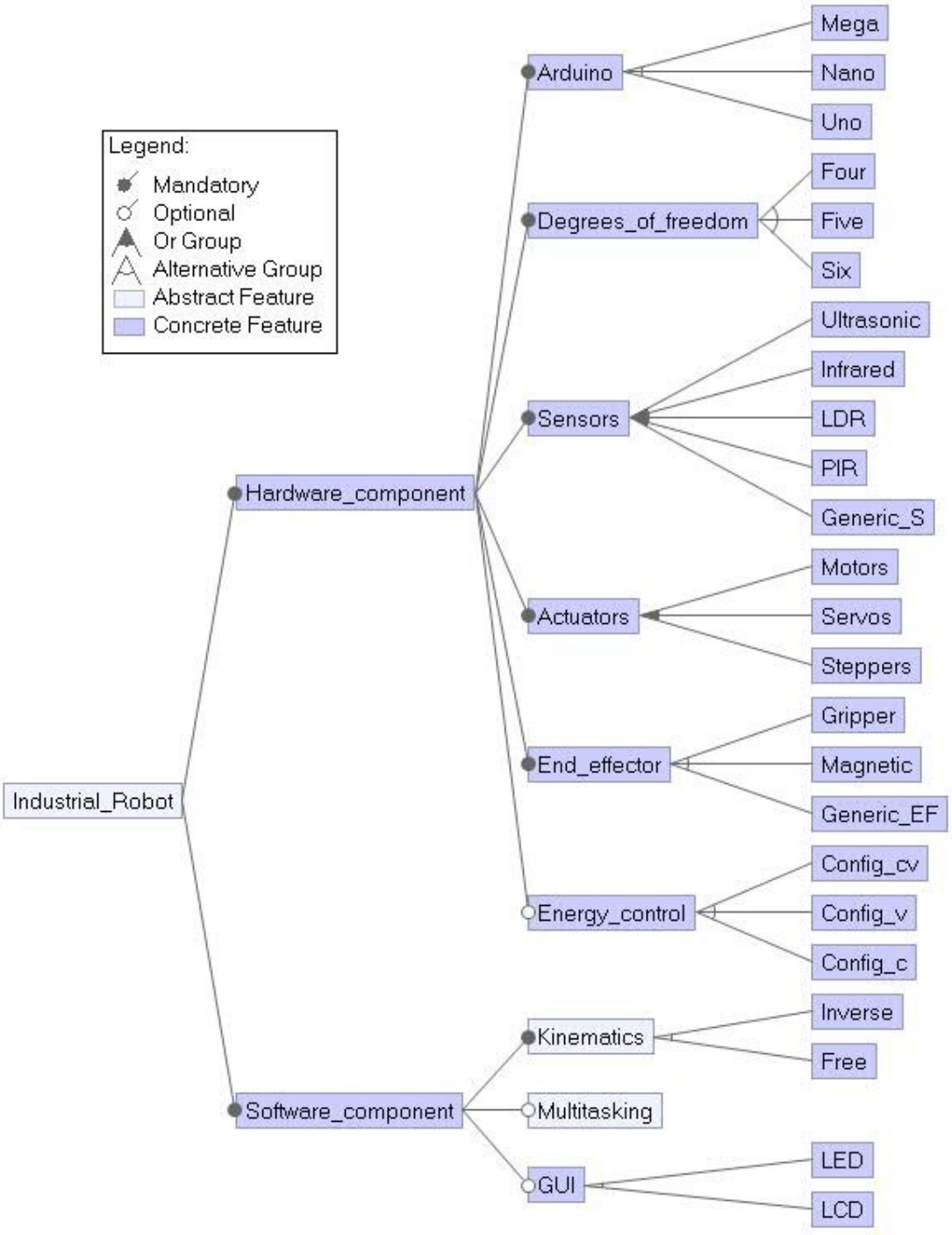


Figura 17. Modelo de características propuesto para la SPL para SR industriales con Arduino.
Fuente: Elaboración propia.

La primera versión del modelo de características (Figura 17) de IRArduino-SPL se definió utilizando FeatureIDE [134], que es un marco extensible para el desarrollo de software orientado a características y permite definir los *features model* de las SPL.

En esta figura, los nodos rectangulares representan las características y las líneas simbolizan las relaciones entre estos. Un único nodo raíz denominado Robot industrial simboliza el concepto del dominio que se está modelando. Las características pueden catalogarse como obligatorias, opcionales o alternativas. Las opcionales se representan con un círculo vacío sobre el rectángulo que las representa y pueden o no formar parte de un producto final, por ejemplo, Control Energía o Multitarea. Las características obligatorias están representadas por círculos negros y forman parte de todos los productos generados de la SPL. Finalmente, las características alternativas son los nodos hoja del modelo que denotan que al menos una debe ser seleccionada como parte de un producto. El modelo contempla cinco tipos de sensores, tres tipos de actuadores y tres tipos de efectores finales.

Análisis sintáctico de la SPL

El alcance de la SPL fue soportado con S.P.L.O.T. [61] y FeatureIDE, que son herramientas que permiten definir, organizar y analizar la variabilidad de una SPL independientemente de su dominio de aplicación. El análisis sintáctico sobre el modelo de características expuso que existen 36 características (35 concretas y 1 abstracta) en el diagrama, de las cuales 8 son obligatorias, 3 son opcionales y 6 son alternativas, mientras que 24 están agrupadas. Asimismo, se determinó que hay 11 características compuestas y 25 características terminales en los nodos hoja del *feature model*. Posteriormente, se definieron 6 restricciones para el modelo propuesto (Tabla 8), entre las que destacan el no poder utilizar las características opcionales como multitarea, GUI, control de energía mientras se seleccione la placa Arduino Nano, debido a que por limitaciones en el procesador puede no soportar el gasto computacional adicional. Igualmente, se localizó que la Relación de Restricciones Transversales del Árbol (CTCR, del inglés Cross-Tree-Constraints Ratio) es del 11%, esta medida permite determinar que la estructura de la SPL es sencilla, ya que la estrategia no presenta restricciones cruzadas entre las *features*, por lo que si se necesitan cambios, los *core assets* no se deben modificar [61].

En la siguiente tabla, se puede apreciar el análisis sintáctico realizado a IRArduino-SPL para la propuesta actual.

Tabla 7. Estadísticas sintácticas del modelo de características para la SPL propuesta.

CONCEPTO	NÚMERO
Características totales	36
Características concretas	35
Características abstractas	1
Características compuestas	11
Características terminales	25
Características agrupadas	24
Características alternativas	6
Características opcionales	3
Restricciones	6
Restricciones cruzadas del árbol (CTC)	4
Relación de restricciones transversales del árbol (CTCR)	0.11

Finalmente, en la siguiente tabla se puede observar las restricciones puntualizadas en la versión primaria de IRArduino-SPL.

Tabla 8. Restricciones cruzadas de la versión de la línea de productos de software para SR industriales.

EXPRESIÓN	SEMÁNTICA
GUI excluye Multitarea	(¬GUI v ¬Multitarea)
GUI excluye Nano	(¬GUI v ¬Nano)
Multitarea excluye Nano	(¬Multitarea v ¬Nano)
Nano excluye Control energía	(¬Nano v ¬Control energía)
Multitarea excluye Control energía	(¬Multitarea v ¬Control energía)
GUI excluye Control energía	(¬GUI v ¬Control energía)

Análisis semántico de la SPL

En esta línea, también se realizó un análisis semántico sobre la SPL que permitió validar que el modelo de características propuesto es consistente y válido para ambas herramientas, además, no presenta características muertas. Igualmente, se identificaron 9 activos principales (*core assets*) que se constituyen como los elementos reutilizables más importantes de la propuesta de reutilización. Asimismo, se estableció que existen 58.590 configuraciones válidas de distintos productos que se podrían derivar del modelo de características, lo que supone un gran número de productos posibles para los SR industriales con Arduino. También, se constató que el grado de variabilidad para la propuesta es de 8.526E-5, este atributo representa que, a mayor variabilidad, más capacidad de personalización en los productos que se pueden derivar. Este resultado fue comparado con otras propuestas en el repositorio S.P.L.O.T [61] y se encontró que el grado de variabilidad es alto, por lo que se podría

analizar detenidamente este aspecto para encontrar un equilibrio entre el costo de desarrollo, el tiempo y la variabilidad esperada [135]. Posteriormente, se encontró que se deben tomar un mínimo de 9 decisiones para tener una derivación funcional a partir de la SPL. Además, se encontraron 2 conjuntos atómicos a partir del análisis del modelo de características, que reflejan adecuadamente los dos componentes principales (hardware y software) expuestos en la Figura 16, lo que permite interpretar que estos conjuntos permiten reducir el problema de análisis en el *feature model* haciéndolo más eficiente. Finalmente, se encontró que la densidad de cláusulas del modelo de características es de 1.5, este atributo se refiere a la relación entre el número de variables y el número de cláusulas (o fórmulas en la semántica del modelo) en el modelo [61].

Tabla 9. Estadísticas semánticas del modelo de características para la SPL propuesta.

CONCEPTO	NÚMERO
Activos principales	9
Características muertas	0
Conjuntos atómicos	2
Configuraciones válidas de productos	58.590
Profundidad del modelo	4 niveles
Densidad de cláusulas	1.5
Grado de variabilidad (%)	8.526E-5

4.3. Ingeniería de aplicación

Se encarga de orientar la construcción y configuración de las derivaciones de la Ingeniería de Dominio, es decir, la generación de productos que satisfacen un conjunto de requisitos y restricciones especificadas por el usuario [136]. En este sentido, el primer paso consiste en exponer cómo se realiza la generación de código por parte de la estrategia de reutilización propuesta.

4.3.1. Descripción del proceso interno de generación de código

Una vez materializado el modelo de dominio, fue necesario construir, catalogar e implementar los componentes de software modelados para la SPL. IRArduino-SPL se basa en las abstracciones (modelo de clases) y en el modelo de dominio concebido, por lo que, como instrumento de creación y reutilización, se definió e implementó un generador de código, que es la herramienta que soporta la infraestructura y la creación de productos software. El generador permite el

desarrollo de activos del núcleo y el desarrollo de productos, que son actividades fundamentales en la construcción de una SPL exitosa. Este generador surge de la combinación del modelo de clases y la Programación Orientada a Objetos (OOP, en inglés). El lenguaje de programación utilizado para construir el generador de código (Apéndice C) fue Python por su versatilidad, fácil adopción y antecedentes exitosos para el desarrollo de este tipo de herramientas [137]. La implementación del generador de código se basó en el diagrama de clases propuesto en la Figura 16, donde se observa que existen dos superclases abstractas denominadas componente hardware y componente software que a su vez tienen subclases, las cuales tienen atributos y operaciones. La subclase sensores tiene tres clases derivadas, que son expresiones concretas del dominio de los sensores con atributos específicos y otros heredados de sus clases padre. Además, están los objetos, que son especificaciones de las abstracciones del dominio, son componentes concretos que representan su programación en el modelo de clases, estos son dispositivos como los sensores de ultrasonido, los servomotores, las pinzas robóticas, o las cinemáticas, entre otros. Esto se debe a la estructura jerárquica expresada en la herramienta desarrollada y permite la generación de código fuente funcional basado en las especificaciones del usuario.

La generación de código para la SPL consta de dos actividades principales, en la primera se establecen las necesidades del usuario según un menú en el que se presentan los posibles elementos con los que se puede crear el software. Entre el repertorio de posibilidades se encuentra la selección de actuadores, sensores, efectores finales, y la posibilidad de seleccionar el número de estos sin un límite predeterminado (Figura 18). La segunda actividad se enmarca en la transformación de los requerimientos del usuario en código utilizable por la plataforma Arduino, a través del paradigma OOP. Las clases son una descripción del modelo de dominio, donde se implementan los atributos (*header*, *footer* y *logic*) que capturan el comportamiento de sus instancias. También, existen algunas clases derivadas (sensores pasivos y activos) que permiten ampliar las posibilidades de los objetos (sensores, actuadores y efectores finales) instanciados con nuevos atributos y métodos, así como heredar otros preexistentes. Los objetos creados pueden manipular los datos de entrada en función de la obtención de datos de salida específicos, por ejemplo, en actividades como declarar variables, donde cada objeto puede manipular el nombre de las variables utilizadas para su funcionamiento evitando conflictos. Esto es importante porque una vez que los objetos son

instanciados, deben ser transformados a un formato de programación que pueda ser legible por Arduino, para esto se utilizan plantillas previamente creadas en base a los modismos de programación encontrados en el modelado de dominio. Las plantillas son expresiones en el lenguaje Arduino de las clases y atributos abstraídos para cada elemento a generar. Los objetos manipulan los datos para determinar operaciones como asignación de pines, determinación de variables, entre otras actividades. Por último, cabe destacar que el generador de código puede asignar pines para evitar conflictos entre sensores, actuadores y efectores finales, para esto se concibió una representación de las placas de desarrollo en forma de diccionarios en Python, donde cada clave del diccionario corresponde a un pin de la placa y su valor equivalente es el estado de este (libre corresponde a 0 y asignado a 1), esto para que la herramienta cada vez que asigne un pin cambia el estado valor de este para que no sea asignado en más oportunidades.

4.3.2. Descripción del proceso externo para generación de código

El generador de código es capaz de crear software para sensores de ultrasonido, sensores infrarrojos, sensores infrarrojos pasivos y fotorresistencias, además de una plantilla o exoesqueleto genérico para sensores (sensor genérico), independientemente del número de pines que ocupe en la placa de desarrollo. A su vez, la herramienta permite generar código para actuadores como servomotores y motores, y también soporta la generación de software en efectores finales como pinzas robóticas o electroimanes. Asimismo, el atributo denominado *logic* tanto en sensores como en actuadores se ha expresado de forma que, si el usuario elige esta opción, el programa genera código que permite comprobar si el dispositivo seleccionado es funcional o no. Finalmente, una vez realizado el proceso de generación de código, la herramienta crea un archivo de texto que contiene el código fuente con todas las especificaciones del usuario.

Figura 18. Interfaz de línea de comandos del generador de código que soporta la SPL para robots industriales.
Fuente: Elaboración propia.

En la Figura 18, se puede observar el menú de línea de comandos que permite al usuario interactuar con la herramienta desarrollada, donde se puede elegir qué elementos se desea crear software en la SPL. En primer lugar, se debe seleccionar la placa Arduino que se va a utilizar, en su momento, la herramienta soportaba el Arduino Uno, Nano, y Mega 2560. Esta elección se realiza porque cada una de las placas tiene diferentes números de pines, por lo que la herramienta debe asignarlos adecuadamente. Posteriormente, se pregunta al usuario si desea que los elementos creados tengan un código que permita determinar si el funcionamiento de estos es correcto (por ejemplo, si se elige esta opción para los motores, el código creado tendrá sentencias para que el motor gire en ambos sentidos cada 3 segundos), esto se enmarca en el atributo *logic* mencionado anteriormente. A continuación, se elige entre los actuadores, sensores y efectores finales soportados, recordando que no existe un número límite para la creación del software para estos dispositivos. Luego, una vez realizadas todas las elecciones, se elige “imprimir” para que el programa genere el código fuente en un archivo de texto, donde podrá ser transferido al Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés Integrated Development Environment) de Arduino para ser compilado en la plataforma. Finalmente, el usuario debe realizar los ajustes necesarios en el código generado por la herramienta, enfocándose en la lógica de funcionamiento a implementar en el robot industrial.

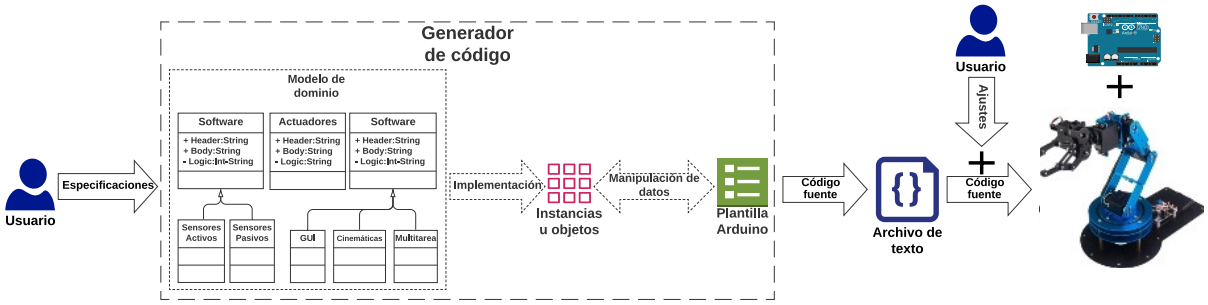


Figura 19. Visión general del proceso de generación de código fuente para la propuesta de reutilización de software.
Fuente: Elaboración propia.

Finalmente, en la Figura 19 se puede apreciar la arquitectura de referencia para la herramienta desarrollada, donde se puede observar el proceso de generación de código fuente interno y externo para la primera experiencia de IRArduino-SPL.

Ahora bien, es importante indicar que la ingeniería de líneas de productos de software es un enfoque de desarrollo que implica un proceso de evolución constante; existe un alto nivel de interdependencia entre los distintos activos de software, la variabilidad y la estrategia de producción [138]. Es por este motivo que en el resto del capítulo se abordará el refinamiento de IRArduino-SPL, donde se tienen en cuenta las experiencias recogidas en la estrategia, especialmente para la Ingeniería de Dominio. Asimismo, se delimita el alcance de IRArduino-SPL ejecutando el método CoMeS-SPL y se redirecciona la Ingeniería de Aplicación hacia un enfoque menos generativo y más orientado al desarrollo de activos núcleo mejor consolidados. Finalmente, es de destacar que en la continuación de este capítulo se hace especial énfasis en la descripción de las actividades principales (desarrollo de activos núcleo, desarrollo de productos y gestión o administración de la SPL) que componen el desarrollo de la SPL, donde intrínsecamente se emplea los elementos encontrados en la Ingeniería de Dominio de la versión anterior y se modifica la Ingeniería de Aplicación hacia un nuevo enfoque basado en los activos núcleo.

4.4. Definición del alcance de la SPL para SR industriales con Arduino

La definición del alcance (*scoping*, en inglés) de una Línea de Productos de Software es un paso crucial en el desarrollo de este tipo de propuesta de reutilización porque

permite determinar sus límites, es decir, precisa qué sistemas están dentro y cuáles fuera de la SPL. Esta actividad no debe tomarse a la ligera, puesto que la correcta planificación y ejecución del alcance puede ser un factor fundamental para determinar el éxito o el fracaso de la implementación de la SPL. Definir el alcance de una línea de productos consiste en establecer y describir qué productos forman parte de la línea de productos y cuáles no. Un alcance demasiado corto puede llevar a desperdiciar oportunidades de reutilización en el dominio, así como, un alcance demasiado amplio llevará a desarrollar activos en los que se puede desperdiciar mucho esfuerzo de desarrollo, generando una rentabilidad insuficiente en la reutilización del dominio [139].

Para definir el alcance de la línea de productos de software, se utilizó CoMeS-SPL, que es un método colaborativo que guía la definición de este tipo de estrategia de reutilización. Este método presenta las tareas y artefactos necesarios para definir correctamente esta actividad. El objetivo principal de CoMeS-SPL es fortalecer la colaboración entre los diferentes roles y mitigar los problemas de interdisciplinariedad con participantes que tienen diferentes intereses, utilizando elementos como thinklets y el Modelo de Proceso de Facilitación [140]. La siguiente figura expone el modelo jerárquico propuesto por CoMeS-SPL, en el que se pueden identificar los objetivos y las tareas necesarias para implementar el método.

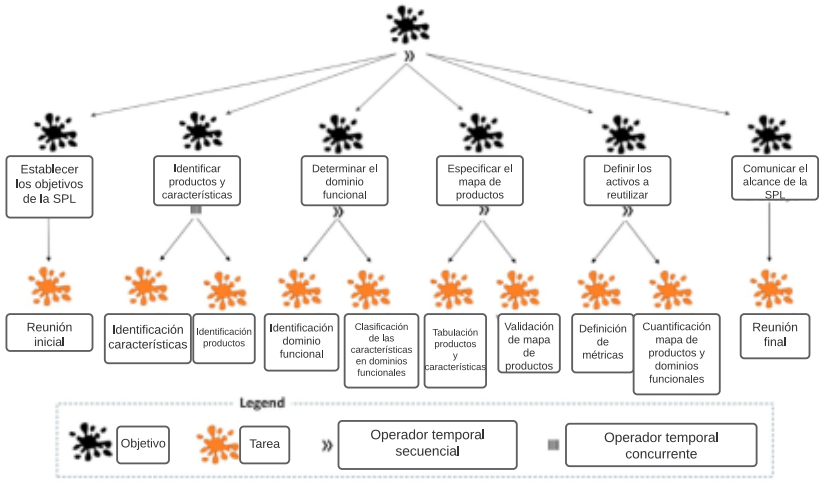


Figura 20. Modelo jerárquico propuesto por el método CoMeS-SPL en el que se pueden observar las principales actividades a seguir.
Fuente: [132].

Un elemento importante en CoMeS-SPL son los roles que componen el equipo, los cuales definen el comportamiento, las habilidades, y las responsabilidades de una o

varias personas en la actividad. Debido a que se utilizó una adaptación de CoMeS-SPL, además, teniendo en cuenta la naturaleza de la implementación de la SPL (Arduino), contexto académico, y por las condiciones de la pandemia COVID-19 [141], redujo la participación de personas en la construcción de la SPL, por lo tanto, algunos de los roles fueron desempeñados por las mismas personas. En la siguiente tabla se muestran los roles utilizados de acuerdo a CoMeS-SPL, además de las responsabilidades y funciones que desempeñan en cada uno de ellos.

Tabla 10. Roles obligatorios en el método CoMeS-SPL para reestructuración de IRArduino-SPL.

FUNCIÓN	PRINCIPALES RESPONSABILIDADES	RESPONSABLE
Expertos en el dominio	• Proponer características a partir de su conocimiento del dominio. • Evaluar qué características pertenecen a los productos objetivo. • Resolver las discrepancias sobre la pertenencia y el tipo de características. • Colaborar en la evaluación de los activos a desarrollar de forma reutilizable.	Saúl Eduardo Ruiz Vladimir Trujillo Arias Ginna Andrea Ramirez
Administrador del negocio	• Comunicar los objetivos de la empresa. • Informar de las preocupaciones de la dirección del proyecto. • Evaluar la viabilidad de la línea teniendo en cuenta los objetivos de la empresa y los recursos previstos para el proyecto de la línea de productos.	Andrés Felipe Solis
Experto en marketing	• Proponer características según el objetivo del mercado y los productos ofrecidos por otras empresas de software. • Identificar las características que hacen que los productos se diferencien de los productos de la competencia. • Ayudar en la identificación de los dominios y productos más relevantes para un determinado segmento de mercado. • Evaluar si las características pertenecen al mercado objetivo y si son comercialmente atractivas.	Andrés Felipe Solis
Arquitecto de software	• Los arquitectos son responsables de determinar qué características constituirán los activos que se desarrollarán reutilizables. • Evaluar la viabilidad técnica de las características propuestas.	Andrés Felipe Solis
Jefe de proyecto de SPL	• Se encarga de la organización de reuniones y talleres durante la actividad de alcance. • Coordinar los esfuerzos, espacios y recursos necesarios para esta actividad.	Andrés Felipe Solis

4.4.1. Ejecución de la adecuación del método CoMeS-SPL con expertos

La ejecución de la adecuación de CoMeS-SPL se realizó para refinar y mejorar la construcción e implementación de IRArduino-SPL en su primera versión, para ello se siguieron las actividades propuestas en la metodología indicada y se invitó a tres expertos en dominios subyacentes al de los robots industriales para evaluar, sugerir y modificar la propuesta de reutilización. Los expertos tienen experiencia en la construcción de robots industriales con Arduino, en la enseñanza de la robótica y en microcontroladores. El perfil de los expertos se puede ver a continuación.

Tabla 11. Caracterización del perfil de los expertos consultados.

NOMBRE EXPERTO	RELACIÓN CON LA ROBÓTICA INDUSTRIAL
Saúl Eduardo Ruiz	• Ingeniero automático y magíster en automática de la Universidad del Cauca, con experiencia en la construcción de robots industriales con Arduino. • Docente de la facultad de ingenierías de la Corporación Universitaria Comfacauca, con 3 años de experiencia en la enseñanza de robótica industrial y robótica móvil.
Vladimir Trujillo Arias	• Ingeniero en electrónica y telecomunicaciones y Magíster en Telecomunicaciones de la Universidad del Cauca, con experiencia en desarrollo de software para microcontroladores. • Experiencia docente de 15 años dictando materias relacionadas con microcontroladores (Arduino y PICs).
Ginna Andrea Ramirez	• Ingeniera mecatrónica y tecnóloga en automatismos mecatrónicos de la Corporación Universitaria Comfacauca, con experiencia en la construcción de SR industriales con Arduino.

La adecuación de CoMeS-SPL se ejecutó utilizando entornos virtuales y medios digitales. Debido a las circunstancias de la pandemia de COVID-19, se realizaron reuniones sincrónicas con un tiempo promedio de ejecución de una hora y media. Los materiales utilizados fueron Google Meet para organizar y llevar a cabo las reuniones, Lucidchart como herramienta colaborativa para modificar, agregar y eliminar *features* al modelo de características propuesto, y Google Docs para contener las plantillas e insumos que los participantes pudieran requerir. Además, se utilizó un modelo de entrevista semiestructurada para obtener algunas respuestas específicas sobre temas relevantes para mejorar la propuesta, con especial énfasis en nuevas abstracciones sobre el dominio. Este tipo de modalidad de entrevista permitió cierta flexibilidad en el desarrollo del diálogo, ya que era del interés del desarrollador del proyecto recoger diferentes experiencias más allá de las que se pedían para contribuir al dominio.

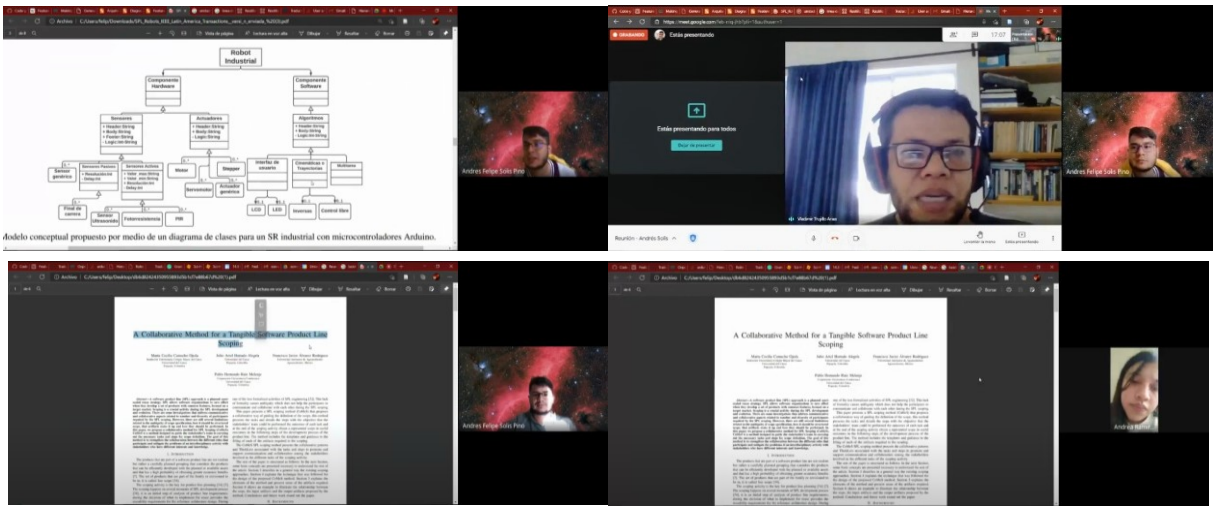


Figura 21. Evidencias del método CoMeS-SPL implementado para definir el *scoping* de la SPL.
Fuente: Elaboración propia.

4.4.2. Consideraciones generales de los expertos con el método CoMeS-SPL

A nivel general, las opiniones de los expertos sobre la IRArduino-SPL fueron positivas, destacando que se trata de una propuesta de investigación novedosa que puede permitir difundir el uso de enfoques de reutilización en el ámbito académico. Asimismo, señalan que la evidencia empírica en el dominio de Arduino en cuanto a la reutilización es casi nula y que es un campo desatendido por la comunidad de investigadores. En cuanto al generador de código que soporta el desarrollo de los activos núcleo y la personalización de la SPL, aluden que es una herramienta que acelera el desarrollo de software para la plataforma Arduino y permite a los estudiantes centrarse en otras problemáticas en la implementación de los SR industriales con Arduino. Conjuntamente, mencionan que la interfaz del generador puede ser propensa a confundir al usuario por lo que se debe trabajar en la usabilidad de la herramienta.

En cuanto a la variabilidad de la SPL, los expertos exteriorizan que son demasiadas configuraciones y esto podría perjudicar el proceso de mantenibilidad de los activos principales, por lo que han expresado varias posibles restricciones a aplicar al modelo de características. Además, sugirieron que la próxima iteración de la línea debería tener un componente de personalización para los dispositivos para los que se crea el código fuente, y de este modo, los estudiantes pueden percibir el concepto de personalización masiva del software en la estrategia de reutilización. Además,

remarcan la importancia de centrarse en un solo tipo de robot industrial porque consideran que el dominio es amplio y sugieren centrarse en brazos robóticos de tres, cuatro y cinco grados de libertad.

El experto en microcontroladores indica que una mejora interesante para la SPL sería que el código generado por la herramienta utilice la OOP porque este paradigma de programación es más cercano a la realidad y a cómo razonan los humanos. En este sentido, cada objeto del programa podría simular los objetos (sustantivos) del dominio (por ejemplo, ultrasonido, servomotor, pinza, entre otros), y los métodos modelan las acciones (verbos, por ejemplo, enviar una señal, girar, entre otros) que pueden realizar los dispositivos. Esto permitirá al estudiante percibir y comprender mejor las ventajas de la propuesta de reutilización de software.

Otro punto importante en el que opinaron los expertos fue en lo referente a los sensores y actuadores, en este sentido, indican que se deben definir correctamente los periféricos, puertos y protocolos de comunicación a utilizar mediante código, para de esta forma llevar un orden en el desarrollo del software en Arduino.

4.4.3. Modelo de características propuesto por los expertos y consideraciones puntuales

Una actividad fundamental comprendida en el modelo jerárquico CoMeS-SPL es identificar, agregar o eliminar *features* poco relevantes o muy importantes en el dominio de estudio. Para ello, se utilizó Lucidchart, que contenía un espacio de trabajo común con un modelo de características (correspondiente a la primera versión de IRArduino-SPL), después, se dio a los expertos plena libertad para modificar el diagrama siempre que hubiera interrelación entre los expertos, para promover la colaboración y los thinklets entre los participantes. También se les pidió que añadieran o sugirieran restricciones al modelo de características propuesto si lo consideraban pertinente.

En la Figura 22, se encuentra consignado el modelo de características concertado entre los diferentes actores de CoMeS-SPL, que sufrió profundos cambios según la opinión de los expertos en comparación con el modelo de características propuesto.

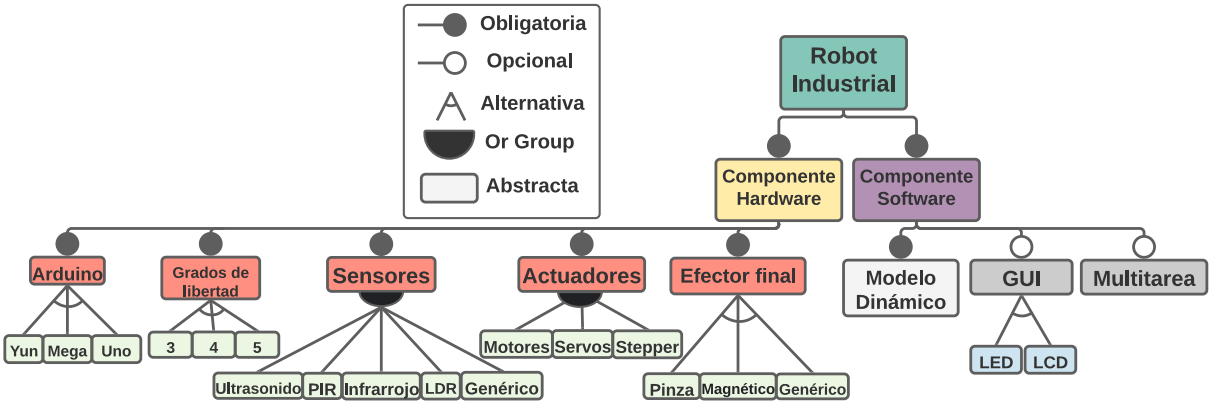


Figura 22. Modelo de características concertado entre los expertos y el desarrollador de IRArduino-SPL.
Fuente: Elaboración propia.

A continuación, para facilitar el análisis de los cambios realizados, se segmenta la justificación de cada modificación en función de las características principales.

***Feature* Arduino**: Se realizó un cambio entre el Arduino Nano y el Arduino Yun, esto se debe a que los pines de entrada y salida pueden ser insuficientes para conectar el número de dispositivos necesarios para un robot. Asimismo, otra razón se debe a la capacidad de procesamiento del Arduino Nano que en algunos casos puede ser limitada, a diferencia del Arduino Yun que utiliza un microprocesador de 32 bits.

***Feature* Grados de Libertad**: Los expertos proponen limitar los grados de libertad del robot industrial a tres, cuatro y cinco. Esto se debe a que cuanto mayor sea el número de grados de libertad, más complicado será el estudio dinámico del robot.

***Feature* sensores y actuadores**: Las características para los sensores y actuadores no se modificaron porque expresaron que son los dispositivos que más se utilizan para construir SR industriales con Arduino, pero mencionan que es importante asegurar cuestiones como los protocolos de comunicación y los periféricos a utilizar, entre otros.

***Feature* efector final:** En cuanto a los efectores finales los expertos aluden que se podría agregar un dispositivo de tipo neumático para lograr movimientos prismáticos, en este punto se tuvo que llegar a un consenso debido a que en Arduino para controlar los efectores y actuadores neumáticos se utilizan relés para controlar su ejecución, por lo que no es posible dictaminar su comportamiento en base a premisas de software.

***Feature* multitarea:** Los expertos indicaron que en la actualidad por las limitaciones de hardware de las placas Arduino es imposible implementar la multitarea real, lo que dificulta la implementación de este tipo de característica. Existen varias opciones para simular este tipo de procesos como la función *millis()*, que permite contabilizar el tiempo transcurrido entre cada ejecución de una tarea determinada [142]. También existen librerías específicas que han sido desarrolladas para cumplir esta tarea, como MultiTasking, AsyncTask y StoryBoard, que basan su funcionamiento en *millis()* y le añaden funciones adicionales. El uso de estas librerías supone un cambio en el paradigma de la construcción de software en Arduino, pues ya no se ejecutan todas las instrucciones dentro del bucle *void loop ()* sino que se utilizan funciones que se ejecutan cada cierto tiempo, esto dificulta aún más el desarrollo de software en Arduino, dificultando el proceso de construcción de los SR industriales. Por esta razón, los expertos han sugerido implementar esta característica en futuros desarrollos en los que puedan centrarse en esta función específica sin complicar el desarrollo del software.

***Feature* Cinemáticas:** En el componente de software del modelo de características se realizaron cambios trascendentales puntualmente en la *feature* cinemática, en este sentido, los expertos indicaron que el estudio del movimiento de los robots es un área con gran complejidad y que incluso los estudios de posgrado están enfocados en lograr una abstracción y estructuración coherente para facilitar la determinación de las ecuaciones de movimiento de los robots [143] [144]. Es por ello que los expertos recomendaron incluir la característica “modelo dinámico” pero que sea de tipo abstracto, esto quiere decir que es un rasgo presente pero que se implementará en futuros desarrollos, para lograr en próximas implementaciones una forma coherente de abstracción de estas funcionalidades. Un ejemplo claro de esta problemática es el estudio propuesto por Abdelhady en [145], donde propone plantear una SPL específica para dar cabida al desarrollo de trayectorias, movimientos y mapeos para este tipo de robots. Lo anterior, describe la dificultad en el desarrollo de estos sistemas que integran elementos de hardware y software en tiempo real.

4.4.4. Recolección de información a personas con conocimientos en robótica

Considerando que en la existieron restricciones para reunirse con personas ajenas al círculo social debido a la pandemia de COVID-19, se decidió realizar y liberar una

encuesta estructurada en línea (Apéndice E) que permitió que personas con diferentes niveles de formación y experiencia en robótica educativa dieran su opinión sobre la propuesta de reutilización y propusieran nuevas características para la SPL. La encuesta se difundió a través de comunidades de robótica en redes sociales, obteniéndose la respuesta de 16 personas de países como Argentina, Colombia, México y Estados Unidos. Este método de recolección de datos se aplicó para encontrar opiniones y aportes relevantes además de la de los expertos sobre IRArduino-SPL; debido a que el dominio de la robótica involucra la unión de múltiples áreas de conocimiento, era esencial obtener la mayor cantidad de retroalimentación posible. Finalmente, las preguntas planteadas a los desarrolladores se exponen en la siguiente tabla.

Tabla 12. Cuestionario planteado a los encuestados para retroalimentación sobre la primera experiencia de IRArduino-SPL.

PREGUNTA	JUSTIFICACIÓN
Nombre del encuestado	Caracterización del encuestado
¿Cómo considera sus conocimientos en la construcción de brazos robóticos con Arduino?	Caracterización del encuestado
En su experiencia y opinión ¿Cuáles son los aspectos de software más relevantes en la construcción de SR industriales con Arduino?	Determinar cuáles son los aspectos más importantes que los encuestados consideran cuando se construye un SR industrial con Arduino, para así evaluar su posible adición a la SPL propuesta.
¿Cuáles son las tres mayores dificultades a nivel de programación que más desmotiva o frustra a los nuevos aprendices en la programación de robots industriales?	Encontrar cuáles son los aspectos de software que más se les dificulta a los desarrolladores, para detectar e intentar resolver este tipo de problemáticas con la propuesta de reutilización.
¿Qué sugerencias tendría para mejorar el modelo de características planteado?	Se pide al encuestado que sugiera nuevas adiciones al diagrama para que sean tenidas en cuenta.
El conocimiento en la construcción de robots industriales con Arduino ¿dónde fue adquirido?	Caracterización del encuestado
¿Estudias alguna carrera relacionada con la construcción de robots?	Caracterización del encuestado

A continuación, se presenta una síntesis de las respuestas dadas por los encuestados y sus respectivos análisis.

P1: ¿Cómo considera sus conocimientos en la construcción de brazos robóticos con Arduino?

La gran mayoría de los encuestados no ha construido más de tres SR industriales con Arduino, esto se debe a que las personas aluden que es una tarea compleja que requiere de conocimientos técnicos en diferentes áreas como programación,

mecánica y electrónica, además, indican que son dispositivos que requieren de un alto presupuesto debido a la cantidad de componentes electrónicos que se utilizan para su funcionamiento.

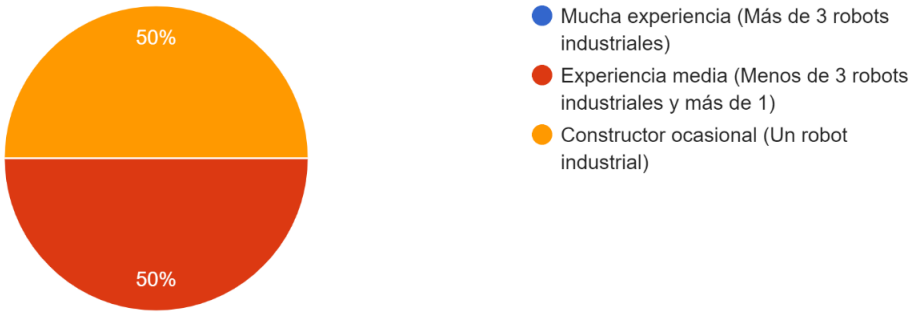


Figura 23. Caracterización de los encuestados según su experiencia en la construcción de SR industriales con Arduino.
Fuente: Elaboración propia.

En esta línea, dos de los encuestados mencionan que la reutilización de software en Arduino se manifiesta con librerías y desarrollos en la web, lo que puede ser un buen punto de partida para estudiar la poca difusión de este tipo de técnica de IS en el dominio.

P2: ¿Cuáles son los aspectos de software más relevantes en la construcción de SR industriales con Arduino?

En esta pregunta, se encontró que existen varios elementos de software que los desarrolladores consideran relevantes en la construcción de los SR industriales con Arduino. Puntualmente, se debe considerar el "estudio dinámico del robot" y la "planificación de trayectorias", que son los aspectos de software más seleccionados reflejando la importancia que se le da al movimiento del robot a través de algoritmos, esto coincide con lo encontrado con los expertos en la sección anterior, quienes mencionaron que los movimientos del robot son un tema relevante, pero que necesitan un estudio aparte y centrado en este tema específico. Ahora bien, el "control de potencia o energía del robot" era una característica que a priori parecía poco demandada, pero al parecer, su elección se debe a que en la industria la monitorización de la energía consumida es fundamental, porque permite proteger a los robots de daños por sobrecorriente.

También, llama la atención que elementos como la multitarea o la frecuencia de muestreo hayan sido tan poco seleccionados, esto puede deberse a que estas

características están intrínsecamente relacionadas con el rendimiento del robot, y es bien sabido que generalmente la intención de construcción de los SR industriales en la academia se realiza para tener una noción de la construcción de estos dispositivos, por lo que el objetivo principal de estos robots con Arduino no está directamente relacionado con tareas que requieran de alta confiabilidad, a diferencia de los brazos robóticos con controladores.

En la siguiente figura se puede observar las principales funcionalidades software que consideran los desarrolladores en la encuesta realizada.

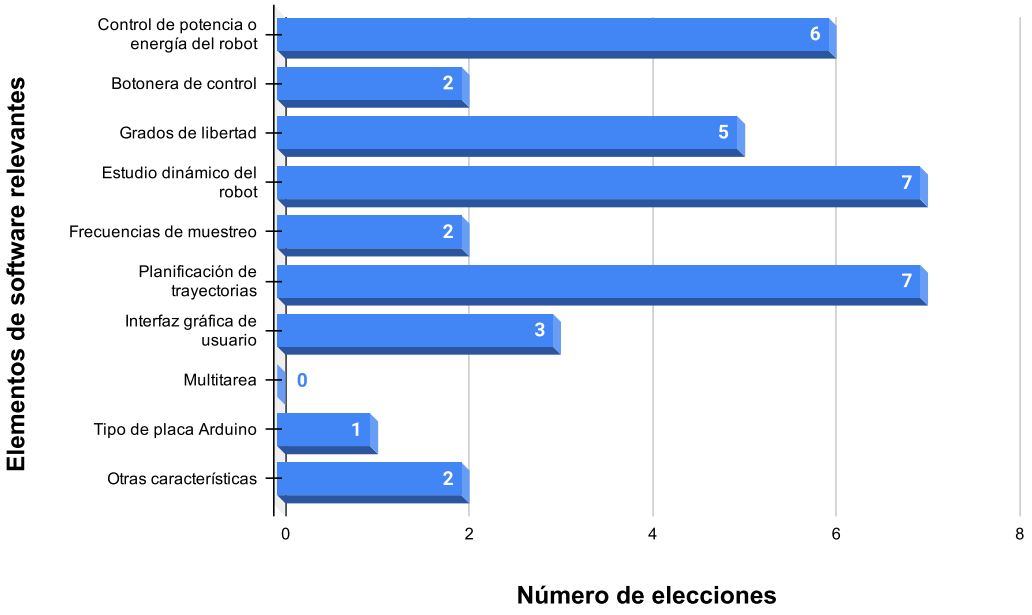


Figura 24. Representación de los componentes de software más importantes para los encuestados.
Fuente: Elaboración propia.

Otra característica que sugieren los desarrolladores es la implementación de un control automático sobre los movimientos del robot, indicando que mediante el modelado dinámico se puede obtener la función de transferencia del dispositivo para luego implementar el control automático. Asimismo, sugieren que el uso de control inteligente, PID o CTC ayudaría al robot a tener una rápida reacción ante posibles obstáculos.

P3: ¿Cuáles son las tres mayores dificultades a nivel de programación que más desmotivan o frustran a los nuevos aprendices en la programación de robots industriales?

Los principales problemas con los que se encuentran los desarrolladores a la hora de construir robots industriales con Arduino tienen que ver con lograr sincronizar los diferentes elementos que componen un robot, esto significa que la multidisciplinariedad en el desarrollo de estos sistemas implica conocer diferentes áreas. En lo que respecta al software, esta problemática se podría solucionar con una estrategia de reutilización del software que permita a los desarrolladores preocuparse de otras cuestiones relacionadas con el hardware y la mecánica del robot.

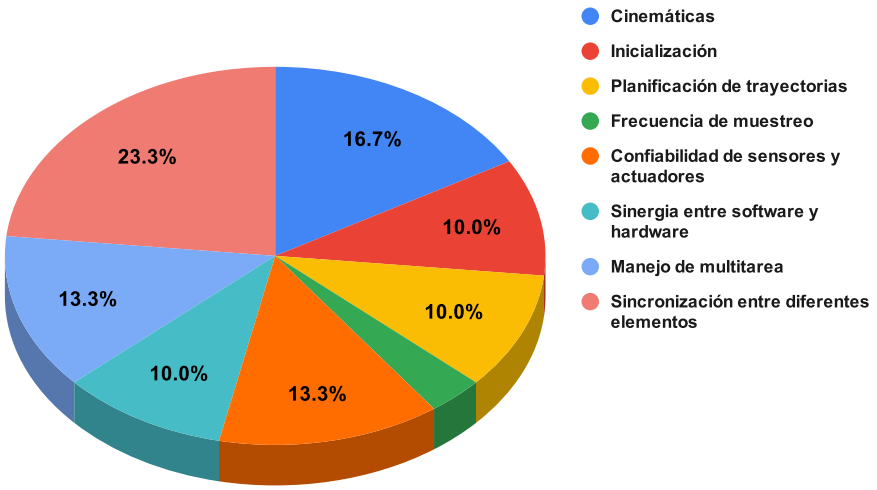


Figura 25. Distribución de las principales dificultades cuando se programan robots industriales con Arduino.
Fuente: Elaboración propia.

Otros aspectos que resultan difíciles para los desarrolladores tienen que ver con encontrar la forma de modelar el movimiento del robot mediante ecuaciones matemáticas, lo que reafirma lo mencionado en las preguntas anteriores. Las cinemáticas se han transformado en uno de los principales problemas a nivel general en la construcción de este tipo de robots. También, otro aspecto como la planificación de las trayectorias han sido una dificultad recurrente, aunque esto debería ser tomado con cierta precaución, ya que, normalmente, esta tarea se ejecuta para evitar que el robot pueda colisionar con personas u objetos circundantes. Esto cambia con Arduino porque su construcción está enfocada al ámbito académico y podría convertirse en un aspecto menos relevante.

P4: ¿Qué sugerencias tendría para mejorar el modelo de características planteado?

Se pidió a los desarrolladores que sugirieran características o prestaciones que puedan ser adoptadas en futuras iteraciones de la SPL, estos no tienen que ser necesariamente adoptados para este proyecto de investigación, sino que pueden ser consignados como trabajo futuro. Una característica solicitada por los desarrolladores es añadir un algoritmo que evite las colisiones entre los objetos que rodean al robot. También demandan agregar elementos software que puedan utilizar multiprocesadores, debido a que en el último tiempo han surgido placas de desarrollo Arduino con mayor capacidad de procesamiento y múltiples núcleos. Una coincidencia entre los desarrolladores y expertos consultados es añadir tres grados de libertad al modelo de características, para permitir una mayor autonomía en los movimientos del robot y no excluir algunas configuraciones de brazos robóticos como los antropomorfos u otros tipos relevantes en el dominio industrial. Finalmente, los desarrolladores indican incluir software para la modelización de datos y los protocolos de comunicación. Además, proponen segmentar el software en múltiples componentes para reflejar las fases de desarrollo, es decir, la capa de software integrada, el software de aplicación y el software de diseño de movimientos.

P5: El conocimiento en la construcción de robots industriales con Arduino ¿dónde fue adquirido?, Si estudias alguna carrera universitaria con respecto a la construcción de robots ¿Cuál es? y ¿Cuál es tu opinión sobre esta?

Para caracterizar a los encuestados y poder encontrar relaciones con sus respuestas, se les preguntó cómo adquirieron sus conocimientos respecto a los SR industriales con Arduino (Figura 26). En este sentido, se encontró que la mitad de los encuestados son autodidactas y sus conocimientos han sido adquiridos mediante medios digitales y redes sociales. También, hubo un segundo grupo de personas que adquirieron sus conocimientos en el ámbito académico, específicamente, en carreras como ingeniería mecatrónica, ingeniería de sistemas e ingeniería automática. Finalmente, hubo un tercer grupo de personas que aportaron sus conocimientos desde la industria hacia los SR industriales con Arduino, aportando su propia experiencia y sugerencias de estos sistemas con controladores a los utilizados en la plataforma de desarrollo.

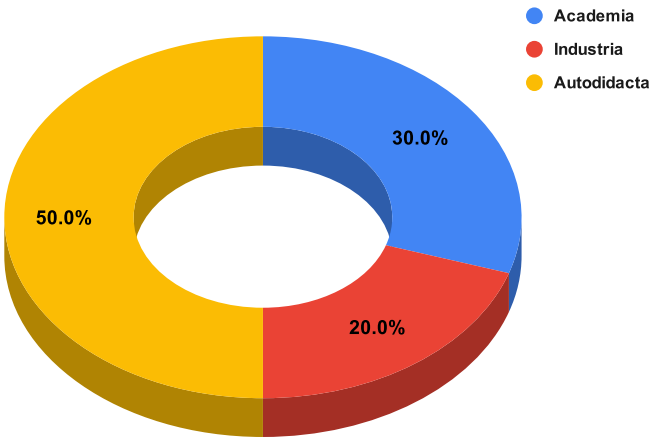


Figura 26. Distribución de los conocimientos adquiridos por los encuestados.
Fuente: Elaboración propia.

Finalmente, con las indicaciones y propuestas encontradas mediante el método Comes-SPL (modelo de características por expertos) y las sugerencias realizadas por los encuestados, se definió el alcance de la línea de productos de software para SR industriales con Arduino. Este es el principal insumo que guía la construcción y desarrollo de la versión 2.0 de IRArduino-SPL. Igualmente, se toman algunas propuestas encontradas en este ejercicio para determinar posibles trabajos futuros, mejoras e ideas para próximas investigaciones sobre este dominio específico.

Finalmente, el framework SPL del SEI menciona que hay tres actividades principales en el desarrollo de líneas de productos de software, la primera es el desarrollo de los activos centrales o núcleo, la segunda es el desarrollo de los productos, y la tercera es la gestión o administración de la propuesta [59]. A continuación, se aborda cada una de las actividades indicadas.

4.5. Desarrollo de activos núcleo para IRArduino-SPL

Teniendo en cuenta la naturaleza inherente y complementaria entre el hardware y el software en el dominio de implementación de IRArduino-SPL, es difícil identificar y analizar cuáles son los componentes de software reutilizables más importantes para la estrategia propuesta, por esta razón, en este apartado se detalla cada uno de los activos reutilizables creados e implementados en base a la Ingeniería de Dominio ejecutada en el capítulo anterior. Para ello, se esboza un diagrama de clases que describe la estructura de los activos núcleo que permite pormenorizar la reutilización efectuada sobre el software en el dominio de la robótica industrial con Arduino.

La reutilización en IRArduino-SPL en términos de software está dada por una librería (Apéndice F) creada para la plataforma Arduino, que permite a los desarrolladores de robots industriales con estos microcontroladores crear software de forma sencilla mediante la OOP, permitiendo emular las entidades reales del dominio que se está modelando (robots industriales con Arduino). En este sentido, la librería proporciona una serie de abstracciones de alto nivel que permiten al constructor emplear elementos y términos comunes en el desarrollo de robots industriales. Ahora bien, aunque efectivamente se está proporcionando una librería para el ecosistema Arduino, también se debe indicar que conceptualmente es un framework (Figura 27) ya que el usuario que emplea la herramienta desarrolla e implementa su lógica gracias a la infraestructura y facilidades que proporciona la librería, proporcionando una serie de abstracciones de alto nivel que permiten al constructor emplear elementos y términos comunes en el desarrollo de robots industriales con Arduino. Esto se refleja en aspectos como la automatización de procesos, el armado y puesta a punto de robots específicos, la implementación de rutinas y actividades predeterminadas, así como la implementación de un esquema o guía conceptual para la creación de software de robots industriales con Arduino. El principio sobre el que se desarrolló la herramienta fue el de abierto-cerrado [146] porque el comportamiento de los módulos o abstracciones puede ampliarse cuando cambian los requisitos de la aplicación (mejorando la mantenibilidad de la SPL), por lo que, al modificar el comportamiento del módulo o abstracción, no se deba hacer modificaciones en el código fuente (programa principal del SR industrial), lo que permite que las aplicaciones sean más fáciles de mantener pues se minimizan los cambios en el código principal de la aplicación y se pueden extender las funcionalidades sin alterar piezas de la aplicación que ya han sido implementadas y probadas.

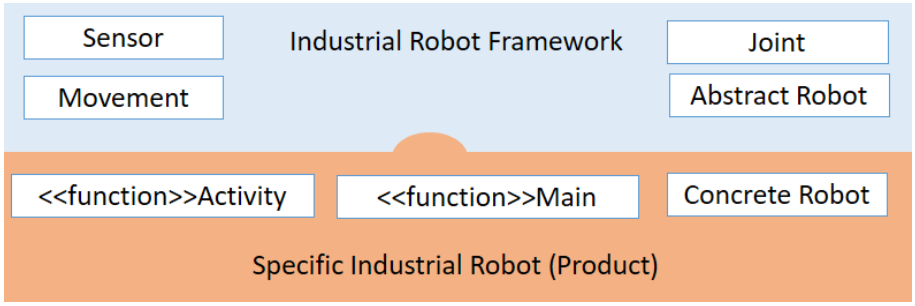


Figura 27. Framework conceptual de IRArduino-SPL.
Fuente: Elaboración propia.

En la Figura 28 se muestra la estructura de la librería desarrollada para IRArduino-SPL, donde se describen las principales clases, los atributos, las operaciones y las relaciones entre las abstracciones realizadas e implementadas en el dominio de los SR industriales con Arduino. En concreto, se puede observar que el primer módulo de clases se denomina "Framework robot industrial" al que pertenecen todos aquellos activos o *core assets* desarrollados para permitir la reutilización del software, aquí el usuario de la librería no necesita interactuar con este módulo ya que es transparente y sólo debe preocuparse de utilizar los métodos proporcionados por esta capa, además de las abstracciones de alto nivel realizadas sobre el dominio que facilitan y contextualizan el desarrollo de software para robots industriales. También, existe un segundo módulo denominado “Aplicación robot industrial” que es el segmento en el que el desarrollador puede emplear las funcionalidades provistas y personalizar el dispositivo en base a sus necesidades.

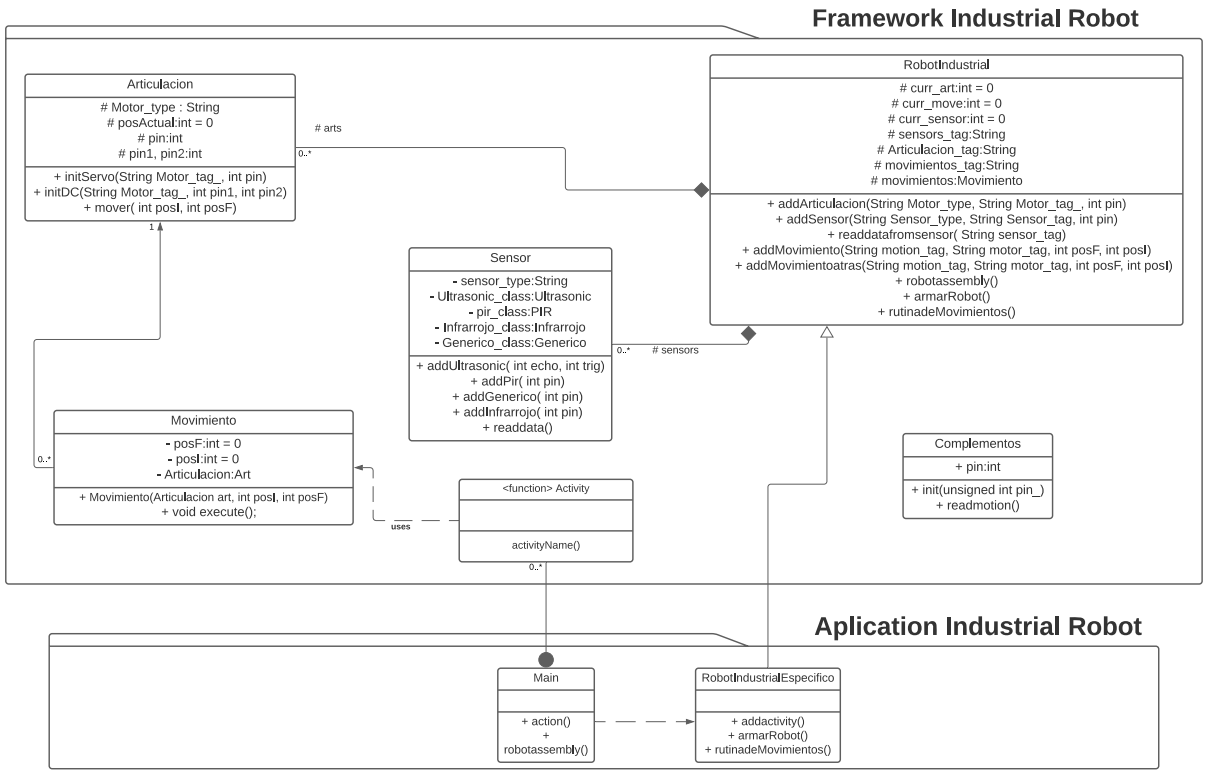


Figura 28. Diagrama de clases de la librería desarrollada seccionada en la capa framework (activos núcleo) y la capa aplicación (usuario).
Fuente: Elaboración propia.

Las abstracciones de alto nivel que tiene la librería de Arduino (IRArduino-SPL) se construyeron en base a lo expresado por los expertos del dominio y las respuestas de los desarrolladores en la encuesta. La abstracción principal de la librería se denominó *RobotIndustrial*, que es una clase de alto nivel conformada por un conjunto de atributos protegidos como *articulación*, *sensores* y *movimientos* que se almacenan en un vector para cada objeto instanciado (robots industriales con Arduino), cada una de estas estructuras de datos guarda las configuraciones de cada robot ofreciendo capacidades de personalización en función de las necesidades del desarrollador. En esta línea, la clase *RobotIndustrial* dispone de una serie de constructores y métodos públicos que permiten adicionar elementos como sensores, actuadores, movimientos, articulaciones y acciones del robot en función de lo que el desarrollador requiera. Asimismo, algunas clases derivadas relevantes de *RobotIndustrial* forman parte de las abstracciones realizadas sobre el dominio; estas son *Complementos*, *Movimiento*, *Articulación*, *Sensor*, *RobotIndustrialespecifico*. Puntualmente, la clase *Movimiento* envía la orden de ejecución de las acciones realizadas por los actuadores del robot; en este caso, los servomotores que permiten el desplazamiento de la estructura necesitan una posición inicial y una posición final, así como la articulación a la que pertenecen. Esta clase proporciona acceso al método *execute()* que remite la orden de movimiento del sistema y permite al usuario de la librería definir los parámetros de movimiento de los actuadores y la articulación a la que pertenecen. Existe otra clase definida dentro de las abstracciones en el dominio que se designó como *Articulación*, que viene determinada por sus equivalentes en el mundo físico y permite trabajar con el hardware de bajo nivel de Arduino, es decir, contiene las instrucciones cotidianas para cada actuador sin abstracciones de alto nivel. Aquí se pueden establecer los pines, los nombres y las posiciones actuales de las articulaciones del dispositivo en cuestión. Igualmente, se tiene la clase *Sensores* que es una abstracción generalizada de los sensores (detectores del ambiente) que pueden utilizarse en los robots industriales con Arduino; ésta emplea una variable global que almacena los datos enviados por el detector y ejecuta una acción dependiendo de las condiciones establecidas por el desarrollador, también permite interactuar con las instrucciones cotidianas de los sensores sin abstracciones de alto nivel (por ejemplo, ultrasonido, PIR, LDR, fotorresistencia entre otros). Asimismo, la clase *Robotico* dentro de las abstracciones realizadas que tiene un rol intermediario entre los objetos y la clase *RobotIndustrial*, permitiendo definir el nombre del robot en cuestión, los métodos de configuración (que permiten el acceso a otras clases), y las

acciones que se ejecutan. Finalmente, el desarrollador se debe encargar de instanciar los objetos, que en este caso son los robots industriales para los que se creará el software con sus respectivas caracterizaciones, así como de accesar a los métodos de la librería que pretenden facilitar el desarrollo de este tipo de sistemas proporcionando un contexto familiar con términos estándar de la robótica. Puntualmente, el desarrollador que utilice la librería deberá especificar el número de articulaciones, el número de movimientos de cada articulación, el nombre de la articulación, la posición inicial y final a ejecutar en el robot, además de poder elegir entre varias rutinas precargadas que el sistema puede ejecutar.

A continuación, se procede a revisar y describir los principales métodos o funcionalidades que aporta la librería desarrollada al usuario (desarrollador de robots industriales con Arduino).

Tabla 13. Descripción de las funcionalidades provistas por IRArduino-SPL.

MÉTODO O FUNCIONALIDAD	CLASE REFERENTE	DESCRIPCIÓN
addArticulacion (String Motor_type, String Motor_tag_, int pin)	Articulacion	Permite agregar articulaciones al robot industrial, especificando el tipo de actuador que le pertenece, el nombre de la articulación y el pin del dispositivo.
addSensor (String Sensor_type, String Sensor_tag, int pin)	Sensor	Añade sensores al robot industrial, detallando el tipo de sensor, el nombre del dispositivo y el pin al que pertenece.
Readdatafromsensor (String sensor_tag)	Sensor	Devuelve el valor de lectura del sensor explícito mediante el nombre del detector
addMovimiento (String motion_tag, String motor_tag, int posF, int posI)	Movimiento	Ejecuta el movimiento del actuador dependiendo del identificador del actuador y la posición inicial y final indicada.
addMovimientoatras(String motion_tag, String motor_tag, int posF, int posI)	Movimiento	Ejecuta el movimiento del actuador dependiendo del identificador y la posición final e inicial indicada.
Robotassembly()	RobotIndustrial	Le indica al robot industrial que se ha terminado de agregar elementos al dispositivo y está listo para ejecutar movimientos.
action()	RobotIndustrial	Ejecuta las configuraciones y movimientos que se desarrollaron por el usuario de la librería,
armarRobot()	RobotIndustrialespcifico	Se especifican todos los dispositivos (sensores, articulaciones y actuadores) requeridos por el usuario.
rutinadeMovimientos()	RobotIndustrialespcifico	Almacena rutinas de movimientos predeterminadas por el usuario o el desarrollador de la librería.
addactivity(String , void (*f)())	RobotIndustrialespcifico	Almacena actividades predeterminadas por el usuario o el desarrollador de la librería.

La librería desarrollada contiene los principales activos núcleo de la SPL planteada, lo que pone de manifiesto la reutilización de software realizada en el dominio y refinada mediante la opinión de expertos. Estos *core assets* permiten la reutilización de fragmentos de código, abstracciones de alto nivel, funcionalidades y conocimientos que fueron encapsulados en forma de librería y encontrados gracias a la Ingeniería de Dominio realizada. Igualmente, estos activos reutilizables se pueden emplear en productos (SR industriales con Arduino) con diferentes capacidades y

características, es decir, a partir de un núcleo común reutilizable se obtienen nuevos productos personalizables.

En la Figura 29 se presentan algunos fragmentos del código fuente de la librería desarrollada, donde se pueden apreciar algunas de las abstracciones de alto nivel realizadas sobre el dominio de los SR industriales con Arduino.

```
class RobotIndustrial
{
  protected:
    std::vector<Articulacion *> arts;
    std::vector<Sensor *> sensors;
    std::vector<Movimiento *> movimientos;

  public:
    RobotIndustrial() = default;

    virtual void addArticulacion(Articulacion *a)
    {
      arts.push_back(a);
    }

    virtual void addSensor(Sensor *s)
    {
      sensors.push_back(s);
    }

    virtual void addMovimiento(Movimiento *m)
    {
      movimientos.push_back(m);
    }

    virtual void configurarRobot()
    {
      this->armarRobot();
      this->definirRecorrido();
    }

    virtual void action()
    {

      for (int i = 0; i < movimientos.size(); i++)
      {
        movimientos[i]->execute();
      }
    }
    virtual void definirRecorrido() = 0;
    virtual void armarRobot() = 0;
};
```

```
class Robotico
{
  public:

    static void run(void)
    {
      RobotIndustrial *robotico = new RobotIndustrialEspecifico();
      robotico->configurarRobot();
      robotico->action();

      RobotIndustrial *robotico2 = new RobotIndustrial2();
      robotico2->configurarRobot();
      robotico2->action();

      delete robotico;
      delete robotico2;
    }
};
```

```
class Movimiento
{
  private:
    Articulacion *art;
    int posI = 0;
    int posF = 0;

  public:
    virtual ~Movimiento()
    {
      delete art;
    }

    Movimiento(Articulacion *art, int posI, int posF)
    {
      this->art = art;
      this->posI = posI;
      this->posF = posF;
    }
    virtual void execute()
    {
      art->mover(posI, posF);
    }
};
```

Figura 29. Fragmentos de código fuente en la librería desarrollada para robots industriales con Arduino como parte de IRArduino-SPL.
Fuente: Elaboración propia.

También, cabe remarcar una limitación que influyó en el desarrollo de la librería para Arduino, y que tiene que ver con la limitada incorporación conceptual de la OOP y la potencia computacional en la plataforma; en este punto, se encontró que los recursos para implementar funcionalidades relacionadas con la OOP como el polimorfismo, la herencia y el encapsulamiento son limitados, lo que se traduce en que las posibilidades y funcionalidades de la librería fueron restringidas, limitando la implementación de abstracciones de alto nivel para obtener un contexto más cercano a la robótica. Lo anterior no es nuevo en el dominio de Arduino debido a que otras investigaciones también se han visto afectadas por la falta de recurso y limitación

[147]–[149]. Por ello, también se han intentado implementar soluciones tradicionales a este problema, como la computación paralela [150] o la computación en niebla [151], aunque sin la aceptación o difusión esperada. Lo anterior plantea una problemática en la plataforma Arduino que puede ser abordada en futuras investigaciones y que representa una limitante importante en la implementación de la ingeniería de software en este tipo de placas de desarrollo.

4.6. Desarrollo de productos en IRArduino-SPL

El desarrollo de productos consiste en fabricar físicamente derivados de la SPL a partir de los activos núcleo, basándose en un plan de producción, para satisfacer los requisitos del usuario. Los insumos fundamentales en el desarrollo de productos son los requisitos, el alcance de la línea de productos, los activos núcleo y el plan de producción para guiar la construcción de software personalizable a partir de activos reutilizables [152]. En concreto, un plan de producción de una SPL es una descripción de cómo se van a utilizar los activos principales para desarrollar un producto específico. Los planes de producción sirven para garantizar que los activos principales se empleen adecuadamente para fabricar productos, así como para estructurar la producción de la organización, asimismo se debe indicar que existen planes de producción sólo para un producto específico o para todos los productos que se pueden fabricar [153]. Sin embargo, es importante señalar que en esta investigación la SPL no está dirigida a una organización o empresa, por el contrario, tiene como público objetivo a los estudiantes, por lo que al diseñar el plan de producción de la estrategia propuesta se omitieron elementos como lo son el cronograma y el presupuesto.

El plan de producción para IRArduino-SPL consiste en especificar los insumos que se deben poseer para obtener un producto final, para ello se emplea la especificación de requisitos de software donde se obtiene una descripción detallada del comportamiento del sistema a desarrollar y los requisitos que debe cumplir. Por esta razón, se ha diseñado una plantilla base (Tabla 14) como elemento guía para estructurar el desarrollo de SR industriales con Arduino a partir de la IRArduino-SPL. La plantilla permite a los desarrolladores concretar elementos como el nombre de los requisitos, la descripción general del dispositivo, el alcance de la solución, el responsable, entre otros elementos. De este modo, se puntualiza cuáles son los

activos núcleos que se deben emplear y cuáles son los elementos variables a añadir a la solución deseada.

Tabla 14. Ejemplo de plantilla general para la descripción de requisitos como insumo para la SPL.

Descripción general del robot	
Nombre del SR industrial	Identificador del robot industrial.
Comportamiento del dispositivo	Descripción detallada de las tareas realizadas por el robot industrial.
Elementos software del robot	Descripción detallada de los elementos que componen el robot y que se pueden producir a partir de la SPL planteada.
Descripción general de los requerimientos	
Nombre de los requerimientos:	Identificador del requerimiento.
Responsable(s):	Desarrollador encargado del requerimiento.
Términos de referencia	
Alcance de la solución	Descripción detallada y clara de lo que incluye y no incluye la solución.
Requerimientos funcionales	Definición de los requerimientos funcionales y una lista de criterios y expectativas que se espera encontrar.
Requerimientos no funcionales y de calidad	Descripción de los requerimientos no funcionales que se deben tener en cuenta para que la solución cumpla con una efectiva ejecución en su entorno.
Viabilidad Técnica	Determinación de la viabilidad del requerimiento.
Requerimientos detallados	
Característica/Funcionalidad	Descripción detallada y suministrada en lenguaje natural por el usuario final, que permite identificar la necesidad puntual.
Razón/Resultado	Producto final esperado.

En la Figura 30 y Figura 31, se pueden observar las diferentes actividades a desarrollar para producir un producto genérico a partir de IRArduino-SPL, en este ejemplo se muestra que se requiere un SR industrial con Arduino. En la primera actividad *Determinar requisitos*, el usuario debe identificar los requerimientos que requiere el robot, los cuales deben detallar características, funcionalidad, y alcance de la solución, esto para estructurar y organizar el proyecto a desarrollar. En la actividad *Especificar Requisitos*, se deben formalizar los requisitos mediante el diligenciamiento de los elementos mostrados en la tabla 15. En la actividad Incluir la librería en el IDE, se debe agregar la librería a Arduino, ya sea mediante los mecanismos establecidos o copiándolo a la raíz del proyecto, con el fin de emplear la librería, la cual da acceso a una serie de abstracciones y funcionalidades de alto nivel (activos núcleo) que emplean términos habituales en el desarrollo de robots industriales, en este sentido, el usuario utiliza los activos núcleo y personaliza el producto deseado mediante la inclusión de sensores, actuadores, recorridos y actividades que puede necesitar, asimismo, es importante indicar que algunos de los elementos que se pueden utilizar han sido desarrollados, pero la librería soporta la

inclusión de nuevos tipos de elementos de hardware (por ejemplo, nuevos tipos de sensores) y de software (por ejemplo, creación de nuevas rutas predeterminadas).

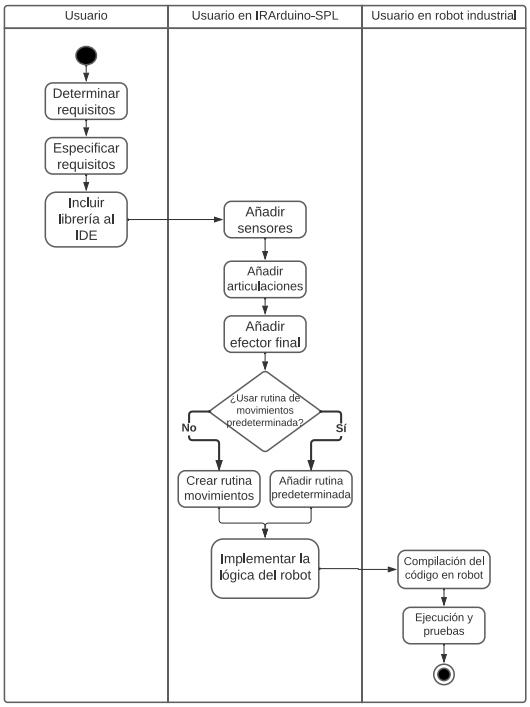


Figura 30. Diagrama de actividades mínimas para generar un producto a partir de IRArduino-SPL.
Fuente: Elaboración propia.

Posteriormente, una vez que el usuario ha personalizado el robot industrial a su gusto, se debe enviar el código fuente, utilizando las funcionalidades preexistentes de la plataforma a través de su IDE, que permite el uso de su compilador y su interfaz de comunicación, para finalmente enviar el código al SR industrial y realizar las pruebas pertinentes.

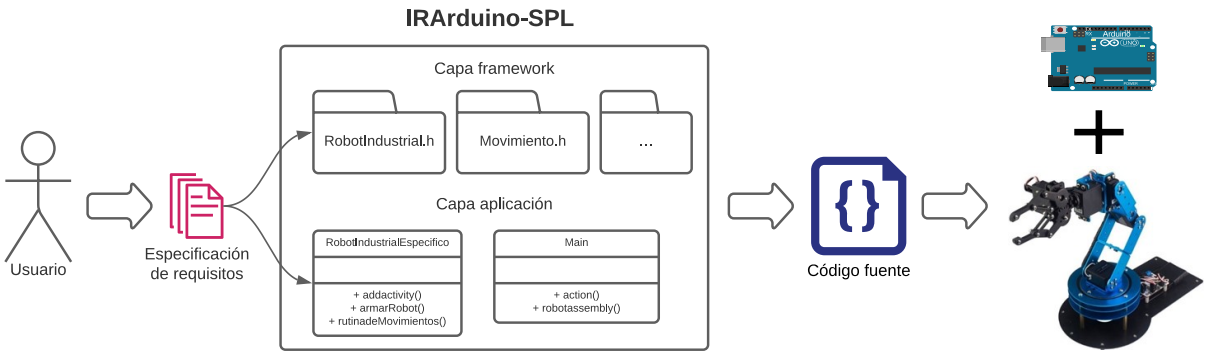


Figura 31. Diagrama genérico de utilización para IRArduino-SPL en la producción de un producto.
Fuente: Elaboración propia.

En la Figura 32 se expone una visión general de las funcionalidades que ofrece IRArduino-SPL como parte del plan de producción de la propuesta; aquí se puede observar el modelo de características de la SPL centrado en las funcionalidades que ofrece la librería, esto con el fin de ayudar a direccionar el desarrollo de los productos de la SPL a nivel de software. Observando el *feature model*, se puede apreciar que de la característica raíz se derivan dos características abstractas, que representan los activos núcleo de cada producto (denominada *Núcleo_Robot*) y sus capacidades de personalización (denominada *Personalización_Robot*). En los activos núcleo, se puede observar que se deben seleccionar las articulaciones, los sensores y los movimientos disponibles en la librería, además de seleccionar opcionalmente si el robot ya está armado y listo para funcionar. Ahora, en cuanto a la capacidad de personalización del producto, *robot_action* es una característica obligatoria que le indica a IRArduino-SPL que ejecute todas las configuraciones realizadas y disponga el robot en funcionamiento. También existe la característica *addActivity* que dota al robot de un conjunto de lógicas secuenciales entre condiciones y movimientos, permitiendo al usuario guardar lógicas predeterminadas o añadir otras nuevas al robot.

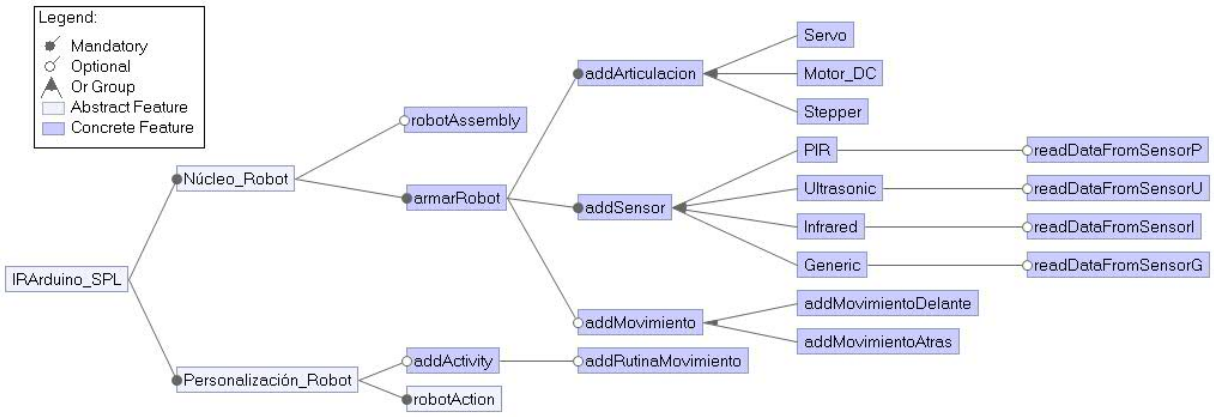


Figura 32. Modelo de características de las principales funcionalidades de la librería desarrollada.
Fuente: Elaboración propia.

Ahora bien, utilizando S.P.L.O.T [61] y [134] se realizó un análisis sintáctico del modelo de características de IRArduino-SPL, se puede indicar que existen 24 características (20 concretas y 4 abstractas) en el modelo, de las cuales 6 son obligatorias, 8 son opcionales y 3 son alternativas, mientras que 9 están agrupadas. Igualmente, se ha determinado que existen 12 características compuestas y 12 características terminales en los nodos hoja. No se definieron restricciones en el

modelo de características porque la multitarea se dejó de lado al definir el alcance de la propuesta, debido a que según los expertos este tipo de utilidad dificulta el desarrollo de los SR industriales en Arduino, por lo que es una función que debe ser cuidadosamente implementada en futuras versiones de la SPL. Al no haber restricciones en el modelo no se presentó ningún CTCR, lo que habla de una mejora en la extensibilidad y modificabilidad de esta nueva versión de IRArduino-SPL especialmente en los activos núcleo. Asimismo, se realizó un análisis semántico del modelo de características que dictaminó que es consistente y válido para ambas herramientas, además, no se presentaron características muertas. Se identificaron 7 activos principales que se constituyen como los elementos reutilizables más importantes de la SPL. Además, se estableció que existen 13.440 configuraciones de productos válidas, lo que indica una disminución de casi el 65% del total de productos respecto a la primera implementación. El grado de variabilidad de la propuesta es de 8.0109E-2, lo que habla de una menor capacidad de personalización del producto, lo que significa un menor coste de desarrollo en cada producto. También, se encontró que se deben tomar un mínimo de 7 decisiones para tener una derivación funcional. A diferencia de la iteración anterior, ésta presentó 1 conjunto atómico, lo que indica que la separación entre hardware y software observada no se produce en esta versión. Finalmente, en la siguiente tabla se puede apreciar un resumen estadístico del modelo de características propuesto.

Tabla 15. Estadísticas sintácticas y semánticas del modelo de características para IRArduino-SPL.

CONCEPTO	NÚMERO
Características totales	24
Características concretas	20
Características abstractas	4
Características compuestas	12
Características terminales	12
Características agrupadas	9
Características alternativas	3
Características opcionales	8
Activos principales	7
Características muertas	0
Conjuntos atómicos	1
Configuraciones válidas de productos	13.340
Grado de variabilidad (%)	8.0109E-2

Es importante indicar que el desarrollo de productos en IRArduino-SPL es la ejecución de la Ingeniería de Aplicación. Ahora bien, si se confronta la Ingeniería de

Aplicación de la primera versión con la actual, se puede mencionar que esta última tiene un enfoque menos generativo y está orientada a emplear, manipular y extender los activos núcleo de la propuesta de reutilización de manera que, cuando estos componentes no sean suficientes, el usuario pueda desarrollar nuevas características y/o implementaciones alternativas (nuevos dispositivos hardware o nuevas trayectorias), lo que amplía las posibilidades de la nueva versión de IRArduino-SPL. En cambio, para la primera implementación, se recurrió a una Ingeniería de Aplicación centrada en la generación de código para facilitar la implementación de la SPL de forma rápida y eficaz, y así determinar si la propuesta de reutilización era viable para el dominio de los SR industriales con Arduino.

4.7. Administración de IRArduino-SPL

La gestión o administración de una línea de productos de software es un proceso fundamental en la correcta implementación de este tipo de estrategias de reutilización, esta actividad se realiza a nivel técnico y organizativo en la entidad donde se implanta. A nivel técnico, se supervisan las actividades de los grupos de desarrollo de activos núcleo y de desarrollo de productos, certificando que los grupos construyen los activos principales y sus derivaciones según lo acordado. Asimismo, a nivel organizativo, la gestión debe asegurarse de que las unidades organizativas reciben los recursos adecuados para la consecución de la SPL [59]. Puntualmente, las actividades más importantes que se desarrollan en la etapa de gestión o administración de una SPL son: el desarrollo de un caso de negocio, la adopción de la línea de productos, la formación de los involucrados en la estrategia, la elaboración de un concepto operativo, el establecimiento de políticas para elegir entre las oportunidades de productos y la previsión tecnológica [154]. Ahora bien, en el caso de IRArduino-SPL el enfoque de gestión se modifica, esto debido a que los usuarios son estudiantes que buscan que se le facilite el desarrollo de software para SR industriales con Arduino, por lo que la estrategia no está dirigida para compañías o empresas de desarrollo con grandes equipos de trabajo con funciones específicas. En consecuencia, el enfoque de gestión de IRArduino-SPL está encaminado a ejecutar alguna de las actividades mencionadas (como enfocar los esfuerzos en la derivación de productos y deja de lado algunas actividades como la capacitación de los involucrados) y a especificar cómo realizar la mantenibilidad y extensibilidad de la propuesta para futuras implementaciones o derivaciones.

4.7.1. Extensibilidad en IRArduino-SPL

En cuanto a la extensibilidad en IRArduino-SPL, cabe destacar que la librería se ha desarrollado con un enfoque de caja blanca, lo que significa que el código fuente puede ser modificado y no hay restricciones para incluir nuevos elementos de hardware, especialmente en lo que refiere a nuevos tipos de sensores y actuadores. Se eligió este mecanismo de extensibilidad porque hay tantos dispositivos de hardware en la plataforma que es demasiado dispendioso incluirlos todos en la versión actual de IRArduino-SPL, por lo que se decidió dar al usuario la posibilidad de ampliar las posibilidades del SPL. La inclusión de nuevos elementos respeta los modismos de programación encontrados en la Ingeniería de Dominio tanto para los sensores como para los actuadores. Ahora, para añadir nuevos sensores en la librería se debe acceder al archivo .h denominado *complements*, que es un espacio reservado para agregar elementos requeridos por el usuario, además de ser la única área de la librería donde se puede interactuar con el hardware de bajo nivel de la plataforma, para esto se debe encapsular cada elemento a añadir (sensor o actuador) en una clase que defina las propiedades (pin, configuración, nombre) y funcionalidades (envió de señal de lectura) de cada sensor, además, de la lógica adyacente si es necesaria.

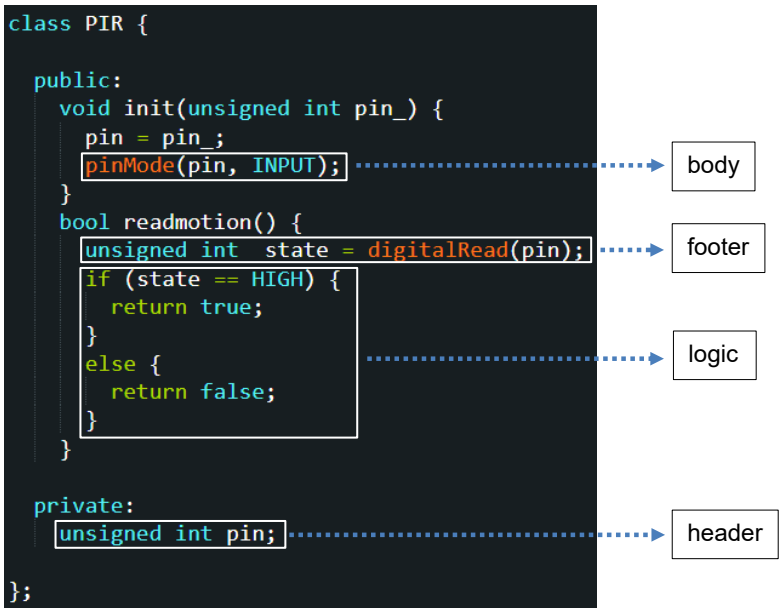


Figura 33. Ejemplo de una clase que abstrae un sensor PIR para IRArduino-SPL donde se marcan los principales modismos de programación encontrados en la Ingeniería de Dominio.
Fuente: Elaboración propia.

En la Figura 33 se expone un ejemplo de implementación para un sensor PIR en IRArduino-SPL, también se remarcan los modismos de programación que se encontraron en la Ingeniería de Dominio, que permiten simplificar y conectar con las abstracciones realizadas en el proceso de abstracción. Lo anterior posibilita utilizar la lógica implementada en la librería, permitiendo utilizar sentencias comunes para todos los sensores agregados y por tanto emplear un lenguaje de programación usual entre todos los dispositivos que funcionan bajo IRArduino-SPL, posibilitando el uso de sentencias como *readdatafromsensor()* independientemente del dispositivo. La principal ventaja de esto es que proporciona un nivel de abstracción apropiado para el dominio de los SR industriales con Arduino y lo aleja de las preocupaciones del hardware de bajo nivel y su programación.

4.7.2. Previsión tecnológica para IRArduino-SPL

Otra actividad que tiene lugar dentro de la gestión de un SPL a nivel corporativo es la denominada previsión tecnológica, que tiene que ver con asegurarse de que los productos derivados y planificados están posicionados para aprovechar las próximas tendencias tecnológicas [154]. En este sentido, es complicado enlazar esta actividad a IRArduino-SPL, porque el público objetivo no es una estructura corporativa. Lo que sí se puede enmarcar es qué tecnologías venideras pueden impactar en la SPL y cómo podrían integrarse.

En primer lugar, la integración de ROS como tecnología establecida puede impactar favorablemente en el dominio de los SR industriales con Arduino, aunque el sistema operativo robótico no es una tendencia tecnológica nueva su implementación en microcontroladores es un área poco explorada, por lo que si esta tecnología pudiera adaptarse con Arduino especialmente en el dominio de los robots industriales agregaría grandes ventajas tecnológicas en la plataforma. Puntualmente, mejoras como la simulación 3D de los robots, la determinación automática de la dinámica del movimiento, la mejora de la visión por computadora, el soporte para la reutilización de grano grande, así como las mejoras en tecnologías como los sistemas de odometría visual, visión estéreo, entre otros, serán grandes beneficios en el dominio. Lo anterior se ve reflejado en [5], donde se expresa que en el dominio general de los SR industriales la integración con ROS ha sido sumamente exitosa, por lo que deberían crearse cursos específicos sobre esta tecnología para que los estudiantes conozcan este marco de trabajo. Otra tendencia tecnológica que podría mejorar

sustancialmente el dominio de aplicación de IRArduino-SPL es el concepto de Fábricas Inteligentes (*Smart Factory* en inglés), que son sistemas de fabricación colaborativa que trabajan en tiempo real para producir bienes personalizados, esta noción engloba áreas de conocimiento como el internet de las cosas, la inteligencia artificial y la analítica de datos masivos [155]. Las *Smart Factories* ampliarían el campo de aplicación de la SPL propuesta debido a que se pasaría de crear SR industriales a la medida a implementar un conjunto de SR colaborativos que trabajan bajo un objetivo común. La principal ventaja de esto sería que el alumno que utilizará IRArduino-SPL no sólo entendería el concepto de desarrollo de software para robots, sino que comprendería un concepto más amplio como el de fábricas inteligentes con elementos de la ingeniería de la colaboración, los macrodatos, y el IoT, además de percibir una tecnología de punta en el dominio. En concreto, el concepto de fábricas inteligentes en el dominio de Arduino, no es ajeno, ya que en los últimos cinco años se han realizado algunas exploraciones en el área [156], pero sin la acogida esperada por lo que quizás nuevas versiones de IRArduino-SPL ayuden a mejorar la adopción más rápidamente. Por último, lo anterior expone las próximas tecnologías y conceptos que pueden añadirse en la SPL en futuras implementaciones y que están en consonancia con las nuevas tendencias tecnológicas en el dominio de los robots.

4.8. Conclusiones derivadas de la construcción de la SPL

En este capítulo se presenta el desarrollo de una propuesta de reutilización de software basada en la Ingeniería de Línea de Producto de Software en el dominio de los SR industriales que hacen uso de microcontroladores Arduino; para la implementación de la propuesta se siguió la Ingeniería de Dominio y la Ingeniería de Aplicación, que guiaron su construcción y permitieron en el contexto de los robots industriales definir elementos fundamentales como el modelo de dominio, el modelo de características, el alcance parcial de la línea y una estrategia de producción basada en la generación de código. Asimismo, se realiza un refinamiento de esta propuesta en base a las actividades propuestas por el framework del SEI [29], que permiten implementar el desarrollo de productos, el desarrollo de activos núcleo y la gestión de la SPL, donde se aplica intrínsecamente la Ingeniería de Dominio encontrada en la experiencia mencionada. La Ingeniería de Aplicación se reorienta hacia un enfoque menos generativo y más volcado en el uso de los activos centrales,

lo que permite la extensibilidad y mantenibilidad del SPL. En concreto, CoMeS-SPL se implementó como metodología central para determinar el alcance del SPL mediante tareas colaborativas (ingeniería colaborativa). Posteriormente, se representaron las principales actividades de implementación de la línea de producto, obteniendo una estrategia funcional con posibilidades de mejora y adaptación a nuevos enfoques de desarrollo.

Finalmente, es importante indicar que un reporte de la SPL desarrollada se resume en un apéndice de este documento (Apéndice H).

4.9. Resumen de los resultados obtenidos en el capítulo

En este capítulo se expone el desarrollo de IRArduino-SPL para robots industriales con Arduino utilizando como base la Ingeniería de Dominio y la Ingeniería de Aplicación; además, se describen y ejecutan las actividades esenciales de este tipo de estrategia de reutilización, como la definición del alcance de la SPL empleando el método CoMeS-SPL, el desarrollo de los activos núcleo, el desarrollo de productos y una aproximación a la gestión de la SPL. Los principales productos obtenidos durante la realización del presente capítulo fueron:

- Capítulo de libro **publicado** en Lecture Notes in Bioengineering (Springer y Scopus) denominado “Software Product Lines for Industrial Robots: A Pilot Case with Arduino.
- Artículo de investigación **publicado** en la revista Electronics (Scopus) denominado “A Software Products Line as Educational Tool to Learn Industrial Robots Programming with Arduino” indexada por Scopus.
- Participación como ponente en el Fourth International Workshop on Gerontechnology (IWoG 2021).
- Abstracción del modelo conceptual genérico de un SR industrial con Arduino a partir de los modismos de programación encontrados en diferentes códigos analizados.
- Definición de un modelo de características y establecimiento del alcance de la propuesta basada en ingeniería de líneas de productos para SR industriales con Arduino.

Resumen de los resultados obtenidos en el capítulo

- Desarrollo e implementación de un generador de código que dé soporte a la infraestructura de la línea de productos de software y la gestión de los activos principales.
- Delimitación del alcance de IRArduino-SPL utilizando tareas colaborativas empleando el método CoMeS-SPL desde la virtualidad.
- Librería desarrollada (Ingeniería de Aplicación) para la plataforma Arduino que contempla las abstracciones (Ingeniería de Dominio) sobre el dominio de los SR industriales, y que funge como activo principal de la propuesta de reutilización.

Capítulo 5

IRArduino-SPL - Validación

Este capítulo presenta la validación de IRArduino-SPL mediante una prueba de concepto y un estudio de caso. Estas dos formas de validación corresponden a una implementación de la SPL desarrollada en un contexto aislado (prueba de concepto) y la otra implementación empírica en un contexto universitario (estudio de caso), ambas cuantificadas a través de métricas e instrumentos. En concreto, para la prueba de concepto se buscó determinar la viabilidad de la propuesta. Para el estudio de caso, se estimó el índice de reutilización, el esfuerzo de desarrollo de software y la usabilidad de la herramienta estableciendo si es o no útil para el desarrollo de software en el dominio de los SR industriales con Arduino.

5.1. Introducción al capítulo

Uno de los elementos fundamentales dentro de la ejecución del método científico es la validación de herramientas y/o desarrollos que contribuyen al campo de investigación en el que se desenvuelven. Por esta razón, se han utilizado dos métodos para validar IRArduino-SPL, primero para observar si la propuesta de una línea de productos de software en el contexto de la robótica industrial con Arduino era viable en el dominio mencionado, se utilizó una prueba de concepto[2], permitiendo determinar la viabilidad de las abstracciones realizadas sobre el dominio y su implementación, esto basado en actividades presentadas en [111]. El segundo método de validación de IRArduino-SPL es el estudio de caso, que en este supuesto buscaba determinar la utilidad de la herramienta desarrollada en términos de reutilización de software, desarrollo de software y usabilidad en un contexto específico como el académico, y así cuantificar la contribución real de la SPL

[2] Una prueba de concepto es la ejecución de un determinado método o idea para demostrar su viabilidad. Una prueba de concepto suele ser pequeña y puede ser completa o no [157]

desarrollada con estudiantes universitarios. La Figura 34 muestra el esquema de la validación para IRArduino-SPL.

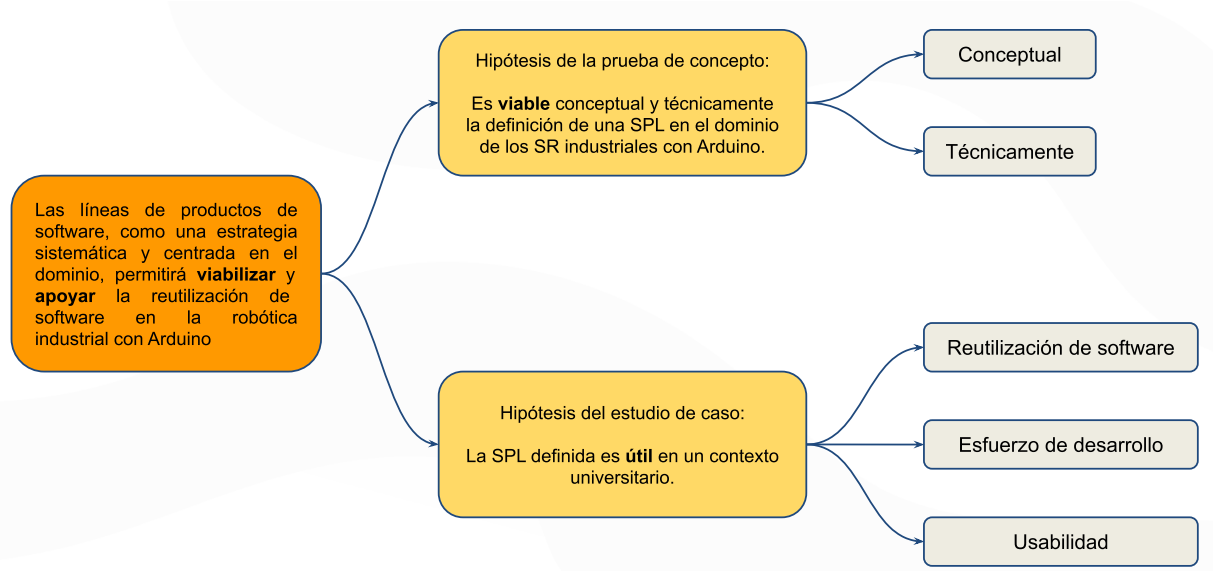


Figura 34. Árbol de hipótesis referente a la validación para IRArduino-SPL.
Fuente: Elaboración propia.

Puntualmente, para evaluar IRArduino-SPL en un contexto académico, se utilizó la herramienta de investigación denominada estudio de caso, para evaluar la utilidad del desarrollo en este contexto puntual. Este tipo de instrumentos en ingeniería de software tiene como objetivo estudiar o indagar sobre los fenómenos relacionados con el desarrollo de software a través de las partes interesadas en un contexto real [31]. Para efectuar el estudio de caso que permitiera la evaluación de IRArduino-SPL se desarrollaron las siguientes actividades generales de acuerdo a [31]: Diseño, Preparación, Ejecución, Análisis y Reporte, ver Figura 35. A continuación, se describe de forma detallada la ejecución y materialización de las actividades mencionadas.

Figura 35. Actividades generales empleadas para el estudio de caso en IRArduino-SPL.
Fuente: [31].

5.2. Prueba de concepto IRArduino-SPL

En esta sección se efectúa una implementación de la SPL propuesta para corroborar la viabilidad en el dominio abstraído y desarrollado en los SR industriales con Arduino. Para ello, se realizó una prueba de concepto donde se implementó un robot industrial con cinco grados de libertad, dos tipos de sensores y una pinza robótica (Figura 36), el cual fue derivado como producto a través de la estrategia de reutilización definida.

Tabla 16. Resumen de las actividades ejecutadas en la prueba de concepto.

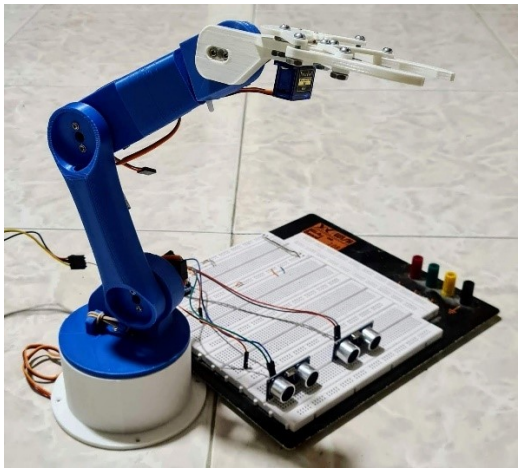
Actividades	Instrumentos de apoyo
Actividad 1. Construcción física del SR industrial**.**	Presentación del proyecto y los elementos conceptuales de la propuesta.
Actividad 2. Determinación de los requisitos del robot (sensores, actuadores, actividades, entre otras)	Plantilla para establecimiento de los requisitos del robot.
Actividad 3. Transformación de las abstracciones a código mediante plantillas.	IRArduino-SPL primera versión
Actividad 4. Adecuaciones al código para ajustar la lógica de funcionamiento del sistema.	Documento de requisitos, documento con guía de IRArduino-SPL, escenarios y plantillas de resultados para su ejecución.
Actividad 5. Compilación del software en la placa de desarrollo.	Plataforma Arduino e IRArduino-SPL primera versión
Actividad 6. Evaluación de la aplicación e introducción a las conclusiones de la prueba de concepto.	Documento de proyectos futuro

La prueba se realizó en un entorno experimental (Apéndice D) con un brazo robótico con Arduino, una placa de desarrollo Mega 2560, la plataforma de desarrollo Arduino, elementos electrónicos de potencia y la implementación de IRArduino-SPL, permitiendo determinar la factibilidad de la propuesta.

Las actividades que se siguieron en la prueba de concepto (Tabla 16) fueron: la construcción física del brazo robótico, la generación de código por la SPL (está actividad se compone por la determinación de los requisitos del robot, la transformación de las abstracciones a código a través de plantillas, las adecuaciones al código para ajustar la lógica de funcionamiento del dispositivo y la compilación en la placa de desarrollo), la determinación del porcentaje de reutilización en la propuesta se hizo en base a la ecuación 1 y por último, el establecimiento de las conclusiones de la prueba de concepto.

$$R = \frac{LR}{TL} * 100 \quad (1)$$

En la ecuación 1, LR es el número de Líneas de Código Fuente (SLOC, por sus siglas en inglés Source Lines of Code) reutilizadas, TL es el número de total de líneas de código del sistema y R la reutilización del software de la propuesta en términos de porcentaje del total de líneas de código. La ecuación anterior es una adaptación de la métrica clásica empleada para determinar el tamaño de una característica o de un software general en una SPL mediante sus líneas de código (denominada *Lines of Code* (LOC)) [158].



```
#include <NewPing.h>
#include <Servo.h>
Servo myservo;
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;
Servo myservo6;
int pos = 0;
int pos2 = 40;
int pinEcho = 4;
int pinTrig = 5;
int contador = 0;
int max_distance = 200;
NewPing sonar(pinTrig, pinEcho, max_distance);
int ban1 = 0;
int pinEcho2 = 2;
int pinTrig2 = 3;
NewPing sonarb(pinTrig2, pinEcho2, max_distance);
void setup() {
  Serial.begin(9600);
  pinMode(3, INPUT);
  myservo.attach(9);
  myservo2.attach(8);
  myservo3.attach(7);
  myservo5.attach(10);
  myservo6.attach(12);
  pinMode(LED_BUILTIN, OUTPUT);
```

Figura 36. (a) Robot industrial derivado como producto de la primera iteración de la SPL propuesta. **(b)** Fragmento de código fuente generado con los ajustes respectivos del desarrollador del robot. **Fuente**: Elaboración propia.

Utilizando la ecuación 1 propuesta en el documento de Frakes y Terry en [159], se determinó que la primera iteración de la línea de productos de software proporciona al desarrollador el 33.3% de los activos de software reutilizables (en términos de líneas de código) utilizando el modelado de dominio propuesto. El 66.7% restante del código, fueron ajustes del desarrollador en cuanto a la lógica de funcionamiento del robot, es decir, adaptar el código obtenido para que el robot industrial realice una acción coherente. En este caso, para simular el transporte de un objeto de un lugar a otro, en función de lo percibido por un sensor de ultrasonido y un final de carrera (sensor genérico). Asimismo, la Tabla 17 muestra algunos datos recogidos en la ejecución de la prueba, donde se puede destacar que las líneas de código fuente reutilizadas en *header*, *body* y *footer* suponen el 86% del total de SLOC reutilizadas en estos atributos, siendo el 14% restante programado por el desarrollador del robot. En cuanto al atributo *logic* las SLOC reutilizadas representan el 17% del total de líneas en esta sección, mientras que el programador desarrolla el 83% restante. Esto es un ejemplo de que se deberían realizar mejores abstracciones en este atributo específico (*logic*), para aumentar las posibilidades de reutilización en las funciones o tareas realizadas por los robots y, por tanto, las posibilidades de la propuesta de reutilización planteada en la prueba de concepto.

Tabla 17. Datos recogidos del código fuente del robot industrial con Arduino en la prueba de concepto

CONCEPTO ESTUDIADO	ESTADÍSTICA
SLOC contribuidas por la propuesta	68 (33.3%)
SLOC contribuidas por el desarrollador	136 (66.7%)
SLOC totales en la prueba de concepto	204
SLOC reutilizadas en *header*	12
SLOC reutilizadas en *body*	10
SLOC reutilizadas en *footer*	21
SLOC reutilizadas en *logic*	25
SLOC contribuidas por el desarrollador en *header*	1
SLOC contribuidas por el desarrollador en *body*	6
SLOC contribuidas por el desarrollador en *footer*	0
SLOC contribuidas por el desarrollador en *logic*	129

En el desarrollo de la prueba de concepto se estableció que los diferentes elementos que forman parte de la SPL son compatibles y trabajan de forma coherente, generando un producto funcional y reutilizando los activos definidos en la estrategia propuesta. De esta forma, se pudo validar que conceptualmente la propuesta de reutilización en el contexto de los SR industriales con Arduino es viable, debido a que según la ejecución de la prueba de concepto se obtuvo un robot industrial con

Arduino que cumple con las tareas para las que fue construido (transportar un objeto de un lugar a otro), con algunas limitaciones. Entre los principales activos de software aportados por la estrategia de reutilización se encuentran, generación de código fuente para los diferentes tipos de sensores, actuadores y efectores finales requeridos por el usuario, así como una sinergia correcta entre el hardware y el software del robot, permitiendo al desarrollador preocuparse por actividades específicas como la lógica de las tareas del dispositivo o la determinación de las cinemáticas. Con la reutilización materializada en la prueba de concepto, se encontró que el código aportado por la herramienta ahorra tiempo de desarrollo al programador, evitándose labores como declarar librerías y variables, asignar pines, inicializar variables y puertos, o incluso brindar un punto de partida para organizar el código del robot de forma estructurada. Esto se traduce en que la propuesta puede ayudar a economizar tiempo de desarrollo y brindar un contexto favorable como punto de partida para iniciar la programación de elementos específicos, tareas concretas, definición y utilización de cinemáticas, o incluso desarrollo de interfaces de usuario.

Dado que esta es la primera implementación de una propuesta de reutilización en un dominio poco explorado, se evidenciaron falencias que se subsanaron en la posterior implementación de IRArduino-SPL. En concreto, se encontró que existen limitantes relacionadas con la generación de código fuente para la programación de las funciones del robot (en la prueba de concepto, sujetar y transportar un objeto), por lo que el desarrollador debe encargarse de esta tarea. Esto sucede por la gran cantidad de labores que desempeñan estos dispositivos electromecánicos. En la segunda versión de IRArduino-SPL, se abstrajo algunos comportamientos comunes de los SR industriales, de forma que se integraron en la estrategia propuesta las principales funcionalidades que ofrecen estos sistemas.

Finalmente, por el estado inicial de esta iteración de la propuesta de reutilización, se encontraron otras limitantes que fueron corregidas. La primera relacionada con el generador de código y su escalabilidad debido a que éste fue desarrollado utilizando la OOP, que dificulta la creación y desarrollo de activos software para elementos específicos (actuadores o sensores) según las abstracciones realizadas, ya que, este paradigma no está enfocado a este objetivo, a diferencia del MDE o CSBE, que se centran en permitir la generación automática de código, proporcionando capacidades de personalización e interoperabilidad. En este sentido, en la siguiente iteración de

IRArduino-SPL se cambió el paradigma de generación de código por uno centrado en los activos del núcleo para permitir un mayor nivel de reutilización del software y facilidad en la creación de activos reutilizables.

5.2.1. Análisis prueba de concepto

Se desarrolló una prueba de concepto en la que se produjo satisfactoriamente un robot industrial con Arduino con cinco grados de libertad, dos tipos de sensores y una pinza robótica, con lo cual se evidenció que la SPL es coherente e implementable, por lo que se pudo evaluar que conceptualmente la propuesta de reutilización en el contexto de los SR industriales con Arduino es viable. Se detectó que la primera propuesta de la SPL enfrenta desafíos en cuanto a cómo lograr la abstracción de las funciones comunes de los robots o la inclusión de nuevos elementos de software. Estos retos pueden haber surgido de la abstracción realizada en el dominio, que quizás se centró más en el hardware y no tuvo en cuenta los elementos de software de base, por lo que en la próxima iteración se redirigió hacia un enfoque centrado en activos núcleo. Particularmente, la primera abstracción del dominio ayuda a materializar la reutilización, pero no incorpora elementos importantes que deben tenerse en cuenta para futuras versiones de la SPL. Adicionalmente, se observó que es necesario mejorar la gestión de la variabilidad en la línea de productos de software, ya que existen demasiadas configuraciones posibles, lo que podría dificultar el desarrollo, la gestión y el soporte de los principales activos en el futuro.

Ahora bien, en las siguientes secciones se procede a exponer el diseño y ejecución del estudio de caso sobre IRArduino-SPL en un contexto académico.

5.3. IRArduino-SPL: Diseño del Estudio de Caso

5.3.1. Contexto

El estudio de caso se realizó en un contexto universitario con la participación de estudiantes de la Corporación Universitaria Comfacauca-Unicomfacauca. En concreto, para el desarrollo del estudio de caso, seis estudiantes (divididos en tres grupos de dos al azar) participaron en la evaluación de IRArduino-SPL. Los estudiantes se encontraban entre el quinto y sexto semestre con conocimientos en microcontroladores, OOP, y el desarrollo de sistemas robóticos en hardware y

software, por lo que contaban con los fundamentos necesarios en las áreas de conocimiento utilizadas en la construcción de sistemas robóticos industriales con Arduino.

5.3.2. Selección del estudio de caso

De acuerdo con López González en [160], los estudios de caso presentan diversas variaciones en función de las siguientes variables: la cantidad de casos (simples o múltiples), las unidades de análisis (holístico o detallado), el objetivo de la investigación (descriptiva, demostrativa, generativa) y la temporalidad (diacrónica o sincrónica). Según Runeson *et al.* [31], el presente estudio de caso se define como holístico, descriptivo, simple y con una temporalidad sincrónica, considerando como unidades de análisis a dos grupos de alumnos, y como fuentes de información sus integrantes. Finalmente, las características del estudio de caso se pueden observar en la siguiente tabla.

Tabla 18. Resumen general del estudio de caso implementado son IRArduino-SPL.

Variables	Tipo	Cantidad
Cantidad de casos	Simple	1
Unidades de análisis	Holístico	3
Objetivo de la investigación	Descriptiva	
Temporalidad	Sincrónico	
Fuente de información		Seis estudiantes

5.3.3. Objetivo y Pregunta de investigación

El objetivo del estudio de caso fue indagar sobre la utilidad de ***IRArduino-SPL*** para acelerar el desarrollo de aplicaciones informáticas en SR industriales en un contexto académico. De acuerdo al objetivo se planteó la siguiente pregunta de investigación: ***¿Es IRArduino-SPL un acercamiento útil para la reutilización de software en el dominio de los robots industriales con Arduino en un contexto académico*?** Para responder a esta pregunta, se estableció que una forma es a partir de una experiencia en el ámbito académico.

5.3.4. Hipótesis del estudio de caso

Este estudio plantea la hipótesis según la cual la implementación de una línea de productos de software en el dominio de los sistemas robóticos industriales con Arduino es útil en el proceso de desarrollo de software de este tipo de sistemas

robóticos, considerando aspectos como el índice de reutilización, el esfuerzo de desarrollo y la usabilidad percibida.

Tabla 19. hipótesis planteadas en relación con la métrica correspondientes en el estudio de caso.

	Utilidad	
H1. La definición de una SPL es útil en el dominio de los sistemas robóticos industriales con Arduino SPL.	**H.1$_0$.** El índice de reutilización alcanzado por el grupo que utilizó IRArduino-SPL es igual o inferior al 31% en el SR industrial desarrollado	**Reutilización**
	H.2$_0$. El esfuerzo de desarrollo del grupo que utilizó IRArduino-SPL es igual o superior a 5 horas-persona para desarrollar el SR industrial.	**Esfuerzo de desarrollo**
	H.3$_0$. La percepción de usabilidad media sobre IRArduino-SPL es mayor o igual a 2.82 según las puntuaciones globales medias de Sauro y Lewis	**Usabilidad**

Índice de reutilización

- ***H.1$_0$*:** El índice de reutilización alcanzado por el grupo que utilizó IRArduino-SPL es igual o inferior al 31% en el SR industrial desarrollado.
- ***H.1$_A$*:** El índice de reutilización alcanzado por el grupo que utilizó IRArduino-SPL es superior al 31% en el SR industrial desarrollado.

Esfuerzo de desarrollo

- ***H.2$_0$*:** El esfuerzo de desarrollo del grupo que utilizó IRArduino-SPL es igual o superior a 5 horas-persona para desarrollar el SR industrial.
- ***H.2$_A$*:** El esfuerzo de desarrollo del grupo que utilizó IRArduino-SPL es inferior a 5 horas-persona para desarrollar el SR industrial.

Usabilidad

- ***H.3$_0$*:** La percepción de usabilidad media sobre IRArduino-SPL es mayor o igual a 2.82 según las puntuaciones globales medias de Sauro y Lewis.

- ***H.3A*:** La percepción de usabilidad media sobre IRArduino-SPL es inferior a 2.82 según las puntuaciones globales medias de Sauro y Lewis.

5.3.5. Indicadores y métricas

En este apartado se define de forma objetiva cómo se evaluará el estudio de caso para responder a las preguntas de investigación planteadas, a partir de un conjunto de métricas e indicadores definidos. La siguiente tabla muestra los indicadores e instrumentos establecidos para el estudio de caso.

Tabla 20. Indicadores, métricas e instrumentos empleados en el estudio de caso para IRArduino-SPL.

Indicador	Concepto	Objeto de estudio	Instrumentos
Índice de reutilización	Se define como la proporción de código reutilizado en nuevos sistemas. El aumento de la productividad puede definirse en términos de número de líneas de código fuente producidas por los programadores/unidad de tiempo o del número de componentes reutilizables [161].	Evaluación del SR industrial obtenido en base a los activos núcleo empleados.	Encuesta y datos se recolectaron basándose en el protocolo de observación.
Esfuerzo de desarrollo de software	Se define como la cantidad de trabajo necesario para desarrollar o mantener un software, normalmente expresado en términos de horas-persona o dinero [162].	Evaluación del SR industrial obtenido en base al tiempo empleado	Encuesta y datos recolectados basándose en el protocolo de observación.
Percepción de Usabilidad	Capacidad del producto de software para ser comprendido, aprendido, utilizado y atractivo para el usuario, cuando se utiliza en determinadas condiciones.	Percepción de los estudiantes basada en la facilidad de uso de la herramienta.	Encuesta

A continuación, se describen los indicadores y la forma en que estos son calculados a través de las métricas identificadas.

Índice de reutilización

Es un indicador para determinar el uso de los activos existentes dentro de un proceso de desarrollo de productos de software [59], concretamente en este proyecto el índice de reutilización se utiliza para determinar la contribución de IRArduino-SPL en el desarrollo de un SR industrial en un contexto académico. Para el índice de reutilización en este proyecto, tres métricas relacionadas determinarán si IRArduino-SPL ayuda a mejorar el proceso de desarrollo en el dominio. La determinación del índice de reutilización se realiza computando el número de líneas de código del proyecto desarrollado, así como el número de características o *features* empleadas

que aportó IRArduino-SPL. Asimismo, se evaluó que el sistema desarrollado cumpliera con los requisitos propuestos para el estudio de caso.

La primera métrica determinada es el **Porcentaje de Reutilización** (R) que estima el número de líneas de código (sin comentarios ni líneas en blanco) de un producto P generado a partir de IRArduino-SPL [159] en términos de porcentaje. La fórmula definida para el cálculo del porcentaje de reutilización es la siguiente.

$$R = \frac{LR}{TL} * 100 \quad (1)$$

Donde LR es el número de líneas de código reutilizadas o aportadas por la SPL y TL es el número de líneas totales de software desarrolladas.

Otra métrica que ayuda a cuantificar el índice de reutilización es la denominada **Líneas de Código de Características** (LoF), que se define como el número de líneas de código que están vinculadas a expresiones de características o que son responsables de la implementación de una determinada característica, para un producto P generado a partir de IRArduino-SPL [163], en términos de porcentaje.

$$LoF = \frac{LRF}{TL} * 100 \quad (2)$$

Donde LRF es el número de líneas de código que están vinculadas a una característica o feature, mientras que TL es el número de líneas totales del software desarrollado.

Finalmente, se emplea la métrica denominada **Número de Características** (NoF), que estima el número de características de un programa para un determinado producto P generado a partir de IRArduino-SPL [158].

$$NoF = \#features \quad (3)$$

Donde $\#features$ se considera el número de características o *features* proporcionados por la herramienta.

Esfuerzo de desarrollo de software

La estimación del esfuerzo de desarrollo de software (EDS) intenta predecir el trabajo (en términos de horas-persona o dinero) necesario para desarrollar un software de calidad [162]. En particular, en esta investigación, se utiliza el esfuerzo de desarrollo de software para determinar cuál es el trabajo necesario para producir un producto P

derivado de IRArduino-SPL. Para ello, se utiliza el producto entre el tiempo (horas) y las personas (estudiantes) que trabajan en el proyecto.

$$EDS = \#P * \#H \quad (4)$$

Donde $\#P$ es el número de personas que trabajan en el proyecto, así como $\#H$ es el número de horas trabajadas.

Percepción de usabilidad

La usabilidad se define como el grado en que el software puede ser utilizado por consumidores específicos para alcanzar objetivos cuantificados con eficacia, eficiencia y satisfacción en el contexto de uso [164]. Para evaluar la satisfacción percibida por los usuarios que emplearon IRArduino-SPL, se aplicó un cuestionario estandarizado de 16 preguntas propuesto por Sauro ***et al***. [165]. Este tipo de test se utiliza ampliamente para medir la satisfacción percibida por los usuarios de un producto informático, permitiendo interpretar qué aspectos del sistema son poco claros y cuáles son satisfactorios. Se obtienen cuatro puntuaciones: una global y tres subescalas que se definen de la siguiente manera.

- **General**: promedio de las respuestas a los ítems 1 a 16 (todos los ítems).
- **Calidad de la herramienta**: media de los ítems 1 a 6.
- **Calidad de la información**: media de los ítems 7 a 12.
- **Calidad de la interfaz**: media de los ítems 13 a 16.

Asimismo, la fórmula que se definió para medir la percepción de usabilidad de los usuarios que emplearon IRArduino-SPL es la siguiente.

$$Pu = \frac{\sum_{i=1}^{n} RA_i}{n} \quad (5)$$

Donde Pu es la percepción de usabilidad, RA_i es el resultado obtenido para la pregunta i, que puede tomar valores de uno a siete, n es el número de tópicos evaluados en la encuesta por los estudiantes.

Finalmente, en la siguiente tabla se puede observar un resumen de los parámetros considerados en el estudio de caso en relación con los indicadores estudiados.

Tabla 21. Indicadores y métricas empleadas en el presente estudio de caso.

Indicador	Métricas	Términos
Índice de reutilización	Porcentaje de reutilización de software (R) $R = \frac{LR}{TL} * 100$	LR número de líneas de código reutilizadas o aportadas por la SPL. TL número de líneas totales del software desarrollado.
	Líneas de código de características (LoF) $LoF = \frac{LRF}{TL} * 100$	LRF número de líneas de código de la característica que están vinculadas a una *feature*. TL número de líneas totales del software desarrollado.
	Número de características (NoF) $NoF = (0, N)$	N es el número de todas las características.
Esfuerzo de desarrollo de software	Esfuerzo de desarrollo (EDS) $EDS = \#P * \#H$	$\#P$ personas que están implicadas en el desarrollo del producto. $\#H$ horas dedicadas al desarrollo del producto.
Percepción de Usabilidad	Percepción de usabilidad (Pu) $Pu = \frac{\sum_{i=1}^{n} RA_i}{n}$	Pu Percepción de usabilidad. RA_i Resultados de la pregunta i. n Número de ítems evaluados en la encuesta por los estudiantes.

5.3.6. Diseño del estudio de caso

En la Tabla 22 resume el diseño y la ejecución de las actividades con sus respectivos tiempos e instrumentos de apoyo previstos para el estudio de caso.

Tabla 22. Resumen de las actividades del estudio de caso.

Actividades	Duración planeada	Instrumentos de apoyo
Actividad 1. Socializar y contextualizar el estudio de caso**.**	30 minutos	Presentación de la introducción al estudio de caso y los elementos conceptuales de la propuesta.
Actividad 2. Presentación de IRArduino-SPL.	15 minutos	Documento con la descripción del enfoque de reutilización y presentación de la herramienta.
Actividad 3. Socialización de los resultados preliminares.	15 minutos	Primera iteración de IRArduino-SPL y presentación de resultados preliminares.
Descanso	10 minutos	Ninguno
Actividad 4. Aplicación de IRArduino-SPL.	120 minutos	Documento de requisitos, documento con guía de IRArduino-SPL, escenarios y plantillas de resultados para su ejecución.
Actividad 5. Evaluación de la aplicación y ejecución de la	15 minutos	Encuesta de percepción

encuesta.		
Actividad 6. Reunión posterior	15 minutos	Documento de proyectos futuro
Tiempo total aproximado: 220 minutos		

- **Actividad 1.** En esta primera parte, se socializa y contextualiza de forma general en qué consiste el estudio de caso. Para ello, se realizó una presentación oral empleando medios audiovisuales para introducir e informar a los participantes sobre las actividades que se iban a desarrollar, así como dar a conocer y aclarar algunos conceptos utilizados en la misma.

- **Actividad 2.** En esta actividad se realizó una presentación oral para dar a los participantes una visión general de IRArduino-SPL como estrategia de reutilización.

- **Actividad 3.** Esta actividad está enmarcada por una presentación oral de los resultados de la aplicación de IRArduino-SPL, con énfasis en la primera implementación de la estrategia propuesta, la abstracción del dominio y los resultados preliminares. Estos insumos fueron empleados para la segunda versión de la SPL (IRArduino-SPL), dando un contexto de aplicación y proporcionando elementos para su ejecución.

- **Actividad 4.** Al inicio del desarrollo de la actividad, es necesario guiar a los participantes en la aplicación de la estrategia de reutilización y el material de apoyo. En la segunda parte de esta actividad, los participantes del estudio de caso aplicarán IRArduino-SPL empleando la guía para efectuar la estrategia en un SR industrial. Como resultado, los participantes implementarán un robot industrial funcional que supere los requisitos propuestos.

- **Actividad 5.** En esta actividad se evaluará la propuesta con los participantes del estudio de caso aplicando una encuesta de percepción y se recogerán sugerencias sobre IRArduino-SPL.

- **Actividad 6.** En esta última actividad se celebrará una reunión con los participantes del estudio de caso para identificar futuros proyectos o trabajos asociados a la investigación mediante la técnica de lluvia de ideas.

5.4. IRArduino-SPL: Ejecución, Análisis y Discusión del Estudio de Caso

A continuación, se presentan los resultados cuantitativos y cualitativos del estudio de caso, las respuestas concretas de la encuesta a los estudiantes, el registro y el cálculo de los valores obtenidos, las observaciones, el análisis de los códigos resultantes empleando IRArduino-SPL y los participantes en el estudio.

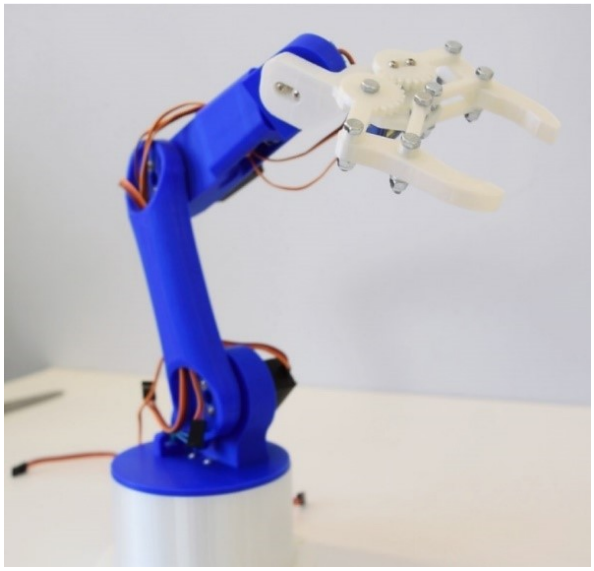


Figura 37. Brazo robótico que se utilizó para la ejecución del estudio de caso.
Fuente: [166].

El estudio de caso se ejecutó con estudiantes de la Corporación Universitaria Comfacauca – Unicomfacauca (Apéndice G), quienes fueron divididos aleatoriamente en grupos de dos personas y se les entregó una hoja de especificaciones con los movimientos, elementos (sensores y actuadores) y tareas a realizar y contener el brazo robótico. El tiempo medio de la actividad fue de cuatro horas, con descansos de quince minutos cada dos horas. Para garantizar que el hardware no fuera un elemento que influyera en los resultados del estudio de caso, se utilizó el mismo SR industrial con Arduino (Figura 37) para testear el código fuente generado por IRArduino-SPL para todos los grupos, garantizando así que el hardware no sesgara los resultados. Finalmente, la Figura 38 expone algunas evidencias fotográficas de los estudiantes de Unicomfacauca desarrollando el estudio de caso.

Figura 38. Grupos aleatorios desarrollando el estudio de caso empleando IRArduino-SPL para la construcción del software del SR industrial con Arduino.
Fuente: Elaboración propia.

Ahora, es importante indicar que a los participantes del estudio de caso se les entregaron nueve posibilidades de sensores (Figura 39) para implementar en el brazo robótico, lo que permitió determinar si la extensibilidad de la herramienta desarrollada es plausible.

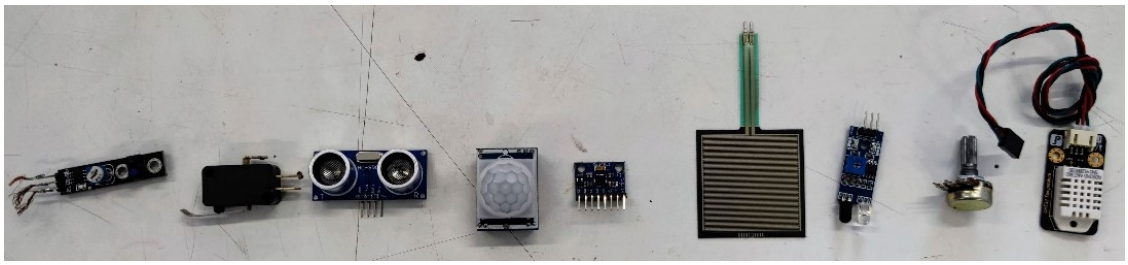


Figura 39. Diferentes opciones de sensores para la ejecución del estudio de caso utilizando IRArduino-SPL.

5.4.1. Análisis cualitativo

En esta sección se detallan los resultados relacionados con la calidad de la herramienta y la ejecución del estudio de caso percibidos por los estudiantes, así

como los principales puntos de vista de los desarrolladores sobre IRArduino-SPL y las dificultades que tuvieron al utilizar las herramientas, así como los puntos positivos encontrados.

A nivel general, la percepción de los desarrolladores (estudiantes) cuando se les socializa la herramienta y el estudio de caso fue, en principio, de curiosidad, ya que se trata de un tema poco difundido en las ingenierías relacionadas con el hardware (en este caso, la ingeniería mecatrónica). En particular, llama la atención que un concepto como el de abstracción de alto nivel sobre un dominio específico fue rápidamente comprendido, gracias al uso de analogías conocidas por ellos como el concepto de capa HAL, que a priori es un concepto similar aplicado en un dominio distinto. Asimismo, el concepto de línea de productos de software, que tiene su equivalente en la manufactura industrial, también se comprendió y adoptó rápidamente. Por otro lado, los elementos tradicionales de la ingeniería de software, como el MDE, los casos de uso, los documentos de especificación de requisitos de software, entre otros, fueron conceptos que los alumnos necesitaron algún tiempo para comprender o incluso consultar por su cuenta. Los problemas mencionados exponen que conceptos muy específicos de la ingeniería de software son desconocidos por los estudiantes y generan que no se aprovechen las principales ventajas de este tipo de técnicas de reutilización, lo que muchas veces lleva a los estudiantes a codificar repetidamente las mismas soluciones en contextos similares, sin ser conscientes de los patrones y estructuras que pueden reutilizarse. Por lo tanto, los trabajos futuros que surjan de esta investigación podrían centrarse en la realización de cursos de ingeniería de software específicos para programas de formación centrados en el hardware, para que los estudiantes comprendan estas perspectivas, desarrollen habilidades y las adopten de forma natural desde su formación, permitiéndoles aplicar la reutilización de software en la industria de una mejor manera cuando sea el caso.

Otra de las opiniones expresadas por los alumnos está relacionada con el hecho de que la herramienta utiliza terminología común dentro del dominio de la robótica industrial, como por ejemplo, son las abstracciones denominadas *articulaciones*, *sensores* o *actuadores*, lo que permite que los desarrolladores no tengan que transpolar sus conocimientos del dominio a la herramienta, ya que están incorporados en la librería, permitiendo que la programación del robot sea más natural y facilitando en cierto modo el desarrollo de software para el sistema

electromecánico. También, es importante señalar que los alumnos tuvieron que pasar por cierta curva de aprendizaje para emplear con éxito IRArduino-SPL, debido a que las abstracciones y lógicas de alto nivel que contiene son muy específicas del dominio. Un ejemplo evidente de esto son las denominadas *Activities*, que son las conceptualizaciones que permiten que cuando un sensor detecta un cambio en el entorno, se realicen las acciones correspondientes, permitiendo guardarlas y utilizarlas en futuras lógicas. Por lo tanto, en futuros trabajos se podrían desarrollar mejores instrumentos para que los alumnos se acondicionen de mejor forma para el uso de la SPL. Otro punto que llama la atención tiene que ver con el hecho de que se observaron algunas dificultades en el conocimiento técnico de la OOP, donde algunos estudiantes no entienden facilidades como la herencia o el polimorfismo que se explotan dentro de la estrategia de reutilización propuesta.

Un factor que no se midió en el estudio de caso es la extensibilidad de la herramienta porque no se encontraron métricas fiables para cuantificar en este contexto. Para este estudio de caso, la extensibilidad está relacionada con poder añadir nuevos elementos de hardware que puedan ser programados a través de IRArduino-SPL, esto es importante porque en la plataforma Arduino hay muchas posibilidades de sensores y actuadores, por lo que sería contraproducente cerrar las posibilidades de añadir nuevos dispositivos. En el estudio de caso, los alumnos tuvieron la posibilidad de seleccionar una variedad de 9 sensores para el SR industrial que desarrollaron, de los cuales 4 ya estaban incorporados en la herramienta y los 5 restantes podrían ser añadidos a IRArduino-SPL en caso de que tuvieran intención de emplearlos. Concretamente, en la implementación, un grupo utilizó un sensor y lo añadió a la herramienta (sensores de gas MQ). Para ello, los alumnos tuvieron que consultar inicialmente cómo se incorporan otros sensores y adaptarlo al sensor de gas, teniendo éxito en el proceso, debido a que el funcionamiento del sensor era correcto. Esto permitió comprobar que los alumnos fueron capaces de incorporar en un periodo de 15 o 20 minutos un nuevo sensor, indicando que la extensibilidad en la herramienta es plausible, y siguiendo ciertas reglas lógicas en IRArduino-SPL, se podrían incorporar nuevas posibilidades de dispositivos como sensores o actuadores.

5.4.2. Tiempos de desarrollo del software

En este apartado se muestra el registro de los datos tomados en la ejecución del estudio de caso, donde los sujetos investigados desarrollaron el software para el

funcionamiento de un sistema robótico industrial basándose en la guía provista e IRArduino-SPL. La medición del tiempo se tomó en horas. En la Tabla 23 se expone la información recolectada de los tiempos requeridos por cada grupo para desarrollar el software de este tipo de dispositivos electromecánicos.

Tabla 23. Registros de tiempos para la ejecución del estudio de caso con IRArduino-SPL.

Desarrollo de software para SR industriales (en horas)				
X	Tiempo desarrollo software	Tiempo de ajustes implementación	Tiempo total	IRArduino-SPL
Grupo 1	2.0	0.3	2.3	Si
Grupo 2	2.3	0.2	2.5	Si
Grupo 3	2.1	0.2	2.3	Si

En cuanto al tiempo empleado por los participantes en la ejecución del estudio de caso, se puede evidenciar que para dos de los tres grupos el tiempo necesario para desarrollar el software del robot industrial no fue suficiente en relación con el tiempo esperado, por lo que podría ser un indicio de que la herramienta tiene una curva de aprendizaje pronunciada y que los estudiantes necesitan tiempo para entender las abstracciones realizadas sobre el dominio y la aplicabilidad de la herramienta. Esto apoya en que los estudiantes tuvieron dudas en la comprensión de la lógica propuesta por los desarrolladores para IRArduino-SPL porque necesitan aclaraciones sobre las abstracciones propuestas, los métodos para introducirlas, así como un tiempo de acondicionamiento para reorientar la forma en que suelen desarrollar (programación procedimental) para el hardware, a un enfoque de programación orientada a objetos. Lo anterior pueden ser las razones por las que los estudiantes tardaron más de lo esperado en el estudio de caso y también podría ser una posible mejora para futuras iteraciones de la propuesta de reutilización.

5.4.3. Índice de reutilización de software

El índice de reutilización de software para este estudio de caso, se compone de un conjunto de métricas que permiten determinar cuánto ahorró IRArduino-SPL al grupo de desarrolladores (estudiantes), para conocer la utilidad real de la herramienta en relación con el software requerido por este tipo de sistemas. El índice de reutilización está compuesto por tres métricas: el porcentaje de reutilización propuesto en [159], las líneas de código de características y el número de características propuesto en [158].

6.3.2.1. Porcentaje de reutilización de software

En cuanto al porcentaje de reutilización de software cuantificado en el estudio de caso, los valores se sitúan entre el 38 y el 43% (Tabla 24), que comparado con la primera iteración del SPL, el incremento de la reutilización es de casi un 10% más, aunque hay que tener en cuenta que son expresiones diferentes de la Ingeniería de Aplicación de la misma propuesta, pero es útil para tener una referencia.

Tabla 24. Registros de porcentaje de reutilización de software en la ejecución del estudio de caso con IRArduino-SPL.

Porcentaje de reutilización de software				
X	Porcentaje de reutilización de software	SLOC aportadas por el desarrollador	SLOC aportadas por IRArduino-SPL	SLOC totales
Grupo 1	43%	241	182	423
Grupo 2	36%	246	139	385
Grupo 3	38%	245	149	391

Si el análisis se centra en las SLOC totales de cada uno de los grupos, se puede observar a nivel general que la variación es de 38 líneas de código entre los grupos 1 y 2, esto puede deberse a que los primeros programan un movimiento extra (no solicitado en los requisitos) para equilibrar el robot y que los servomotores (actuadores) no pierdan torque, lo cual es un problema externo a la propuesta propio del hardware del robot. Los grupos 2 y 3 tienen un porcentaje de reutilización similar presentando una diferencia del 2%, al igual que en el total de líneas de código. Este pequeño contraste se manifiesta porque el grupo 2 en la selección de sensores optaron por un sensor de gas que no se encontraba en el repositorio de IRArduino-SPL, por lo que tuvieron que crear el código para este sensor utilizando la lógica de la herramienta propuesta, a diferencia del grupo 3 que en la selección de sensores prefirieron dos sensores (sensor PIR y sensor ultrasonido) que ya se encontraba en el repositorio por lo que no fue necesario crear el código por parte de los estudiantes.

6.3.2.2. Líneas de código de características

La métrica LoF permite determinar si una fracción pequeña del código fuente es variable o no. Un valor alto indica que hay muchas líneas de código dedicadas a la realización de la característica. El valor calculado revela la cantidad de código variable y, por tanto, la complejidad de la mantenibilidad.

Las sentencias de código fuente relacionadas con el llamado a características o su implementación son pocas, debido al número de *features* que se plantean para el modelo de rasgos de la versión 2.0 de IRArduino-SPL. A nivel general, se puede indicar que las características más implementadas en el estudio de caso son aquellas con un nivel de abstracción más bajo, estas son por ejemplo las *features* como *Servo*, *Ultrasonic*, *addMovimientoDelante* o *addMovimientoAtras*, lo que podría significar que para lograr mayores niveles de reutilización en el código, estas sentencias específicas deberían ser abstraídas a un nivel superior, y vincularlas directamente a los elementos de hardware. La siguiente tabla muestra los valores encontrados para esta métrica.

Tabla 25. Líneas de código de características cuantificadas en la ejecución del estudio de caso con IRArduino-SPL.

Líneas de código de características			
X	Porcentaje SLOC vinculadas a *features* (LoF)	SLOC vinculadas a *features*	SLOC totales
Grupo 1	16%	70	423
Grupo 2	15%	58	385
Grupo 3	17%	66	391

Aunque efectivamente, el valor obtenido en LoF podría no interpretarse por ser un resultado calculado, algunos autores relacionan esta métrica en general, con que un valor alto sugiere una alta complejidad en las tareas de mantenimiento, debido a que cada *feature* es un bloque de código que implementa una o varias funcionalidades y su modificación puede acarrear una serie de consecuencias en el sistema desarrollado. En concreto, si se compara los resultados del estudio de caso con otras investigaciones, los valores de la métrica LoF son un poco más altos de lo normal, en el caso de IRArduino-SPL los resultados muestran entre un 15 y un 17%, mientras que las investigaciones consultadas presentan valores en torno al 13%, sin olvidar que se trata de líneas de producto implementadas únicamente en dominios relacionados con el software. Por lo tanto, en próximas iteraciones se debería trabajar para reducir el valor de la métrica y así mejorar la mantenibilidad de IRArduino-SPL.

6.3.2.3. Número de características

La métrica NoF está directamente relacionada con el número de características utilizadas o instanciadas en la implementación de una línea de productos de

software. Por lo general, cuando el valor es más cercano a N, más características tiene un programa y su mantenimiento puede ser más complejo [158].

Las implementaciones de IRArduino-SPL realizadas por los grupos fueron diversas porque se les dio la posibilidad de incluir en su robot diferentes elementos de hardware soportados por la herramienta o de crear algunos sensores o actuadores, esto también se refleja en el número de características empleadas por los estudiantes. Por lo tanto, en la métrica NoF se puede observar cierta variedad en el número de características empleadas por cada grupo. La diversidad de características empleadas por los grupos también puede deberse a la lógica propia implementada en IRArduino-SPL, donde *features* como *robotAssembly* o *armarRobot* son necesarias para que el código funcione correctamente.

En la Figura 40 se muestran las características instanciadas por cada grupo en la realización del estudio de caso de IRArduino-SPL. Las imágenes muestran las características utilizadas con un recuadro rojo y el número de veces que se utilizaron.

En concreto, el grupo 1 instancia 65 características que son una representación del producto generado (SR industrial con Arduino) de IRArduino-SPL para su caso específico. Este grupo empleó un sensor PIR, un sensor de ultrasonido y 5 servomotores. En lo referente a la sensórica, se puede indicar que también se utilizan 4 características emparentadas con las funcionalidades de lectura de datos (*readDataFromSensorP* y *readDataFromSensorU*), que podrían en futuras iteraciones no ser un rasgo aparte, sino incluirse dentro de la *feature* del sensor, para no complicar la mantenibilidad de la SPL. También se puede observar que se empleó la característica *addActivity*, que está vinculada a las actividades recurrentes que ejecutan los robots (movimientos) tras detectar un cambio en el ambiente (sensor). En este caso, los estudiantes utilizaron este rasgo para que una vez que el sensor PIR detecta; el robot haga una serie de movimientos preprogramados que tienen que ver con agarrar un objeto y dejar al robot en una posición central de sus ejes, permitiendo ahorrar líneas de código fuente para que el robot regrese a una posición central después de agarrar el objeto.

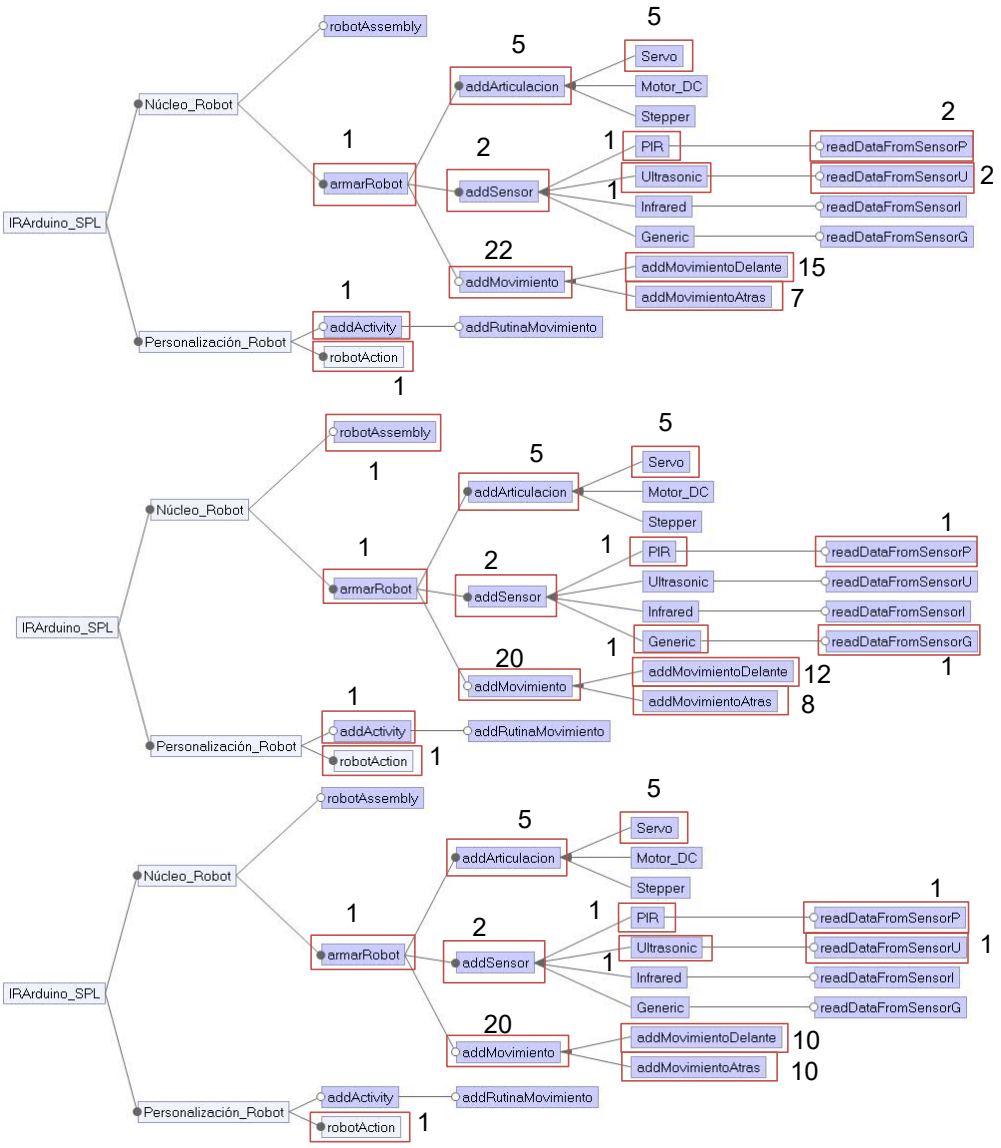


Figura 40. Características empleadas por cada uno de los grupos en la ejecución del estudio de caso para IRArduino-SPL.
Fuente: Elaboración propia.

Para el grupo 2 puede observarse que el número de *features* (60) son similares a las del grupo 1, con sólo 5 rasgos de diferencia, esto se debe a que los sensores de este grupo censan dos veces para asegurar que la detección sea correcta lo cual no ocurre para el grupo 2, esto está relacionado más con el hardware subyacente y su exactitud y no con el código del robot. De igual forma, los estudiantes del grupo 2 utilizaron una lógica similar a la del grupo 1, utilizando el rasgo *addActivity* agregan actividades predeterminadas cuando el robot detecta un cambio en el ambiente. Finalmente, cabe destacar que la feature *Generic* en los sensores se utilizó porque

los estudiantes querían utilizar un sensor de gas que no estaba entre las opciones proporcionadas en IRArduino-SPL.

Por último, en el grupo 3, instancian 58 características, que en comparación con los otros grupos, es el número más pequeño, lo que llama la atención ya que también obtuvieron el menor porcentaje de reutilización. Por lo tanto, se podría indicar que para IRArduino-SPL, cuantas más características se utilicen, mayor será la reutilización, pero la mantenibilidad del producto puede ser más complicada porque cualquier modificación en el producto final podría requerir una serie de cambios en otras *features* vinculadas.

5.4.4. Esfuerzo de desarrollo de software

La estimación del esfuerzo de desarrollo es el proceso que busca determinar el esfuerzo (expresado en términos de horas-persona) requerido para desarrollar o mantener el software basado en una entrada incompleta, incierta y ruidosa [167].

En el caso de la presente investigación, el esfuerzo de desarrollo se determina mediante la siguiente fórmula matemática.

$$EDS = \#P * \#H \tag{4}$$

Puntualmente, en la ejecución del estudio de caso de IRArduino-SPL, el esfuerzo de desarrollo para los tres grupos fue muy similar, aunque con mejoras apenas perceptibles en los grupos que presentaron mayor reutilización utilizando la herramienta propuesta.

Tabla 26. Esfuerzo de desarrollo de software en cada uno de los grupos para la ejecución del estudio de caso con IRArduino-SPL.

X	Esfuerzo de desarrollo de software
Grupo 1	4.0 horas-persona
Grupo 2	4.6 horas- persona
Grupo 3	4.2 horas- persona

En cuanto al esfuerzo de desarrollo de software realizado por los tres grupos, se puede indicar que las cifras medias obtenidas se sitúan entre 4 y 4,6 horas- persona. Sin embargo, al no existir un grupo de control con el que contrastar los resultados bajo los mismos parámetros, no es posible establecer un paralelismo entre el uso o no de la herramienta, pero gracias a la métrica de reutilización del apartado anterior, se puede afirmar que la herramienta ayuda a facilitar el desarrollo de software,

fomentando la reutilización en un dominio poco extendido. Además, no se debe omitir las ventajas de IRArduino-SPL más allá de la reutilización, como son el hecho de proporcionar abstracciones de alto nivel que permiten a los desarrolladores hablar un lenguaje cercano al dominio de aplicación o un repositorio de movimientos preestablecidos, entre otras ventajas.

5.4.5. Percepción de usabilidad de IRArduino-SPL

El Cuestionario de Usabilidad del Sistema de Estudio Posterior (PSSUQ, por sus siglas en inglés Post-Study System Usability Questionnaire) es una prueba estandarizada de 16 preguntas que se utiliza ampliamente para medir la satisfacción percibida por los usuarios de un software, una web o un producto final. Existen varias versiones de este cuestionario, pero la más utilizada es la versión 3, que fue perfeccionada por Sauro y Lewis en un estudio en el que se promediaron cerca de 200 evaluaciones de diferentes productos de software en distintos dominios y se establecieron puntuaciones globales para que la comunidad investigadora pudiera tener una referencia teórica de comparación. El PSSUQ se mide en una escala Likert de 7 puntos (más una opción de no respuesta), donde la puntuación global se calcula ponderando las puntuaciones de las 16 preguntas, la usabilidad del sistema con las preguntas 1 a 6, la calidad de la información con las preguntas 7 a 12 y la calidad de la interfaz con las puntuaciones medias de las preguntas 13 a 15. Finalmente, es importante aclarar que la puntuación del PSSUQ comienza con 1 (totalmente de acuerdo) y termina con 7 (totalmente en desacuerdo). Cuanto menor sea la puntuación, mejor será la satisfacción percibida del sistema evaluado [165].

Se utilizó el cuestionario estandarizado indicado anteriormente, después de la ejecución del estudio de caso, donde cada uno de los grupos, después de realizar el SR industrial con Arduino e IRArduino-SPL, respondió a las 16 preguntas. Los resultados ponderados se pueden ver en la siguiente tabla.

Tabla 27. Percepción de usabilidad para la ejecución del estudio de caso con IRArduino-SPL.

Unidad de análisis				
X	Utilidad del sistema	Calidad de la información	Calidad de la interfaz	Promedio general
Grupo 1	3.0	4.83	5.50	4.44
Grupo 2	3.16	6.0	5.25	4.80
Grupo 3	2.66	5.83	4.0	4.16

Las puntuaciones globales obtenidas por cada uno de los tres grupos en la media general de los sistemas evaluados por Sauro *et al.* en su estudio no se acercan a las medias ponderadas calificadas por los estudiantes. En este sentido, se ha comprobado que para una puntuación media igual o inferior a 2.82, se puede considerar como un sistema usable. Es importante indicar que la puntuación obtenida por IRArduino-SPL no alcanzó el valor medio (4.46 de promedio general), por lo tanto; es una característica a trabajar en futuras iteraciones de la SPL desarrollada. Esto pudo deberse a que esta segunda versión se centró en establecer los principales activos núcleo de la línea basados en la OOP, para ser la base principal en la que se fundamentan las futuras iteraciones, que se centrarán en aspectos como la usabilidad, la interfaz y las nuevas funcionalidades en el lado del cliente.

Un análisis más profundo de cada una de las subescalas del test PSSUQ muestra que los mejores resultados están relacionados con la utilidad del sistema con una media de 2,94, lo que podría indicar que, en general, los alumnos se sintieron cómodos con las funcionalidades que ofrece IRArduino-SPL, permitiéndoles cumplir con las tareas y escenarios propuestos en el estudio de caso. A diferencia de los resultados obtenidos anteriormente, las subescalas denominadas calidad de la información (5.55) y calidad de la interfaz (4.91) obtuvieron puntuaciones altas, esto puede estar relacionado con el hecho de que los elementos de la interfaz de la librería son los mismos que los del IDE de Arduino, ya que la plataforma no ofrece la posibilidad de una interfaz específica para cada una de las librerías, sino que debe ser la general del ecosistema. La opinión negativa no es algo aislado del estudio de caso y, por el contrario, es recurrente dentro de la comunidad, ya que, debido a la masificación en el uso de este tipo de placas de desarrollo, los programadores demandan cada vez más tanto a nivel de hardware como de software, y la opinión generalizada es que se debe hacer un cambio drástico para profesionalizar y/o ayudar al desarrollo en Arduino [168]. Se han llegado a hacer diferentes propuestas para mejorar el desarrollo sobre la plataforma [169], [170] e incluso se ha planteado un nuevo IDE por parte de la empresa encargada de estos microcontroladores. Teniendo en cuenta lo anterior, otro trabajo que podría derivarse de esta investigación puede estar relacionado con el desarrollo de un programa informático que cuente con las facilidades que ofrece Arduino, pero enfocado al dominio de los robots que utilizan estos microcontroladores, con especial énfasis en elementos como la virtualización, la dinámica de movimiento, acondicionamiento de señales y

reutilización de software, entre otros aspectos concretos de estos sistemas electromecánicos.

Finalmente, con los resultados de la ejecución del estudio de caso se pueden obtener las siguientes pruebas de hipótesis:

- *H.1.* es aceptada. De este modo, se puede afirmar que la reutilización de software en IRArduino-SPL es superior al 31%.
- *H.2.* es aceptada. De este modo, se puede afirmar que el esfuerzo de desarrollo para generar una solución específica a partir de IRArduino-SPL fue inferior a 5 horas-persona.
- Con los resultados de la encuesta a los estudiantes y con una puntuación global de 4.46, se puede rechazar *H.3.* Por lo tanto, se puede decir que los desarrolladores no están de acuerdo con la usabilidad general de IRArduino-SPL.

5.4.6. Amenazas a la validez

Para garantizar la veracidad de los resultados del estudio de caso y evitar que estén sesgados por el juicio del investigador que realiza el estudio, existen diferentes formas de clasificar los aspectos de validez y las amenazas a la validez, las cuales fueron consideradas y mitigadas durante el diseño y ejecución de la investigación. Por lo tanto, en este proyecto se optó por practicar la validez de constructo, la validez interna y la validez externa.

Validez de constructo: Una de las amenazas a la validez es que no todos los estudiantes interpretan las preguntas de la encuesta de la misma manera, por lo que para minimizar este efecto los instrumentos utilizados en el estudio de caso fueron validados por expertos y docentes en el dominio de los SR industriales, microcontroladores e ingeniería de software. Otra amenaza identificada en el estudio de caso tiene que ver con la incorporación de nuevos conceptos y lógicas relacionadas con la herramienta desarrollada y la ingeniería de las líneas de productos de software. Para reducir esta amenaza en el marco del estudio de caso, se planificó una actividad inicial en la que se socializa y contextualiza a los alumnos del grupo que utiliza IRArduino-SPL y la estrategia de reutilización para lograr una unificación de conceptos.

Validez interna: Una amenaza a la validez del estudio de caso es el tiempo invertido en esta actividad; al ser una jornada larga, los alumnos en las fases finales del estudio pueden sentir cansancio, lo que puede influir directamente en los resultados. Para mitigar esta amenaza, se definen una serie de pausas en el transcurso de la ejecución en las que no debe haber comunicación entre los participantes de los diferentes grupos.

Validez externa: Una amenaza a la validez en el estudio de caso tiene que ver con el conocimiento previo y de fondo en la construcción de un SR industrial. Para minimizar, se consideraron todos los estudiantes de ingeniería mecatrónica de Unicomfacauca a partir del quinto semestre, dado que a partir de este momento se garantiza que los estudiantes hayan cursado al menos una asignatura sobre microcontroladores y robótica industrial, asegurando así que tengan los conocimientos en los diferentes dominios involucrados en la construcción de un SR industrial. Otra amenaza para la validez está relacionada con el hardware del SR industrial que deben construir en el estudio de caso, debido a que la relación entre el hardware y el software puede influir en el rendimiento del robot y esto podría influir en el resultado de cada grupo. Tratando de mitigar esta situación, se les facilita el mismo robot industrial para cada uno de los grupos, lo que permite eliminar cualquier sesgo emparentado con el hardware de estos sistemas. Finalmente, es importante indicar que la validez externa de este estudio depende en cierta medida de la similitud del contexto en el que se aplicará el método con el de este estudio de caso

5.5. Resumen de los resultados derivados del capítulo

En este capítulo se validó la contribución de la propuesta de reutilización denominada IRArduino-SPL a través de una prueba de concepto y un estudio de caso, lo que permitió observar su viabilidad en el dominio y su contribución en un contexto académico, se utilizaron métricas de reutilización, se determinaron las amenazas a la validez y se realizó el análisis de los resultados obtenidos. Los principales productos obtenidos en este capítulo son:

- Artículo de investigación aceptado en la revista Electronics (Scopus) denominado “A Software Products Line for Industrial Robots with Arduino in an Educational Context: IRArduino-SPL”.

- Participación como ponente en el XI congreso internacional de nuevas tecnologías (Expotecnología 2021).
- Ejecución de una prueba de concepto que permitió determinar la viabilidad de la implementación de la SPL propuesta y recolección de experiencias para aplicar en próximas iteraciones de la estrategia de reutilización.
- Diseño y ejecución de un estudio de caso teniendo en cuenta los lineamientos planteados por Runeson *et al* en el contexto de la ingeniería de software.
- Análisis y discusión de los resultados obtenidos en el estudio de caso y la prueba de concepto.
- Determinación de posibles trabajos futuros que se pueden derivar de la investigación realizada.

Capítulo 6

Conclusiones, Limitaciones y Trabajos Futuros

En este trabajo de grado se plantea la implementación de una línea de productos de software para robots industriales con Arduino. Dado que existen pocas propuestas de reutilización en la plataforma Arduino y muy escasas en el dominio de los robots industriales con estos microcontroladores, se propone una estrategia de reutilización de software para estos dispositivos que permite acelerar el desarrollo de software. Indirectamente, la propuesta también permite que los desarrolladores (estudiantes) puedan beneficiarse de la reutilización planificada, así como entender y familiarizarse con enfoques de ingeniería de software desde su formación académica, permitiendo que así que la reutilización de software en la industria de la robótica esté en el escenario y sea más factible ser aplicada en el desarrollo de soluciones con estos dispositivos.

6.1. Conclusiones

En este proyecto de investigación se desarrolla e implementa una estrategia de reutilización de software (IRArduino-SPL) basada en la ingeniería de líneas de productos para sistemas robóticos industriales con Arduino. Para esto, se ejecutaron dos iteraciones de IRArduino-SPL en las que se siguieron los procesos esenciales de la Ingeniería de Dominio e Ingeniería de Aplicación en un contexto académico. En este contexto, se definieron activos fundamentales como el modelo de dominio, el modelo de características, el alcance y una estrategia de producción basada en la generación de código y la utilización de activos núcleo. Con la SPL construida y su verificación mediante un piloto preliminar y un estudio de caso en el contexto educativo. Los estudiantes en el desarrollo de una solución lograron reutilizar cerca del 40% del código fuente, lo que implica un menor esfuerzo de programación. Adicionalmente la reutilización les implicó elevar el nivel de abstracción en el marco del dominio, lo cual permitió valorar y desarrollar este tipo de habilidad en su

formación como ingenieros en mecatrónica. Esto permite acelerar el desarrollo de software en términos de reutilización y, por tanto, de eficiencia en la programación y construcción de este tipo de sistemas electromecánicos, posibilitando a los desarrolladores (estudiantes) centrarse en otras preocupaciones propias de un dominio tan específico como la robótica o la informática.

El principal resultado de esta investigación tiene que ver con el hecho de que IRArduino-SPL es una estrategia de reutilización coherente, viable, implementable y con alto potencial de reutilización de software en el dominio de Arduino, validada a través de una prueba de concepto y un estudio de caso con estudiantes de ingeniería mecatrónica, lo que permite pensar que tiene potencial en entornos educativos. Es importante indicar que, la mayor contribución de este proyecto está orientada a generar evidencia empírica de una propuesta de reutilización específica, como son los SPLs, en un dominio poco explorado desde el punto de vista de la reutilización. La implementación de IRArduino-SPL contribuye también al dominio tecnológico de Arduino, aportando conocimientos y experiencias donde la evidencia es limitada y poco clara. La SPL mostró tener un valor como herramienta educativa, permitiendo que el estudiante pueda establecer bases desde la ingeniería de software para en el futuro desarrollar robots en entornos industriales de calidad y en forma productiva. La propuesta es un insumo para que los investigadores puedan continuar con el desarrollo de esta estrategia de reutilización de software más en robots industriales con Arduino e incluso complementarla con las otras estrategias de reutilización.

Aunque IRArduino-SPL fue desarrollada e implementada como una estrategia de reutilización de software con fines académicos en un dominio poco explorado en el que se obtuvieron resultados positivos en términos de reutilización de software, como estrategia de mejora en el desarrollo de software para robots, pueden tener un impacto positivo en múltiples dominios, como la robótica de servicios, por ejemplo, el cuidado del adulto mayor, contexto en el cual se hizo una publicación. Esto brindaría ventajas en la aceleración del desarrollo de software y la incorporación de funcionalidades novedosas considerando la anticipación al cambio, beneficiando así a dominios que a priori podrían considerarse ajenos a la estrategia propuesta. En esta línea, no sólo las líneas de productos de software podrían ser una solución para mejorar la programación de robots con Arduino, sino que enfoques de reutilización como la ingeniería dirigida por modelos, el desarrollo basado en componentes, o incluso herramientas específicas como ROS o Acceleo, podrían contribuir a este

objetivo en un dominio donde las evidencias de reutilización de software son limitadas.

Finalmente, la evidencia empírica recolectada permite concluir que la reutilización de software se ha transformado en una preocupación latente en el desarrollo de los sistemas robóticos industriales, tanto para los robots que hacen uso de controladores como para los que emplean microcontroladores. Por ello, la comunidad investigadora ha tratado de desarrollar mejores y más completas estrategias de reutilización de software para estos sistemas en la industria. Dejan de lado los robots que utilizan Arduino (contexto académico), lo cual ha generado estrategias ineficaces y limitadas en este ámbito específico. Por lo tanto, la implementación de enfoques de reutilización de software en Arduino que sean maduras, específicas y replicables, permitirán a desarrolladores y estudiantes de carreras centradas en el hardware asimilar de mejor forma lo que es la reutilización de software y adoptarla desde su formación, permitiendo que la reutilización de software en Arduino y, posteriormente a nivel industrial sea más evidente llevarla a la práctica.

6.2. Limitaciones

El estudio evidenció que las abstracciones realizadas sobre el dominio pueden ser extendidas permitiendo cubrir nuevas características, incorporando nuevos enfoques de reutilización o incluso portando la estrategia de reutilización a nuevas plataformas de desarrollo. Con respecto a IRArduino-SPL, es importante señalar que los usuarios deben disponer de enlaces a fuentes externas para entender la lógica de la herramienta, además de carecer de una interfaz gráfica que facilite el desarrollo de la librería propuesta.

Al ser un estudio de caso holístico, también tiene limitaciones como las características de las unidades de estudio (alumnos), por lo que no se pueden generalizar los resultados, sobre todo porque los alumnos tienen características y conocimientos propios que no son iguales a los de los demás y que no son necesariamente una característica típica de todos. Además, el número de estudiantes empleados durante el estudio de caso puede ser una limitante dentro de la validación de la herramienta, por lo que en nuevas investigaciones alrededor de la SPL se podrían aumentar el número de participantes en el estudio, con el fin de poder generalizar los resultados.

6.3. Trabajo futuro

Este proyecto de investigación detalla algunos enfoques de reutilización de software, tecnologías emergentes y herramientas computacionales que pueden considerarse a corto, medio y largo plazo para mejorar la propuesta de reutilización en el dominio de los SR industriales con Arduino.

1. A corto plazo, la implementación de una interfaz gráfica en IRArduino-SPL es uno de los principales trabajos futuros a realizar sobre la propuesta, debido a que permitiría facilitar la programación en los SR industriales. Este software podría ser del tipo Plug-and-Play, en el que los estudiantes podrían añadir features de la SPL y personalizarlas a gusto de los desarrolladores. Esto probablemente iría ligado a la portación de la herramienta a una nueva plataforma de desarrollo, como ROS, que proporcionan facilidades en el desarrollo de interfaces gráficas o incluso permitirían incorporar nuevas funcionalidades centradas en el aprendizaje o un dominio puntual como la robótica de servicios u otro.

2. A corto plazo, las futuras iteraciones de la SPL podrían incorporar enfoques y herramientas centradas en MDE (reutilización generativa), SOA (Soluciones escalables y multitenencia con enfoque a la IoT) o CBSE (Componentes distribuidos) en combinación con esta propuesta para obtener algunas de las ventajas de estos enfoques. Además, combinar IRArduino-SPL con herramientas de generación de código fuente como Acceleo o Telosys, facilitaría las implementaciones de las abstracciones realizadas en el dominio.

3. Realizar una rigurosa tarea de representar los dominios relevantes, incorporando expertos en robótica para determinar mejor el dominio, delimitar la variabilidad de la SPL y potenciar aún más la reutilización.

4. A mediano plazo, la combinación de varios enfoques de reutilización de software podría suponer una mejora en la SPL propuesta. Elementos como la abstracción realizada sobre el dominio y su implementación en la línea de productos se verían beneficiadas de esta unión. La ingeniería dirigida por modelos, el desarrollo basado en componentes, o incluso paradigmas como la computación en la nube o en el borde (Edge computing) juegan un papel clave

en la propuesta para el futuro, donde no sólo se pretende mejorar el desarrollo de software a nivel de robots en Arduino desde el punto de vista de la abstracción, funcionalidades, entre otros, sino también incorporar nuevas vertientes como la ingeniería de la colaboración, la robótica asistencial y resolver otras cualidades del software relevantes a nivel de la industria: confiabilidad, seguridad, portabilidad, escalabilidad y desempeño.

5. A mediano plazo, respecto a aspectos complejos como la planificación y control del movimiento del robot, estos elementos requieren de un estudio detallado para adecuarlos e incorporarlos a la propuesta, debido a que cada robot es diferente con ecuaciones y algoritmos distintos, por lo que se debe realizar un proceso de abstracción centrado en estos elementos particulares, para modelar de manera general estas expresiones matemáticas y establecer una base común para ser integrada en la SPL propuesta. Para este caso se incorporarían al framework patrones de diseño tales como el adaptador (para reutilizar elementos fuera de las abstracciones), el puente (para separar abstracciones de representaciones específicas en las cinemáticas y sus algoritmos), el comando (para abstraer mejor las acciones y facilitar actividades más complejas puedan ser ejecutadas en forma segura), y los patrones visitante e intérprete para agregar nuevas acciones sin modificar las abstracciones existentes y definir gramáticas para facilitar un lenguaje específico al dominio.

6. A largo plazo y una vez desarrollados los trabajos futuros indicados donde la abstracción del dominio y su aplicación es coherente en el contexto educativo, el paso natural es integrar IRArduino-SPL en un entorno profesional, para adaptarlo a nuevas especificaciones, tecnologías y paradigmas. Lo anterior va de la mano con el desarrollo de nuevas placas de carácter más profesional, un claro ejemplo de ello es la Arduino Portenta H7 que permite ejecutar código tanto para Arduino como para otros lenguajes de programación como Python, lo que abriría un mundo de posibilidades al ecosistema y a la propuesta permitiendo ejecutar software relacionado con la inteligencia artificial o la visión por computador, que son elementos que podrían sumarse a la propuesta de reutilización.

Nuevas investigaciones y nuevos estudios de campo permitirían establecer con mayor confianza los niveles de reutilización, que seguramente serán mejores que los

propuestos en esta tesis que aborda un nivel de abstracción básico que tiene bastante camino para extender, complementar y proponer.

Lista de referencias

[1] J. Weinberg *et al.*, “A Multidisciplinary Model for Using Robotics in Engineering Education,” 2001.

[2] C. Pons, R. Giandini, and G. Arévalo, “A systematic review of applying modern software engineering techniques to developing robotic systems,” *Ing. E Investig.*, vol. 32, no. 1, pp. 58–63, 2012.

[3] D. Jawawi, S. Deris, and R. Mamat, “Software Reuse for Mobile Robot Applications Through Analysis Patterns,” *Int. Arab J. Inf. Technol.*, vol. 4, no. 3, p. 9, 2007.

[4] R. Smith, G. Smith, and A. Wardani, “Software reuse in robotics: Enabling portability in the face of diversity,” in *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, Singapore, 2005, vol. 2, pp. 933–938. doi: https://doi.org/10.1109/RAMECH.2004.1438043.

[5] A. Solis and J. Hurtado, “Reutilización de software en la robótica industrial: un mapeo sistemático,” *Rev. Iberoam. Automática E Informática Ind.*, vol. 17, no. 4, pp. 354–367, 2020, doi: https://doi.org/10.4995/riai.2020.13335.

[6] M. Pellicciari, G. Berselli, F. Leali, and A. Vergnano, “A method for reducing the energy consumption of pick-and-place industrial robots,” *Mechatronics*, vol. 23, no. 3, pp. 326–334, Apr. 2013, doi: https://doi.org/10.1016/j.mechatronics.2013.01.013.

[7] R. Möckel, L. Dahl, and S. M. Christopher, “Interdisciplinary Teaching with the Versatile Low-Cost Modular Robotic Platform EDMO,” in *Educational Robotics in the Context of the Maker Movement*, Cham, 2020, vol. 946, pp. 135–146. doi: https://doi.org/10.1007/978-3-030-18141-3_11.

[8] F. Ciccozzi, D. Di Ruscio, I. Malavolta, P. Pelliccione, and J. Tumova, “Engineering the software of robotic systems,” in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Buenos Aires, May 2017, pp. 507–508. doi: https://doi.org/10.1109/ICSE-C.2017.167.

[9] H. Ren, S. Zhou, A. Hu, and H. Cheng, “Industrial Robots and Jobs Turnover: Evidence from Chinese Firm Level Data,” *SSRN Electron. J.*, 2018, doi: https://dx.doi.org/10.2139/ssrn.3213959.

[10] S. Garnier, K. Subrin, P. Arevalo-Siles, G. Caverot, and B. Furet, “Mobile robot stability for complex tasks in naval industries,” *Procedia CIRP*, vol. 72, pp. 297–302, Jan. 2018, doi: https://doi.org/10.1016/j.procir.2018.03.101.

[11] E. Larrondo Pons, G. Cervantes Montero, and A. Sánchez Roca, “Impact of mechatronic in medicine,” *MediSan*, vol. 22, no. 04, pp. 421–432, 2018.

[12] T. Heineck, E. Goncalves, A. Sousa, M. Oliveira, and J. Castro, “Model-Driven Development in Robotics Domain: A Systematic Literature Review,” in *2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, Maringá, Brazil, Sep. 2016, pp. 151–160. doi: https://doi.org/10.1109/SBCARS.2016.12.

[13] F. Siepmann, L. Ziegler, M. Kortkamp, and S. Wachsmuth, “Deploying a modeling framework for reusable robot behavior to enable informed strategies for domestic service robots,” *Robot. Auton. Syst.*, vol. 62, no. 5, pp. 619–631, May 2014, doi: https://doi.org/10.1016/j.robot.2012.10.013.

[14] C. W. Krueger, “Software reuse,” *ACM Comput. Surv.*, vol. 24, no. 2, pp. 131–183, Jun. 1992, doi: https://doi.org/10.1145/130844.130856.

[15] Y. Hua, S. Zander, M. Bordignon, and B. Hein, “From AutomationML to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning,” in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Germany, Sep. 2016, pp. 1–8. doi: https://doi.org/10.1109/ETFA.2016.7733579.

[16] A. Bubeck, F. Weisshardt, and A. Verl, “BRIDE - A toolchain for framework-independent development of industrial service robot applications,” in *ISR/Robotik 2014; 41st International Symposium on Robotics*, 2014, pp. 1–6.

[17] H. Fischer, M. Vulliez, P. Laguillaumie, P. Vulliez, and J. P. Gazeau, “RTRobMultiAxisControl: A Framework for Real-Time Multiaxis and Multirobot Control,” *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1205–1217, 2019, doi: https://doi.org/10.1109/TASE.2018.2889813.

[18] N. de J. Londoño Ospina, “Arquitectura software para robots móviles aplicando la metodología MDASR,” *Av. En Sist. E Informática*, vol. 6, no. 3, pp. 133–144, 2009.
[19] E. Vrochidou, M. Manios, G. A. Papakostas, C. N. Aitsidis, and F. Panagiotopoulos, “Open-Source Robotics: Investigation on Existing Platforms and Their Application in Education,” in *26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Sep. 2018, pp. 1–6. doi: https://doi.org/10.23919/SOFTCOM.2018.8555860.
[20] E. Estévez, A. Sánchez García, J. Gámez García, and J. Gómez Ortega, “Aproximación Basada en UML para el Diseño y Codificación Automática de Plataformas Robóticas Manipuladoras,” *Rev. Iberoam. Automática E Informática Ind. RIAI*, vol. 14, no. 1, pp. 82–93, Jan. 2017, doi: https://doi.org/10.1016/j.riai.2016.11.001.
[21] D. Brugali, Ed., *Software Engineering for Experimental Robotics*, vol. 30. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. doi: https://doi.org/10.1007/978-3-540-68951-5.
[22] D. Brugali, “Model-Driven Software Engineering in Robotics: Models Are Designed to Use the Relevant Things, Thereby Reducing the Complexity and Cost in the Field of Robotics,” *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 155–166, Sep. 2015, doi: https://doi.org/10.1109/MRA.2015.2452201.
[23] L. Gherardi and D. Brugali, “Modeling and reusing robotic software architectures: The HyperFlex toolchain,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014, pp. 6414–6420. doi: https://doi.org/10.1109/ICRA.2014.6907806.
[24] S. Apel, D. Batory, C. Kästner, and G. Saake, “Software Product Lines,” in *Feature-Oriented Software Product Lines*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 3–15. doi: http://dx.doi.org/10.1007/978-3-642-37521-7_1.
[25] M. Garduno-Aparicio, J. Rodriguez-Resendiz, G. Macias-Bobadilla, and S. Thenozhi, “A Multidisciplinary Industrial Robot Approach for Teaching Mechatronics-Related Courses,” *IEEE Trans. Educ.*, vol. 61, no. 1, pp. 55–62, Feb. 2018, doi: https://doi.org/10.1109/TE.2017.2741446.
[26] S. R. un N. Jafri, A. Ahmed, A. Azam, U. B. Ihsan, S. N. Syed, and R. Uddin, “Assistive mobile robot for industrial and academic applications,” in *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, Islamabad, Pakistan, 2020, pp. 332–337. doi: https://doi.org/10.1109/IBCAST47879.2020.9044588.
[27] M. Bunge, *Philosophy of Science: Volume 1, From Problem to Theory*. Routledge, 2017.
[28] M. C. Camacho, J. A. Hurtado-Alegria, and P. H. Ruiz-Melenje, “Un Método Incremental para el Análisis visual de modelos de Proceso software,” *Rev. Gerenc. Tecnológica Informática*, vol. 15, no. 43, pp. 79–91, 2016.
[29] L. Northrop and P. Clements, “A framework for software product line practice, version 5.0,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pensilvania, 2012. [Online]. Available: http://www.sei.cmu.edu/plp/framework.html
[30] J. Bayer *et al.*, “PuLSE: a methodology to develop software product lines,” in *Proceedings of the 1999 symposium on Software reusability - SSR ’99*, Los Angeles, California, United States, 1999, pp. 122–131. doi: https://doi.org/10.1145/303008.303063.
[31] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009, doi: 10.1007/s10664-008-9102-8.
[32] M. Shaw, “What makes good research in software engineering?,” *Int. J. Softw. Tools Technol. Transf.*, vol. 4, no. 1, pp. 1–7, Oct. 2002, doi: https://doi.org/10.1007/s10009-002-0083-4.
[33] A. Barrientos, L. F. Peñin, C. Balaguer, and R. Aracil, *Fundamentos de robótica*, vol. 2. McGraw-Hill Madrid, 2007.
[34] M. L. P. Salamanca, N. B. Lombana, and W. J. P. Holguín, “Uso de la robótica educativa como herramienta en los procesos de enseñanza,” *Ing. Investig. Desarro. I2 D*, vol. 10, no. 1, pp. 15–23, 2010.
[35] M. Shahinpoor and M. Shahinpoor, *A robot engineering textbook*. Harper & Row New York, 1987.

[36] E. Abele, M. Weigold, and S. Rothenbücher, “Modeling and Identification of an Industrial Robot for Machining Applications,” *CIRP Ann.*, vol. 56, no. 1, pp. 387–390, 2007, doi: https://doi.org/10.1016/j.cirp.2007.05.090.
[37] J. A. S. Sánchez, *Avances en robótica y visión por computador*, vol. 38. Univ de Castilla La Mancha, 2002.
[38] A. O. Baturone, *Robótica: manipuladores y robots móviles*. Marcombo, 2005.
[39] J.-J. E. Slotine and Weiping Li, “On the Adaptive Control of Robot Manipulators,” *Int. J. Robot. Res.*, vol. 6, no. 3, pp. 49–59, Sep. 1987, doi: https://doi.org/10.1177%2F027836498700600303.
[40] A. Araújo, D. Portugal, M. S. Couceiro, and R. P. Rocha, “Integrating Arduino-Based Educational Mobile Robots in ROS,” *J. Intell. Robot. Syst.*, vol. 77, no. 2, pp. 281–298, Feb. 2015, doi: https://doi.org/10.1109/Robotica.2013.6623520.
[41] “Industrial intelligence 4.0_beyond automation,” *KUKA AG*. https://www.kuka.com/es-es (accessed Dec. 14, 2021).
[42] P. G. López, “La reutilización: un camino hacia la industrialización del desarrollo del software,” *Ens. Rev. Fac. Educ. Albacete*, no. 9, pp. 267–282, 1994.
[43] M. D. McIlroy, J. Buxton, P. Naur, and B. Randell, “Mass-produced software components,” in *Proceedings of the 1st international conference on software engineering, Garmisch Pattenkirchen, Germany*, 1968, pp. 88–98.
[44] G. C. Gini and M. L. Gini, “Dealing with world-model-based programs,” *ACM Trans. Program. Lang. Syst. TOPLAS*, vol. 7, no. 2, pp. 334–347, Apr. 1985, doi: https://doi.org/10.1145/3318.3479.
[45] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” in *ICRA Workshop on Open Source Software*, 2009, vol. 3.
[46] W. C. Lim, “Effects of reuse on quality, productivity, and economics,” *IEEE Softw.*, vol. 11, no. 5, pp. 23–30, Sep. 1994, doi: 10.1109/52.311048.
[47] L. B. R. Oliveira, F. S. Osório, and E. Y. Nakagawa, “An investigation into the development of service-oriented robotic systems,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC ’13*, Coimbra, Portugal, 2013, p. 223. doi: https://doi.org/10.1145/2480362.2480410.
[48] E. Freeman, E. Robson, B. Bates, and K. Sierra, *Head first design patterns*. O’Reilly Media, Inc., 2008.
[49] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design Patterns: Abstraction and Reuse of Object-Oriented Design,” in *ECOOP’ 93 — Object-Oriented Programming*, vol. 707, O. M. Nierstrasz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 406–431. doi: https://doi.org/10.1007/3-540-47910-4_21.
[50] R. E. Johnson, “Frameworks = (components + patterns),” *Commun. ACM*, vol. 40, no. 10, pp. 39–42, Oct. 1997, doi: https://doi.org/10.1145/262793.262799.
[51] R. A. De Paez, “Un Acercamiento a La Reutilización En Ingeniería De Software,” *EAFIT Mag.*, vol. 35, no. 114, pp. 45–63, 2012.
[52] J. D. Poole, “Model-driven architecture: Vision, standards and emerging technologies,” in *Workshop on Metamodeling and Adaptive Object Models, ECOOP*, 2001, vol. 50.
[53] W. Adi and K. Sekiyama, “A component-based framework for molecular robotic development as smart drug system,” in *2015 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, Nagoya, Japan, Nov. 2015, pp. 1–5. doi: https://doi.org/10.1109/MHS.2015.7438319.
[54] D. Brugali and P. Scandurra, “Component-based robotic engineering (Part I) [Tutorial],” *IEEE Robot. Autom. Mag.*, vol. 16, no. 4, pp. 84–96, Dec. 2009, doi: https://doi.org/10.1109/MRA.2009.934837.
[55] M. A. S. y J. L. S. y J. Z. Cortés, “Arquitectura orientada a servicios en el contexto de la arquitectura empresarial,” *Av. En Sist. E Informática*, vol. 7, no. 2, pp. 74–88, 2010.
[56] A. Ahmad and M. A. Babar, “Software architectures for robotic systems: A systematic mapping study,” *J. Syst. Softw.*, vol. 122, pp. 16–39, Dec. 2016, doi: https://doi.org/10.1016/j.jss.2016.08.039.

[57] P. Clements and L. Northrop, “Software product lines: Patterns and practice,” *Boston MA EUA Addison Wesley Longman Publ. Co*, 2001.
[58] P. Runeson and E. Engström, “Regression Testing in Software Product Line Engineering,” in *Advances in Computers*, vol. 86, Elsevier, 2012, pp. 223–263. doi: https://doi.org/10.1016/B978-0-12-396535-6.00007-7.
[59] L. M. Northrop, “SEI’s software product line tenets,” *IEEE Softw.*, vol. 19, no. 4, pp. 32–40, Jul. 2002, doi: https://doi.org/10.1109/MS.2002.1020285.
[60] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. doi: https://doi.org/10.1007/3-540-28901-1.
[61] M. Mendonca, M. Branco, and D. Cowan, “S.P.L.O.T.: software product lines online tools,” in *Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications - OOPSLA ’09*, Orlando, Florida, USA, 2009, p. 761. doi: https://doi.org/10.1145/1639950.1640002.
[62] E. J. M. Riascos, L. Rincón, and A. Tecnológico, “Modelo de características de una línea de productos de software para aplicaciones que usan dispositivos de Reconocimiento Biométrico,” *Reutil. Var. EN Ind. Softw. EN Colomb.*, p. 68, 2014.
[63] Arduino Company, “Arduino,” *Arduino*, Sep. 23, 2020. https://www.arduino.cc/en/Guide/Introduction
[64] Ó. T. Artero, *ARDUINO. Curso práctico de formación*. RC libros, 2013.
[65] K. Ashton and others, “That ‘internet of things’ thing,” *RFID J.*, vol. 22, no. 7, pp. 97–114, 2009.
[66] “Arduino Uno Rev3,” *Arduino Official Store*. http://store.arduino.cc/products/arduino-uno-rev3 (accessed Dec. 14, 2021).
[67] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, “Industry 4.0,” *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, Aug. 2014, doi: https://doi.org/10.1007/s12599-014-0334-4.
[68] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, “A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT),” in *2015 Internet Technologies and Applications (ITA)*, Wrexham, United Kingdom, Sep. 2015, pp. 219–224. doi: https://doi.org/10.1109/ITechA.2015.7317398.
[69] P. Simoens, M. Dragone, and A. Saffiotti, “The Internet of Robotic Things: A review of concept, added value and applications,” *Int. J. Adv. Robot. Syst.*, vol. 15, no. 1, p. 1729881418759424, 2018, doi: https://doi.org/10.1177/1729881418759424.
[70] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering – A systematic literature review,” *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, Jan. 2009, doi: https://doi.org/10.1016/j.infsof.2008.09.009.
[71] A. M. Soaita, B. Serin, and J. Preece, “A methodological quest for systematic literature mapping,” *Int. J. Hous. Policy*, vol. 20, no. 3, pp. 320–343, 2020, doi: https://doi.org/10.1080/19491247.2019.1649040.
[72] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic Mapping Studies in Software Engineering,” Jun. 2008. doi: 10.14236/ewic/EASE2008.8.
[73] F. C. Souza, A. Santos, S. Andrade, R. Durelli, V. Durelli, and R. Oliveira, “Automating Search Strings for Secondary Studies,” in *Information Technology - New Generations*, vol. 558, S. Latifi, Ed. Cham: Springer International Publishing, 2018, pp. 839–848. doi: https://doi.org/10.1007/978-3-319-54978-1_104.
[74] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. doi: https://doi.org/10.1007/978-3-642-29044-2.
[75] N. I. R. Yassin, S. Omran, E. M. F. El Houby, and H. Allam, “Machine learning techniques for breast cancer computer aided diagnosis using different image modalities: A systematic review,” *Comput. Methods Programs Biomed.*, vol. 156, pp. 25–45, Mar. 2018, doi: https://doi.org/10.1016/j.cmpb.2017.12.012.
[76] L. Marchezan, G. Bolfe, E. Rodrigues, M. Bernardino, and F. P. Basso, “Thoth: A Web-based Tool to Support Systematic Reviews,” in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Porto de Galinhas, Recife, Brazil, Sep. 2019, pp. 1–6. doi: 10.1109/ESEM.2019.8870160.

[77] T. Degueule, B. Combemale, A. Blouin, O. Barais, and J.-M. Jézéquel, “Safe model polymorphism for flexible modeling,” *Comput. Lang. Syst. Struct.*, vol. 49, pp. 176–195, Sep. 2017, doi: https://doi.org/10.1016/j.cl.2016.09.001.

[78] G. Doukas and K. Thramboulidis, “A Real-Time-Linux-Based Framework for Model-Driven Engineering in Control and Automation,” *IEEE Trans. Ind. Electron.*, vol. 58, no. 3, pp. 914–924, Mar. 2011, doi: https://doi.org/10.1109/TIE.2009.2029584.

[79] P. Bhavsar, S. H. Patel, and T. M. Sobh, “Hybrid Robot-as-a-Service (RaaS) Platform (Using MQTT and CoAP),” Atlanta, GA, USA, Jul. 2019, pp. 974–979. doi: https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00171.

[80] M. Haage *et al.*, “Teaching Assembly by Demonstration Using Advanced Human Robot Interaction and a Knowledge Integration Framework,” *Procedia Manuf.*, vol. 11, pp. 164–173, 2017, doi: https://doi.org/10.1016/j.promfg.2017.07.221.

[81] H. Wei *et al.*, “RT-ROS: A real-time ROS architecture on multi-core processors,” *Future Gener. Comput. Syst.*, vol. 56, pp. 171–178, Mar. 2016, doi: https://doi.org/10.1016/j.future.2015.05.008.

[82] G. Lunghi, R. Marin, M. Di Castro, A. Masi, and P. J. Sanz, “Multimodal Human-Robot Interface for Accessible Remote Robotic Interventions in Hazardous Environments,” *IEEE Access*, vol. 7, pp. 127290–127319, 2019, doi: https://doi.org/10.1109/ACCESS.2019.2939493.

[83] E. Estévez, A. S. García, J. G. García, and J. G. Ortega, “ART2ool: a model-driven framework to generate target code for robot handling tasks,” *Int. J. Adv. Manuf. Technol.*, vol. 97, no. 1–4, pp. 1195–1207, Jul. 2018, doi: https://doi.org/10.1007/s00170-018-1976-z.

[84] E. Estévez, A. Sánchez-García, J. Gámez-García, J. Gómez-Ortega, and S. Satorres-Martínez, “A novel model-driven approach to support development cycle of robotic systems,” *Int. J. Adv. Manuf. Technol.*, vol. 82, no. 1–4, pp. 737–751, Jan. 2016, doi: 10.1007/s00170-015-7396-4.

[85] D. Brugali and N. Hochgeschwender, “Software product line engineering for robotic perception systems,” *Int. J. Semantic Comput.*, vol. 12, no. 01, pp. 89–107, 2018, doi: https://doi.org/10.1142/S1793351X18400056.

[86] N. Arne, H. Nico, W. Dennis, and W. Sebastian, “A survey on domain-specific modeling and languages in robotics,” *J. Softw. Eng. Robot.*, vol. 7, no. 1, pp. 75–99, 2016.

[87] H. Bruyninckx, M. Klotzbücher, N. Hochgeschwender, G. Kraetzschmar, L. Gherardi, and D. Brugali, “The BRICS component model: a model-based development paradigm for complex robotics software systems,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC ’13*, Coimbra, Portugal, 2013, p. 1758. doi: https://doi.org/10.1145/2480362.2480693.

[88] A. Bubeck, B. Maidel, and F. G. Lopez, “Model Driven Engineering for the Implementation of User Roles in Industrial Service Robot Applications,” *Procedia Technol.*, vol. 15, pp. 605–612, 2014, doi: https://doi.org/10.1016/j.protcy.2014.09.021.

[89] H. Wei, X. Duan, S. Li, G. Tong, and T. Wang, “A component-based design framework for robot software architecture,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, Oct. 2009, pp. 3429–3434. doi: https://doi.org/10.1109/IROS.2009.5354161.

[90] J. E. Beck, J. M. Reagin, T. E. Sweeny, R. L. Anderson, and T. D. Garner, “Applying a component-based software architecture to robotic workcell applications,” *IEEE Trans. Robot. Autom.*, vol. 16, no. 3, pp. 207–217, Jun. 2000, doi: https://doi.org/10.1109/70.850639.

[91] M. Rudorfer, J. Guhl, P. Hoffmann, and J. Kruger, “Holo Pick’n’Place,” 2018, vol. 2018-September, pp. 1219–1222. doi: https://doi.org/10.1109/ETFA.2018.8502527.

[92] Y. Chen, Z. Du, and M. García-Acosta, “Robot as a Service in Cloud Computing,” in *2010 Fifth IEEE International Symposium on Service Oriented System Engineering*, Nanjing, China, Jun. 2010, pp. 151–158. doi: https://doi.org/10.1109/SOSE.2010.44.

[93] S. M. Salman, V. Struhar, A. V. Papadopoulos, M. Behnam, and T. Nolte, “Fogification of Industrial Robotic Systems: Research Challenges,” in *Proceedings of the Workshop on Fog Computing and the IoT*, New York, NY, USA, 2019, pp. 41–45. doi: https://doi.org/10.1145/3313150.3313225.

[94] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, “Cloud Robotics: Current Status and Open Issues,” *IEEE Access*, pp. 1–1, 2016, doi: https://doi.org/10.1109/ACCESS.2016.2574979.
[95] C. Tibermacine, S. Sadou, M. T. Ton That, and C. Dony, “Software architecture constraint reuse-by-composition,” *Future Gener. Comput. Syst.*, vol. 61, pp. 37–53, Aug. 2016, doi: 10.1016/j.future.2016.02.006.
[96] S. Trapani and M. Indri, “Task modeling for task-oriented robot programming,” in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Limassol, Sep. 2017, pp. 1–8. doi: 10.1109/ETFA.2017.8247650.
[97] J. Backhaus and G. Reinhart, “Digital description of products, processes and resources for task-oriented programming of assembly systems,” *J. Intell. Manuf.*, vol. 28, no. 8, pp. 1787–1800, Dec. 2017, doi: 10.1007/s10845-015-1063-3.
[98] K. R. Guerin, S. D. Riedel, J. Bohren, and G. D. Hager, “Adjutant: A framework for flexible human-machine collaborative systems,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, Sep. 2014, pp. 1392–1399. doi: 10.1109/IROS.2014.6942739.
[99] M. Wenger, W. Eisenmenger, G. Neugschwandtner, B. Schneider, and A. Zoitl, “A model-based engineering tool for ROS component compositioning, configuration and generation of deployment information,” in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Germany, Sep. 2016, pp. 1–8. doi: 10.1109/ETFA.2016.7733559.
[100] A. Lotz *et al.*, “Combining robotics component-based model-driven development with a model-based performance analysis,” in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, San Francisco, CA, USA, Dec. 2016, pp. 170–176. doi: 10.1109/SIMPAR.2016.7862392.
[101] L. H. Yoong, Z. E. Bhatti, and P. S. Roop, “Combining iec 61499 model-based design with component-based architecture for robotics,” in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, 2012, pp. 349–360. doi: 10.1007/978-3-642-34327-8_32.
[102] N. Rastogi, P. Dutta, V. Krishna, and K. K. Gotewal, “Implementation of an OROCOS based Real-Time Equipment Controller for Remote Maintenance of Tokamaks,” in *Proceedings of the Advances in Robotics on - AIR ’17*, New Delhi, India, 2017, pp. 1–6. doi: 10.1145/3132446.3134900.
[103] “ROS.org | Powering the world’s robots,” *ROS.org | Powering the world’s robots*, 2020. http://www.ros.org/ (accessed May 18, 2020).
[104] S. Zug, M. Schulze, A. Dietrich, and J. Kaiser, “Programming abstractions and middleware for building control systems as networks of smart sensors and actuators,” in *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, Bilbao, Sep. 2010, pp. 1–8. doi: 10.1109/ETFA.2010.5641341.
[105] Y. Sun, J. Gray, K. Bulheller, and N. von Baillou, “A Model-Driven Approach to Support Engineering Changes in Industrial Robotics Software,” in *Model Driven Engineering Languages and Systems*, vol. 7590, R. B. France, J. Kazmeier, R. Breu, and C. Atkinson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 368–382. doi: 10.1007/978-3-642-33666-9_24.
[106] K. Lee and T. Kim, “UMIICA: A Model-Driven Integrated Development Environment for Industrial Control Applications,” *IEEE Access*, vol. 6, pp. 43290–43301, 2018, doi: 10.1109/ACCESS.2018.2862944.
[107] M. Y. Jung and P. Kazanzides, “An architectural approach to safety of component-based robotic systems,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016, pp. 3360–3366. doi: 10.1109/ICRA.2016.7487511.
[108] L. Gherardi, D. Hunziker, and G. Mohanarajah, “A Software Product Line Approach for Configuring Cloud Robotics Applications,” in *2014 IEEE 7th International Conference on Cloud Computing*, Anchorage, AK, USA, Jun. 2014, pp. 745–752. doi: 10.1109/CLOUD.2014.104.
[109] T. Heikkila, T. Dobrowiecki, and L. Dalgaard, “Dealing with configurability in robot systems,” in *2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and*

Applications (MESA), Auckland, New Zealand, Aug. 2016, pp. 1–7. doi: 10.1109/MESA.2016.7587120.
[110] R. T. Geraldi, S. Reinehr, and A. Malucelli, “Software product line applied to the internet of things: A systematic literature review,” *Inf. Softw. Technol.*, vol. 124, p. 106293, Aug. 2020, doi: 10.1016/j.infsof.2020.106293.
[111] S. Bonfanti, M. Carissoni, A. Gargantini, and A. Mashkoor, “Asm2C++: A Tool for Code Generation from Abstract State Machines to Arduino,” in *NASA Formal Methods*, vol. 10227, C. Barrett, M. Davies, and T. Kahsai, Eds. Cham: Springer International Publishing, 2017, pp. 295–301. doi: 10.1007/978-3-319-57288-8_21.
[112] A. Ataide, J. P. Barros, I. S. Brito, and L. Gomes, “Towards automatic code generation for distributed cyber-physical systems: A first prototype for Arduino boards,” in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Limassol, Sep. 2017, pp. 1–4. doi: 10.1109/ETFA.2017.8247737.
[113] B. Hu, H. Wang, P. Zhang, B. Ding, and H. Che, “Cloudroid: A Cloud Framework for Transparent and QoS-Aware Robotic Computation Outsourcing,” in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honolulu, CA, USA, Jun. 2017, pp. 114–121. doi: 10.1109/CLOUD.2017.23.
[114] I. Maurtua *et al.*, “Natural multimodal communication for human–robot collaboration,” *Int. J. Adv. Robot. Syst.*, vol. 14, no. 4, p. 172988141771604, Jul. 2017, doi: 10.1177/1729881417716043.
[115] D. Weyns, “Engineering Self-Adaptive Software Systems – An Organized Tour,” in *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Trento, Sep. 2018, pp. 1–2. doi: 10.1109/FAS-W.2018.00012.
[116] P. Estefo, J. Simmonds, R. Robbes, and J. Fabry, “The Robot Operating System: Package reuse and community dynamics,” *J. Syst. Softw.*, vol. 151, pp. 226–242, May 2019, doi: 10.1016/j.jss.2019.02.024.
[117] S. Hallsteinsen, M. Hinchey, Sooyong Park, and K. Schmid, “Dynamic Software Product Lines,” *Computer*, vol. 41, no. 4, pp. 93–95, Apr. 2008, doi: 10.1109/MC.2008.123.
[118] N. Iscoe, G. B. Williams, and G. Arango, “Domain modeling for software engineering,” in *[1991 Proceedings] 13th International Conference on Software Engineering*, Austin, TX, USA, 1991, pp. 340–343. doi: 10.1109/ICSE.1991.130660.
[119] M. Allamanis and C. Sutton, “Mining idioms from source code,” in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*, Hong Kong, China, 2014, pp. 472–483. doi: 10.1145/2635868.2635901.
[120] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, “code2vec: learning distributed representations of code,” *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 1–29, Jan. 2019, doi: 10.1145/3290353.
[121] A. V. Aho, R. Sethi, and J. D. Ullman, *Compiladores: principios, técnicas y herramientas*. Pearson Educación, 1998.
[122] R. Mak, *Writing Compilers and Interpreters: A Software Engineering Approach*. Wiley, 2011. [Online]. Available: https://books.google.es/books?id=EUs94gRZHEUC
[123] Arduino, “Guía de Referencia de Arduino,” *Arduino*, Nov. 05, 2020. https://www.arduino.cc/reference/es/ (accessed Nov. 05, 2020).
[124] Yan Zhang, Xinyu Gao, Ce Bian, Ding Ma, and Baojiang Cui, “Homologous detection based on text, Token and abstract syntax tree comparison,” in *2010 IEEE International Conference on Information Theory and Information Security*, Beijing, China, Dec. 2010, pp. 70–75. doi: 10.1109/ICITIS.2010.5689624.
[125] Y. Gou, H. K. Dam, and A. Ghose, “Towards Semantic Merging of Versions of BDI Agent Systems,” in *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, Berlin, Heidelberg, 2013, pp. 437–444. doi: https://doi.org/10.1007/978-3-642-44927-7_32.
[126] V. Fernández, A. Pazos, J. Saborido, and M. Seco, “Arduino and Nagios integration for monitoring,” *J. Phys. Conf. Ser.*, vol. 513, no. 6, p. 062015, Jun. 2014, doi: 10.1088/1742-6596/513/6/062015.
[127] K. Garcés, C. Parra, H. Arboleda, A. Yie, and R. Casallas, “Variability Management in a Model-Driven Software Product Line,” *Av. En Sist. E Informática*, vol. 4, no. 2, pp. 3–12, 2007.

[128] D. Gleeson, S. Bjorkenstam, R. Bohlin, J. S. Carlson, and B. Lennartson, “Optimizing robot trajectories for automatic robot code generation,” in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, Gothenburg, Sweden, Aug. 2015, pp. 495–500. doi: 10.1109/CoASE.2015.7294128.
[129] Kisron, B. S. B. Dewantara, and F. Ardilla, “Self Monitoring, Failure-Detection and Decision-Making System to Support E-TrashBot (EEPIS Trash Bin Robot) Operations: Preliminary Report,” in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Kuta, Jul. 2018, pp. 1–6. doi: 10.1109/ICITEED.2018.8534932.
[130] S. S. Mohammed and D. Devaraj, “Design, Simulation and Analysis of Microcontroller based DC-DC Boost Converter using Proteus Design Suite,” in *Proc. of Int. Conf. on Advances in Electrical & Electronics, AETAEE*, 2013, pp. 599–606.
[131] D. Brugali and M. Valota, “Software Variability Composition and Abstraction in Robot Control Systems,” in *Computational Science and Its Applications – ICCSA 2016*, Cham, 2016, pp. 358–373.
[132] M. C. C. Ojeda, J. A. H. Alegría, F. J. Á. Rodriguez, and P. H. R. Melenje, “A Collaborative Method for a Tangible Software Product Line Scoping,” in *2018 ICAI Workshops (ICAIW)*, Bogotá, Colombia, 2018, pp. 1–6. doi: 10.1109/ICAIW.2018.8554999.
[133] H. Gomaa, “Designing Software Product Lines with UML 2.0: From Use Cases to Pattern-Based Software Architectures,” in *10th International Software Product Line Conference*, Los Alamitos, CA, USA, Aug. 2006, pp. 218–218. doi: 10.1109/SPLINE.2006.1691600.
[134] C. Kastner *et al.*, “FeatureIDE: A tool framework for feature-oriented software development,” in *2009 IEEE 31st International Conference on Software Engineering*, Vancouver, BC, Canada, 2009, pp. 611–614. doi: 10.1109/ICSE.2009.5070568.
[135] E. Bagheri and D. Gasevic, “Assessing the maintainability of software product line feature models using structural metrics,” *Softw. Qual. J.*, vol. 19, no. 3, pp. 579–612, Sep. 2011, doi: 10.1007/s11219-010-9127-2.
[136] D. Mellado, E. Fernandez-Medina, and M. Piattini, “Applying a Security Domain Requirements Engineering Process for Software Product Lines,” *IEEE Lat. Am. Trans.*, vol. 6, no. 3, pp. 298–305, Jul. 2008, doi: 10.1109/TLA.2008.4653861.
[137] D. Radošević and I. Magdalenić, “Python implementation of source code generator based on dynamic frames,” in *2011 Proceedings of the 34th International Convention MIPRO*, 2011, pp. 969–974.
[138] G. Botterweck and A. Pleuss, “Evolution of Software Product Lines,” in *Evolving Software Systems*, T. Mens, A. Serebrenik, and A. Cleve, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 265–295. doi: 10.1007/978-3-642-45398-4_9.
[139] K. Schmid, “Scoping Software Product Lines,” in *Software Product Lines*, P. Donohoe, Ed. Boston, MA: Springer US, 2000, pp. 513–532. doi: 10.1007/978-1-4615-4339-8_27.
[140] A. F. Solis Pino, L. M. Vargas-Ordoñez, and C. A. Collazos, “Model for Writing Scientific Articles Remotely Through Collaborative Tasks,” *TecnoLógicas*, vol. 24, no. 50, p. e1701, Jan. 2021, doi: 10.22430/22565337.1701.
[141] M. Ciotti, M. Ciccozzi, A. Terrinoni, W.-C. Jiang, C.-B. Wang, and S. Bernardini, “The COVID-19 pandemic,” *Crit. Rev. Clin. Lab. Sci.*, vol. 57, no. 6, pp. 365–388, Aug. 2020, doi: 10.1080/10408363.2020.1783198.
[142] P. Buonocunto, A. Biondi, and P. Lorefice, “Real-time multitasking in Arduino,” in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, Pisa, Jun. 2014, pp. 1–4. doi: 10.1109/SIES.2014.7087331.
[143] D. Rakita, B. Mutlu, and M. Gleicher, “An analysis of RelaxedIK: an optimization-based framework for generating accurate and feasible robot arm motions,” *Auton. Robots*, vol. 44, no. 7, pp. 1341–1358, Sep. 2020, doi: 10.1007/s10514-020-09918-9.
[144] E. Sariyildiz, E. Cakiray, and H. Temeltas, “A Comparative Study of Three Inverse Kinematic Methods of Serial Industrial Robot Manipulators in the Screw Theory Framework,” *Int. J. Adv. Robot. Syst.*, vol. 8, no. 5, p. 64, 2011, doi: 10.5772/45696.
[145] M. A. Abdelhady, D. Dresscher, and J. F. Broenink, “Reuse-oriented SLAM Framework using Software Product Lines,” in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, 2020, pp. 187–190. doi: 10.1109/IRC.2020.00037.

[146] B. Meyer, *Object-oriented software construction*, vol. 2. Prentice hall Englewood Cliffs, 1997.
[147] F. de Almeida Florencio, E. D. Moreno, H. Teixeira Macedo, R. J. P. de Britto Salgueiro, F. Barreto do Nascimento, and F. A. Oliveira Santos, “Intrusion Detection via MLP Neural Network Using an Arduino Embedded System,” in *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, 2018, pp. 190–195. doi: 10.1109/SBESC.2018.00036.
[148] G. Barbon, M. Margolis, F. Palumbo, F. Raimondi, and N. Weldin, “Taking Arduino to the Internet of Things: The ASIP programming model,” *Comput. Commun.*, vol. 89–90, pp. 128–140, 2016, doi: https://doi.org/10.1016/j.comcom.2016.03.016.
[149] R. Moran, M. Teragni, and G. Zabala, “Simplifying concurrency and monitoring on Arduino for Internet of Things,” 2020.
[150] N. Ádám, T. Gergely, M. Hulič, J. Hurtuk, and B. Madoš, “Proto-cluster: A Multi-device Approach to Parallel Computing,” in *2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2019, pp. 176–180. doi: 10.1109/SAMI.2019.8782772.
[151] Y. N. Krishnan, C. N. Bhagwat, and A. P. Utpat, “Fog computing — Network based cloud computing,” in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, 2015, pp. 250–251. doi: 10.1109/ECS.2015.7124902.
[152] M. I. Ullah and G. Ruhe, “Towards Comprehensive Release Planning for Software Product Lines,” in *2006 International Workshop on Software Product Management (IWSPM’06 - RE’06 Workshop)*, 2006, pp. 51–56. doi: 10.1109/IWSPM.2006.9.
[153] G. Chastek and J. McGregor, “Guidelines for Developing a Product Line Production Plan,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, CMU/SEI-2002-TR-006, 2002. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6067
[154] P. C. Clements, L. G. Jones, L. M. Northrop, and J. D. McGregor, “Project management in a software product line organization,” *IEEE Softw.*, vol. 22, no. 5, pp. 54–62, 2005, doi: 10.1109/MS.2005.133.
[155] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, “Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges,” *IEEE Access*, vol. 6, pp. 6505–6519, 2018, doi: 10.1109/ACCESS.2017.2783682.
[156] R. Votrubec and F. Koblasa, “Control System of Vehicle for Smart Factory Model with Principles of Industry 4.0,” in *Proceedings of the 30th DAAAM International Symposium*, Vienna, Austria, 2019, pp. 0261–0267. doi: 10.2507/30th.daaam.proceedings.034.
[157] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, Jun. 1994, doi: 10.1109/32.295895.
[158] S. El-Sharkawy, N. Yamagishi-Eichler, and K. Schmid, “Metrics for analyzing variability and its implementation in software product lines: A systematic literature review,” *Inf. Softw. Technol.*, vol. 106, pp. 1–30, Feb. 2019, doi: 10.1016/j.infsof.2018.08.015.
[159] W. Frakes and C. Terry, “Software Reuse and Reusability Metrics and Models,” Virginia Polytechnic Institute & State University, Blacksburg, 1995. [Online]. Available: http://hdl.handle.net/10919/19897
[160] W. O. López González, “El estudio de casos: una vertiente para la investigación educativa,” *Educere*, 2013, [Online]. Available: https://www.redalyc.org/articulo.oa?id=35630150004
[161] Y. Umeda, S. Kondoh, T. Sugino, and H. Yoshikawa, “Analysis of Reusability using ‘Marginal Reuse Rate,’” *CIRP Ann.*, vol. 55, no. 1, pp. 41–44, 2006, doi: 10.1016/S0007-8506(07)60362-X.
[162] M. Jørgensen, “A review of studies on expert estimation of software development effort,” *J. Syst. Softw.*, vol. 70, no. 1–2, pp. 37–60, Feb. 2004, doi: 10.1016/S0164-1212(02)00156-5.
[163] R. Abilio, G. Vale, E. Figueiredo, and H. Costa, “Metrics for Feature-Oriented Programming,” in *2016 IEEE/ACM 7th International Workshop on Emerging Trends in Software Metrics (WETSoM)*, 2016, pp. 36–42. doi: 10.1109/WETSoM.2016.014.
[164] N. Bevan, J. Carter, and S. Harker, “ISO 9241-11 Revised: What Have We Learnt About Usability Since 1998?,” in *Human-Computer Interaction: Design and Evaluation*, vol. 9169, M. Kurosu, Ed. Cham: Springer International Publishing, 2015, pp. 143–151. doi: 10.1007/978-3-319-20901-2_13.

[165] J. Sauro and J. R. Lewis, “Standardized usability questionnaires,” in *Quantifying the User Experience*, Elsevier, 2016, pp. 185–248. doi: 10.1016/B978-0-12-802308-2.00008-4.
[166] “HowToMechatronics - Learn Arduino, electronics, mechatronics and more.,” *HowToMechatronics*. https://howtomechatronics.com/ (accessed Dec. 14, 2021).
[167] B. Leonard, *GAO Cost estimating and assessment guide: Best practices for developing and managing capital program costs*. Diane Publishing, 2009.
[168] C. X. R. Chandi and C. F. V. Orellana, “Metodología para evaluación de usabilidad del entorno de desarrollo integrado de Arduino,” *SATHIRI*, vol. 13, no. 1, pp. 214–226, 2018.
[169] A. B. Pratomo and R. S. Perdana, “Arduviz, a visual programming IDE for arduino,” in *2017 International Conference on Data and Software Engineering (ICoDSE)*, Palembang, Nov. 2017, pp. 1–6. doi: 10.1109/ICODSE.2017.8285871.
[170] D. Li and H. Quichen, “Ardublock: A Graphical Programming Language for Arduino.” On, 2020.
[171] J. McGregor, “Arcade game maker pedagogical product line: Concept of operations,” *Version*, vol. 2, p. 2005, 2005.
[172] F. J. van der Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer Berlin Heidelberg, 2007. [Online]. Available: https://books.google.es/books?id=7sU8TEyyHCsC
[173] S. P. Gregg, R. Scharadin, E. LeGore, and P. Clements, “Lessons from AEGIS: organizational and governance aspects of a major product line in a multi-program environment,” in *Proceedings of the 18th International Software Product Line Conference - Volume 1*, Florence Italy, Sep. 2014, pp. 264–273. doi: 10.1145/2648511.2648541.
[174] J. Åkesson, S. Nilsson, J. Krüger, and T. Berger, “Migrating the Android Apo-Games into an Annotation-Based Software Product Line,” in *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A*, Paris France, Sep. 2019, pp. 103–107. doi: 10.1145/3336294.3342362.
[175] A. Buccella, A. Cechich, M. Pol′la, M. Arias, M. del Socorro Doldan, and E. Morsan, “Marine ecology service reuse through taxonomy-oriented SPL development,” *Comput. Geosci.*, vol. 73, pp. 108–121, Dec. 2014, doi: 10.1016/j.cageo.2014.09.004.
[176] P. Toft, D. Coleman, and J. Ohta, “A Cooperative Model for Cross-Divisional Product Development for a Software Product Line,” in *Software Product Lines*, P. Donohoe, Ed. Boston, MA: Springer US, 2000, pp. 111–132. doi: 10.1007/978-1-4615-4339-8_6.
[177] A. Tang, W. Couwenberg, E. Scheppink, N. A. de Burgh, S. Deelstra, and H. van Vliet, “SPL migration tensions: an industry experience,” in *Proceedings of the 2010 Workshop on Knowledge-Oriented Product Line Engineering - KOPLE ’10*, Reno, Nevada, 2010, pp. 1–6. doi: 10.1145/1964138.1964141.

Apéndices

Apéndice A. Tabla de artículos revisados en el mapeo sistemático de literatura

Artículo	Técnicas de reutilización de software utilizada	Año de creación del software	Tipo de contribución al área de reutilización	Contexto de aplicación
Effective Engineering of Multi-Robot Software Applications	Component-based	2009	Arquitectura	Académico
Model driven engineering for the implementation of user roles in industrial service robot applications	Model-driven	2014	Arquitectura	Académico
Hiding real-time: A new approach for the software development of industrial robots	OOR	2019	Arquitectura	Académico
Knowledge Based Hierarchical Decomposition of Industry 4.0 Robotic Automation Tasks	Domain-specific	2018	framework	Ambos
Modeling and reusing robotic software architectures: The HyperFlex toolchain	Model-driven	2014	Arquitectura	Académico
A system integration approach for service-oriented robotics	Service-oriented	2013	framework	Académico
BRIDE - A toolchain for framework-independent development of industrial service robot applications	Model-driven	2014	framework	Ambos
Towards easier human-robot interaction	No especifica	2015	Arquitectura	Industrial
Programming abstractions and middleware for building control systems as networks of smart sensors and actuators	Component-based	2010	Middleware	Académico
A model based engineering tool for ROS component compositioning, configuration and generation of deployment information	Model-driven	2016	framework	Académico
Rubik's cube as a benchmark validating MRROC++ as an implementation tool for service robot control systems.	OOR	2007	framework	Académico
ON THE USE OF SERVICE ORIENTED SOFTWARE PLATFORMS FOR INDUSTRIAL ROBOTIC CELLS	Service-oriented	2007	Middleware	Académico
Telerobotic systems design based on real-time CORBA	component-based	2005	framework	Académico
Model-driven online correction welding robot automatic programming system	Model-driven	2019	Generador de código	Ambos
Intuitive instruction of industrial robots: Semantic process descriptions for small lot production	OOR	2016	Método o modelo	Ambos
Robot as a Service in Cloud Computing	Service-oriented	2010	Arquitectura	Ambos
The BRICS Component Model: A Model-Based Development Paradigm for Complex Robotics Software Systems	Component-based	2013	framework	Académico
BRICS - Best practice in robotics	No especifica	2010	framework	Ambos
Engineering Robotics Software Architectures with Exchangeable Model Transformations	Model-driven	2017	Arquitectura	Académico
ART2ool: a model-driven framework to generate target code for robot handling tasks	Model-driven	2018	Generador de código	ambos
A novel model-driven approach to support development cycle of robotic systems	Model-driven	2016	Generador de código	Académico
A Model-Driven Engineering Approach for {ROS} using Ontological Semantics	Model-driven	2016	Método o modelo	Académico
From AutomationML to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning	Model-driven	2016	Método o modelo	Ambos
The Robot Operating System: Package reuse and community dynamics	No especifica	2019	framework	Académico
Robotic task and system design for task directed robotics case study: deburring task	Task-Oriented	1994	Método o modelo	Académico
Simplified Programming of Re-usable Skills on a Safe Industrial Robot: Prototype and Evaluation	No especifica	2017	Graphical Programming	Académico

ROPSIM, A ROBOT OFF-LINE PROGRAMMING AND REAL-TIME SIMULATION SYSTEM INCLUDING DYNAMICS	Model-driven	1991	Arquitectura	Académico
Object-oriented and distributed approach for programming robotic manufacturing cells	OOR	2000	Método o modelo	Académico
MD-SIR: a methodology for developing sensor—guided industry robots	Component-based	2002	Método o modelo	Académico
Industrial robot programming and upnp services orchestration for the automation of factories	Service-oriented	2012	framework	Académico
Designing telerobotic systems as distributed CORBA-based applications	Component-based	2003	framework	Académico
Designing Dynamic and Collaborative Automation and Robotics Software Systems	Service-oriented	2017	framework	Académico
ControlShell component-based real-time programming system	Component-based	1995	Graphical Programming	Industrial
Applying a component-based software architecture to robotic workcell applications	Component-based	2000	Arquitectura	Ambos
A novel model-based approach to support development cycle of robotic arm applications	Model-driven	2014	framework	Académico
An Integrated Agent-based Software Architecture for Mobile and Manipulator Systems	Service-oriented	2007	Arquitectura	Académico
Adjutant: A framework for flexible human-machine collaborative systems	Model-driven	2014	framework	Académico
Adaptive Robot Design and Applications in Flexible Manufacturing Environments	Component-based	2009	Middleware	Ambos
The design of general off-line graphical robot programming system	Model-driven	2016	framework	Industrial
Designing a multimodal human-robot interaction interface for an industrial robot	Model-driven	2015	Graphical Programming	Académico
Teaching Assembly by Demonstration Using Advanced Human Robot Interaction and a Knowledge Integration Framework	Component-based	2017	framework	Ambos
Development of industrial robots based applications by SOA universal plug-and-play architecture [Splicación de la arquitectura orientada a servicios universal plug-and-play para facilitar la integración de robots industriales en líneas de producción]	Service-oriented	2012	Método o modelo	Académico
RTRobMultiAxisControl: A Framework for Real-Time Multiaxis and Multirobot Control	OOR	2019	Middleware	Académico
A formal model-based design method for robotic systems	Model-driven	2019	framework	Académico
An UML based approach for designing and coding automatically robotic arm platforms [Aproximación Basada en UML para el Diseño y Codificación Automática de Plataformas Robóticas Manipuladoras]	Model-driven	2017	Generador de codigo	Académico
A model-driven approach to support engineering changes in industrial robotics software	Domain-specific	2012	Generador de código	Industrial
A real-time-linux-based framework for model-driven engineering in control and automation	Model-driven	2011	framework	Ambos
Programming-by-demonstration in the coworker scenario for SMEs	Service-oriented	2009	Generador de codigo	Académico
Using digital pens to program welding tasks	No especifica	2007	Graphical Programming	Académico
Off-line robot programming framework	OOR	2005	framework	Académico
Reusable framework for Web-based teleoperation	Component-based	1998	framework	Académico
A holonic component-based approach to reconfigurable manufacturing control architecture	Component-based	2000	Arquitectura	Académico
Model-based Design of Time-triggered Real-time Embedded Systems for Digital Manufacturing	Model-driven	2015	framework	Ambos
RobotML for industrial robots: Design and simulation of manipulation scenarios	Model-driven	2016	framework	Ambos
A framework for the automatic programming of advanced production machinery	Task-Oriented	1994	Generador de código	Académico
Software Methodologies for the Engineering of Service-Oriented Industrial Automation: The Continuum Project	Service-oriented	2009	Método o modelo	Académico
The Robotics API: An object-oriented framework for modeling industrial robotics applications	OOR	2010	API	Académico
A component-based software architecture for control and simulation of robotic manipulators	Component-based	2013	framework	Académico

Experiments with service-oriented architectures for industrial robotic cells programming	Service-oriented	2009	framework	Ambos
Programming multirobot applications using the ThinkingCap-II Java framework	Component-based	2010	Arquitectura	Académico
Component-Based Modelling for Generating Robotic Arm Applications Running Under OROCOS Middleware	Component-based	2013	framework	Académico
Domain-Specific Modeling for Robotics: From Language Construction to Ready-made Controllers and End-user Applications	Domain-specific	2016	framework	Ambos
Modular Pick and Place Simulator Using ROS Framework	Component-based	2015	framework	Académico
The Java Fieldbus Control Framework-Object Oriented Control of Fieldbus Devices	OOR	2001	framework	Académico
Dealing with World-model-based Programs	No especifica	1985	Método o modelo	Académico
Projection-Based Augmented Reality Interface for Robot Grasping Tasks	Task-oriented	2019	Graphical Programming	Académico
Holo Pick'n'Place	Service-oriented	2018	Graphical Programming	Académico
A knowledge centric approach to conceptualizing robotic solutions	Model-driven	2019	Arquitectura	Ambos
Control of Nao Robot Arm using Myo Armband	No especifica	2019	Graphical Programming	Industrial
Integration of Computervision and Artificial Intelligence Subsystems with Robot Operating System Based Motion Planning for Industrial Robots	Component-based	2018	framework	Académico
Learning to Serve: an Experimental Study for a new Learning from Demonstrations Framework	No especifica	2019	framework	Académico
mROS: A Lightweight Runtime Environment for Robot Software Components onto Embedded Devices	Component-based	2019	framework	Ambos
Multimodal Human-Robot Interface for Accessible Remote Robotic Interventions in Hazardous Environments	Model-driven	2019	Graphical Programming	Industrial

Apéndice B. Código fuente del analizador lexicográfico.

```
import re
from collections import Counter

words_to_keep = {"for", "if", "goto", "return", "while", "else", "return", "switch", "continue",
"break","setup()","loop()","setup ", "loop "} #estructuras de control

words_to_keep3 = {"#define", "#include","Adafruit","unsigned int ", "unsigned long ", "byte ", "char ",
"float ", "int ","long ","word ","boolean ", "String ","const ","static ","volatile ", "PROGMEM
","sizeof", "double"} #tipos de variables

words_to_keep4 = {"digitalRead", "digitalWrite", "pinMode", "analogRead", "analogReference",
"analogWrite","analogReadResolution","analogWriteResolution","noTone ","pulseIn", "shiftIn","shiftOut",
"tone","pulseInLong","shiftIn", "shiftOut", "tone", "delay", "delayMicroseconds", "micros", "millis",
"abs", "random","randomSeed", "constrain","bit","bitClear","bitRead","bitSet","bitWrite","highByte"
,"lowByte","attachInterrupt", "detachInterrupt", "interrupts", "noInterrupts", "Serial.begin",
"Serial.available","Serial.availableForWrite","bit","random","randomSeed","Serial.print"} #Funciones

pattern = re.compile('|'.join(re.escape(word) for word in words_to_keep))
pattern3 = re.compile('|'.join(re.escape(word3) for word3 in words_to_keep3))
pattern4 = re.compile('|'.join(re.escape(word4) for word4 in words_to_keep4))

def word_count(filename):
    with open(filename, 'r') as f:
        words_found = pattern.findall(f.read())
        return Counter(words_found)

def word_count3(filename3):
    with open(filename3, 'r') as f3:
        words_found3 = pattern3.findall(f3.read())
        return Counter(words_found3)

def word_count4(filename4):
    with open(filename4, 'r') as f4:
        words_found4 = pattern4.findall(f4.read())
        return Counter(words_found4)

print("------------------------Estructuras de control-----------------------------\n")
for word, count in word_count('hola.txt').items():
    print(word, ":", count)
print("\n---------------------tipos de variables-----------------------------------\n")
for word3, count3 in word_count3('hola.txt').items():
    print(word3, ":", count3)
print("\n-------------------------Funciones---------------------------------------\n")
for word4, count4 in word_count4('hola.txt').items():
    print(word4, ":", count4)
print("\n\n")
```

Apéndice C. Código fuente del generado de código para la primera iteración de la SPL.

Disponible en el siguiente enlace: https://github.com/andbrs/L-nea-de-productos-de-software-para-Arduino.git

Apéndice D. Recursos experimentales de la prueba de concepto de la primera versión de IRArduino-SPL.

Disponible en el siguiente enlace:

https://drive.google.com/drive/u/3/folders/1TpyoLZGypPwL2-1o68apZApeSeRtUUtx

Apéndice E. Encuesta realizada para abstraer el dominio de los sistemas robóticos industriales.

Identificación de aspectos de software relevantes en la construcción de robots industriales con Arduino

LEER POR FAVOR:

La presente encuesta tiene como objetivo identificar cuáles son los aspectos de software relevantes en la construcción de un sistema robótico (SR) industrial con Arduino (comúnmente llamados brazos robóticos), para de esta forma aplicarlos a una estrategia de reutilización de software.

Ahora bien, puntualmente para esta investigación buscamos elementos de software relevantes (por ejemplo, cinemáticas, sensores y actuadores, más allá de que son dispositivos físicos estos necesitan de programación para controlarlos, entre otros) y no nos centramos en aspectos relevantes de hardware (por ejemplo, los engranajes no se les puede programar, por lo que para esta investigación no son tan relevantes).

CONTEXTO DE LA INVESTIGACIÓN:

Hola, ¿Qué tal?, Mi nombre es Andrés Solís y soy estudiante de la maestría en computación de la Universidad del Cauca, actualmente estoy trabajando en mi proyecto de grado que es una estrategia de reutilización de software para los SR industriales con Arduino (normalmente, utilizados en la academia). Esta idea surge porque en un mapeo sistemático de la literatura (https://polipapers.upv.es/index.php/RIAI/article/view/13335) se detectó que las estrategias de reutilización de software en el dominio de los robots industriales (en la industria) no han sido adoptadas adecuadamente, por lo que intentamos implementar una estrategia de reutilización en la academia para que así se naturalice su uso en las áreas profesionales cuando sea el caso. En este sentido, hemos realizado un primer acercamiento a la estrategia de reutilización (artículo en revisión) en el que hemos abstraído los principales componentes software de este tipo de robots, pero queremos validar estos componentes con personas que tengan conocimiento en la construcción de SR industriales con Arduino, por eso necesitamos tu ayuda.

Los siguientes aspectos relevantes de software fueron identificados en la primera interacción de la estrategia:

- Cinemáticas
- Sensores
- Actuadores
- Efector final
- Elementos de control de potencia o energía
- Interfaz de usuario
- Multitarea
- Grados de libertad

Entre otras.

Ahora, necesitamos tu ayuda, cuales aspectos crees que son correctos y cuales crees que hacen falta.

GLOSARIO DE TÉRMINOS:

Reutilización de software: se refiere al comportamiento y a las técnicas que garantizan que una parte o la totalidad de un programa informático existente se pueda emplear en la construcción de otro programa.

Sistema robótico industrial: También denominados brazos robóticos, entendemos que dentro de la robótica industrial existen otros tipos de robots (paralelos o SCARA), pero para esta encuesta solo consideramos los brazos robóticos independientemente de los grados de libertad que tengan.

Nota: Este es un ejercicio académico que pretende establecer un panorama sobre el software para los robots industriales.

*Obligatorio

1. ¿Cuál es su nombre? *

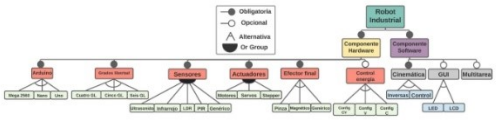
2. ¿Cómo considera sus conocimientos en la construcción de brazos robóticos? *

   Marca solo un óvalo.

   - Mucha experiencia (Más de 3 robots industriales)
   - Experiencia media (Menos de 3 robots industriales y más de 1)
   - Constructor ocasional (Un robot industrial)

3. Con respecto a la construcción de brazos robóticos ¿en cuántos ha participado? (solo números) *

Aquí se puede observar un modelo de características con los aspectos de software relevantes que se consideraron para la primera versión de la estrategia de reutilización. Este modelo nos permite ver cuales características siempre se van a requerir en los proyectos (obligatorias), y las opcionales y alternativas nos indican las características cuya inclusión depende de las necesidades especificas de algunos proyectos.



4. En su experiencia y opinión ¿Cuáles son los aspectos de software más relevantes en las construcción de SR industriales? (Puede seleccionar varios aspectos y sugerir otros en la opción otro) *

Selecciona todos los que correspondan.

- [ ] Control de potencia o energía del robot (Un conjunto de sensores y/o actuadores que se encargan de limitar la energía para evitar cortocircuitos)
- [ ] Botonera de control (Botón de apagado, parada de emergencia y stop)
- [ ] Grados de libertad
- [ ] Determinación de Cinemáticas (Directas e inversas) (Ecuaciones matemáticas que permiten calcular la posición de su actuador final a partir de valores específicos denominado)
- [ ] Establecimiento de una frecuencias de muestreo para el funcionamiento del robot
- [ ] Planificación de trayectorias
- [ ] Interfaz grafica de usuario (GUI) (Puede ser una LCD)
- [ ] Agregar una simulación de Multitarea
- [ ] Elección del tipo de placa Arduino

Otro: [ ] _____

5. ¿Cuáles son las tres mayores dificultades a nivel de programación que más desmotiva o frustra a los nuevos aprendices en la programación de robots industriales? *

Selecciona todos los que correspondan.

- [ ] Cinemáticas
- [ ] Inicialización
- [ ] Planificación de trayectorias
- [ ] Muestreo
- [ ] Confiabilidad de la información y las lógicas de los Sensores/Actuadores
- [ ] Sinergia correcta entre software y hardware del robot
- [ ] Manejo de multitareas
- [ ] Sincronización
- [ ] Grados de libertad

6. Qué sugerencias tendria para mejorar el modelo de características planteado (agregar/eliminar/modificar alguna o varias características)

7. Su conocimiento en la construcción de robots industriales con Arduino lo adquirió en la: *

Marca solo un óvalo.

- ◯ Academia
- ◯ Industria
- ◯ Autodidacta

8. ¿Estudias alguna carrera relacionada con la construcción de robots? *

Marca solo un óvalo.

- ◯ Sí
- ◯ No

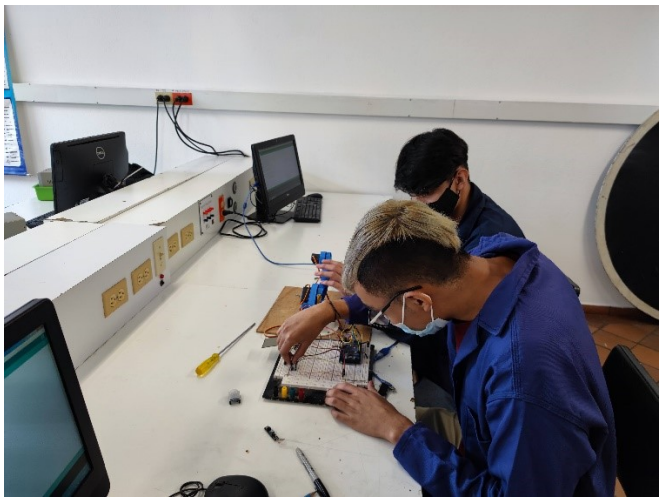
9. Sí estudias alguna carrera universitaria con respecto a la construcción de robots ¿Cuál es? y ¿Cuál es tu opinión sobre esta?

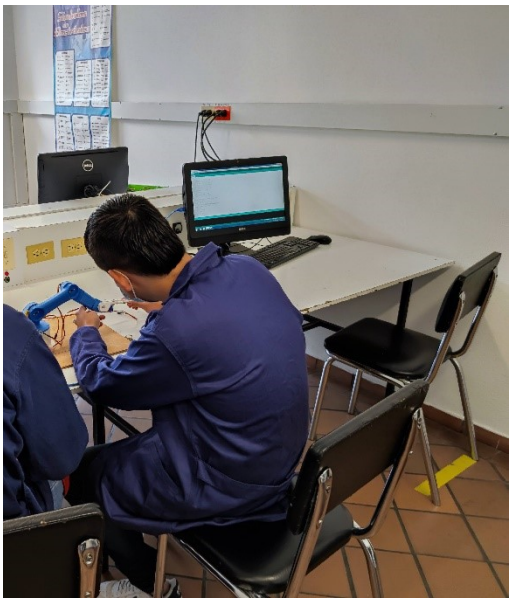
Apéndice F. Código fuente de la segunda iteración de la SPL para sistemas robóticos industriales en Arduino (IRArduino-SPL).

Disponible en el siguiente enlace:

https://github.com/andbrs/IRArduino-SPL.git

Apéndice G. Evidencias fotográficas de la ejecución del estudio de caso.





Apéndice H. Documento reporte de IRArduino-SPL.

IRArduino-SPL: Reporte

En este apéndice se expone un resumen y una idea general detrás de IRArduino-SPL y sus principales particularidades, basado en el reporte de la SPL pedagógica denominada AGM [171]. Además, se presenta la SPL propuesta como un enfoque sistemático para el desarrollo de software en robots industriales con Arduino basado en activos núcleo reutilizables detallando cada uno de los elementos esenciales en la estrategia de reutilización.

IRArduino-SPL: Principios rectores de la línea de productos

La estrategia SPL pretende mejorar la producción de software de una organización planificando y construyendo un conjunto de productos reutilizables como un grupo, en lugar de tratar cada producto por separado. Hay varios fundamentos que afectan al funcionamiento de la organización de la línea de productos [172].

La idea de solución detrás de IRArduino-SPL está basada en el desarrollo de una SPL que ayuda acelerar el proceso de desarrollo en robots industriales con Arduino, haciendo uso de la reutilización de software (activos núcleo) en un contexto específico. En esta línea, la construcción de elementos comunes o similares permite utilizar una base conjunta sobre la que construir productos de características parecidas, manteniendo particularidades propias o variables en este tipo de dispositivos electromecánicos (Figura 41).

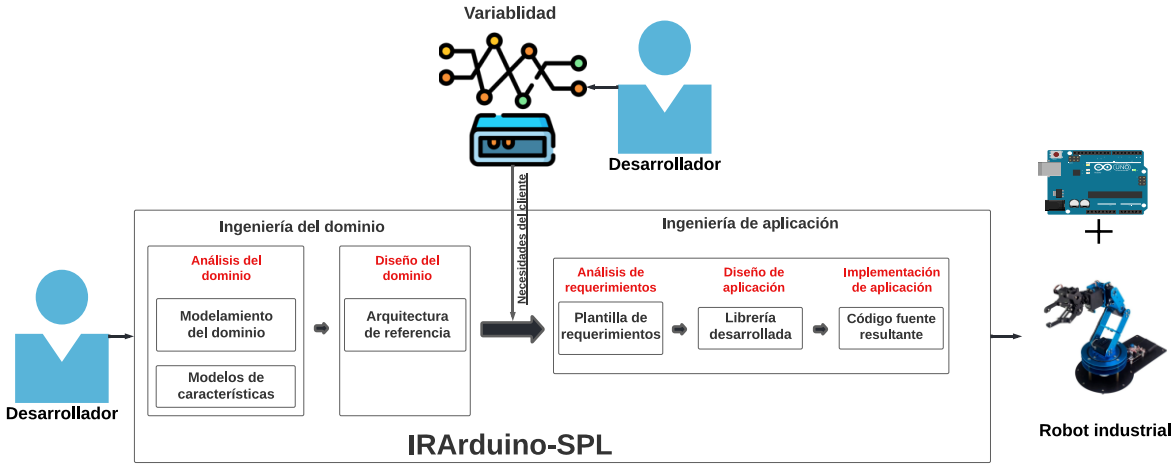


Figura 41. Idea general sobre IRArduino-SPL y su dominio de aplicación.
Fuente: Elaboración propia.

IRArduino-SPL pretende conseguir una forma de producir software para sistemas robóticos industriales con Arduino de forma que se puedan utilizar activos reutilizables, pero que los productos resultantes tengan suficientes características individuales para satisfacer las necesidades de cada usuario (en el caso de la investigación, los estudiantes).

Alguno de los aspectos más destacados IRArduino-SPL:

- **Plataforma**: IRArduino-SPL ofrece un entorno de desarrollo de software común que obedece a las abstracciones realizadas en el dominio (ingeniería de dominio e ingeniería de aplicación) de los SR industriales con Arduino. Lo anterior significa que la reutilización se planificó de forma proactiva, y se construyeron piezas reutilizables. La SPL propuesta permite la personalización masiva de productos (SR industriales) utilizando el concepto de variabilidad gestionada, es decir, se modelaron en conjunto los puntos comunes y la variabilidad en las implementaciones.

- **Flujos de trabajo**: IRArduino-SPL presenta un conjunto de flujos de trabajo que son los componentes operativos de una tarea, lo que permite estructurar el desarrollo de SR industriales en la medida que la variabilidad lo permita.

- **Tarea y pasos**: IRArduino-SPL presenta un conjunto de elementos básicos para construir un producto a partir de la SPL.

- **Estructura**: IRArduino-SPL ofrece una plantilla estándar para expresar la lógica que los desarrolladores proponen para su sistema electromecánico. Permiten a personas sin experiencia realizar tareas como el movimiento del robot o funcionalidades estándar.
- **Hardware**: IRArduino-SPL permite abstracciones del hardware que se utiliza en los SR industriales en contextos académicos (sensores, actuadores y efectores finales), permitiendo la reutilización del código en elementos parecidos.

Visión para IRArduino-SPL

Se espera que IRArduino-SPL, junto con el enfoque de ingeniería de línea de producto, permita establecerse como una sólida alternativa de reutilización en el dominio de los sistemas robóticos industriales con Arduino, permitiendo generar productos más rápidamente con un menor coste de desarrollo en el software, haciendo que los desarrolladores se preocupen por otras particularidades del dominio. En este sentido, se pretende que los estudiantes entiendan las abstracciones sobre el dominio y construyan productos a partir de los activos reutilizables. Con el tiempo, se supone que se añadirán nuevas tecnologías específicas del dominio de la robótica, que aumentarán las capacidades de la SPL, como la visión artificial, la automatización o la simulación. Asimismo, la incorporación de tecnologías de ingeniería de software como Acceleo o Telosys aumentará las capacidades de la línea. Por último, en las futuras propuestas de la SPL, se espera que puedan utilizarse en diferentes contextos, como la industria.

IRArduino-SPL: Contexto

La línea de productos IRArduino-SPL se implementará con estudiantes de diferentes ingenierías aplicadas (enfocadas al hardware) con conocimientos en robótica, electrónica y programación de microcontroladores que generalmente no tienen fundamentos teóricos o empíricos, especialmente que refiere la ingeniería de software, y la reutilización de software mediante las SPL. La propuesta se encuentra actualmente en su segunda versión y tiene una pequeña trayectoria entre los estudiantes, pero ha tenido aceptación en la comunidad académica en la que se ha socializado. IRArduino-SPL emplea un método de desarrollo centrado en la

estructuración del desarrollo de SR industriales y activos núcleo reutilizables. La propuesta es capaz de derivar productos mediante su estructura de activos y generación de código. Las experiencias de la primera versión de la propuesta y la del personal de desarrollo son los principales activos con los que se cimenta la SPL. El plan de acción para esta línea de productos debe establecer prácticas básicas de ingeniería de software y de la línea de productos.

IRArduino-SPL: Caso de negocio

Resumen

Existe un vacío en las tecnologías existentes relacionadas con la ejecución de la reutilización de software en el dominio de los SR industriales con Arduino. Esto no ha permitido fomentar y aplicar técnicas de reutilización de software para mejorar el proceso de desarrollo de aplicaciones para robots industriales con microcontroladores. Asimismo, los estudiantes no han podido beneficiarse de la reutilización planificada, eficaz y probada en otros dominios. Además, los estudiantes generalmente no entienden y no están familiarizados con estos enfoques de IS desde su formación académica, lo que ha dificultado su viabilidad e implementación en estos dispositivos cuando se necesitan a nivel profesional. Teniendo en cuenta lo anterior, realizar una propuesta desde la academia donde se realicen los primeros esfuerzos para intentar la implementación de enfoques de reutilización de software, más específicamente la ingeniería de líneas de productos de software, podría ser una excelente alternativa para remediar los problemas indicados. En esta sección se analiza la viabilidad de la producción de productos mediante el enfoque de líneas de productos de software en el dominio de los SR industriales.

Inicialmente, se pensó y planificó una estrategia de reutilización de software beneficiosa para el dominio de los sistemas robóticos industriales, aportando elementos como la abstracción de alto nivel, la no reinvención de soluciones y la aceleración en el desarrollo de software como punto de partida. Una revisión de la literatura mostró que no se habían desarrollado implementaciones de reutilización de software en el dominio, y mucho menos habían sido efectivas; propuestas como ROS han sido sólo algunas de las inducciones no específicas sobre SR industriales que han sido hechas por la comunidad, pero no adoptadas por la academia. Además, la evidencia indica que los esfuerzos para mejorar el desarrollo de software en

diferentes dominios dan como resultado una mejora dramática de la productividad, por lo que una estrategia basada en la ingeniería de línea de productos podría hacer que el desarrollo en estos sistemas mejore para que los alumnos puedan preocuparse de otros elementos importantes del dominio.

Objetivos

La SPL propuesta sigue unos objetivos estratégicos. A continuación, se expone cómo se relaciona el enfoque de la línea de productos con los objetivos de la investigación.

Convertirse en una alternativa viable de reutilización

El enfoque de línea de productos busca convertirse en una alternativa plausible para realizar la reutilización en el dominio de la SR industriales para mejorar la calidad del desarrollo de software de los robots realizados por los estudiantes. En la literatura se han reportado mejoras sustanciales en los dominios en los que se han desarrollado SPLs; este es el caso de [173] que describe Aegis, uno de los programas más exitosos del Departamento de Defensa para operar una línea de productos, con nueve programas separados operando bajo una arquitectura común. Del mismo modo, el sistema operativo Android (AOSP) se compone de millones de líneas de código fuente basadas en una arquitectura común a partir de la cual se pueden crear variantes personalizadas de la pila Android en cientos de dispositivos de hardware diferentes proporcionados por decenas de fabricantes [174]. Los anteriores son ejemplos de las ventajas que pueden ofrecer las SPL en los ámbitos en los que se implementan de forma correcta.

Reducir el tiempo de desarrollo

La línea de productos permitirá acelerar el tiempo de desarrollo de los SR industriales mediante la reutilización de componentes de software, logrando un conjunto de sistemas reutilizables (activos núcleo) pero con la suficiente variabilidad propia del dominio (variabilidad) como las tareas que pueden realizar estos dispositivos electromecánicos. Otras propuestas a nivel académico e industrial han mejorado sustancialmente el tiempo de fabricación de un producto utilizando el enfoque SPL [175].

Aumentar la productividad

El enfoque basado en línea de productos denominado IRArduino-SPL busca conseguir aumentar la productividad en el desarrollo de aplicaciones para robots industriales. Empresas como Hewlett-Packard han reportado que el número de empleados en proyectos de líneas de productos es cuatro veces menor que en proyectos similares que no utilizan el enfoque [176]. IRArduino-SPL debería alcanzar mayores niveles de productividad para seguir siendo competitiva con respecto a otras propuestas que puedan surgir en el futuro, pero por ahora la prioridad principal es establecer una alternativa real de reutilización en el dominio.

Estrategia de generación de productos

Existen varias opciones para generar productos para IRArduino-SPL; para ello, se consideraron algunos enfoques estándar como el proactivo, el reactivo o el incremental. Específicamente, en el caso de la propuesta, se adoptó una interpretación del enfoque proactivo, en el que los activos núcleo se construyen antes que los productos para formar una base conjunta sobre la que se cimentan las derivaciones. Este enfoque permite facilitar la gestión, ya que la atención se centra únicamente en la construcción de la base reutilizable y se deja de lado la producción [177]. En las empresas en las que la principal preocupación es el dinero, este enfoque es difícil de financiar y mantener porque no hay nuevos productos que generen ingresos mientras se crean los activos reutilizables. En cambio, en el caso concreto de la propuesta, el proactivo es un excelente enfoque para la generación de productos porque lo que se busca es establecer una alternativa reutilizable viable en un dominio poco explorado, más que una rentabilidad económica.

Como se indicó anteriormente, se tomó una adaptación del enfoque proactivo, en cuanto a que se generaron en primer término los activos núcleo como base reutilizable y posteriormente se generaron productos; en la primera versión, se utilizó una perspectiva en la que no hubo una preocupación particular por los activos, sino en producir código fuente basado en las abstracciones del dominio. En la segunda versión, se empleó una perspectiva basada en activos núcleo, donde la preocupación se centraba en crear activos núcleo utilizables o modificables para futuras versiones.

Análisis del entorno

En este apartado se presentan los resultados del análisis de las fortalezas, debilidades, oportunidades y amenazas que caracterizan el entorno en el que se ubicará la línea de productos IRArduino-SPL.

Puntos fuertes

El contenido de la línea de productos es un conjunto de abstracciones para acelerar el desarrollo de aplicaciones en SR industriales. IRArduino-SPL tiene la ventaja de mejorar el proceso de desarrollo de software de robots industriales con Arduino; también intenta acercar el desarrollo en Arduino a un lenguaje similar al del dominio.

Desde el punto de vista de la extensibilidad de la SPL, es posible realizar una serie de ampliaciones de los productos básicos que podrían ser la base para otras funcionalidades si la propuesta inicial tiene éxito.

La técnica de desarrollo considerada (enfoque proactivo) ha demostrado su eficacia en proyectos industriales reales [177]. Tiene la ventaja de ayudar a alcanzar los objetivos de la propuesta, centrada en el contexto educativo.

Puntos débiles

El principal punto débil de la propuesta está relacionado con la comprensión de las abstracciones realizadas sobre el dominio de los robots industriales; ya que la propuesta requiere una curva de aprendizaje pronunciada en cuanto a la forma de entender que es la reutilización en un dominio poco explorado, el reconocimiento de las bondades de este tipo de enfoques y sus ventajas a largo plazo.

La línea de productos inicial tiene la debilidad de que incluso una SPL sencilla es más compleja que un solo algoritmo. Para contrarrestarlo, una guía podría destacar las ventajas de por qué es mejor desarrollar sobre una SPL y no generar una única solución sin posibilidad de evolución, como podría ocurrir si se utilizara IRArduino-SPL.

Oportunidades

IRArduino-SPL tiene la oportunidad de establecerse como una de las primeras alternativas de reutilización de software en un dominio en que la evidencia es muy limitada. Como resultado, los alumnos de ingenierías aplicadas podrían encontrar una oportunidad para entender las bondades de la reutilización y aplicarlas en sus proyectos.

Segmento del mercado

IRArduino-SPL está enfocada en los alumnos de ingenierías aplicadas que utilizan la plataforma Arduino donde la reutilización en robótica industrial es mínima, y existe un vacío de conocimiento que puede ser llenado. Inicialmente, la SPL está pensada para permitir la reutilización de funcionalidades básicas (desplazamientos, por ejemplo), pero este mercado sigue siendo viable. También se espera que se puedan añadir nuevas funcionalidades o combinarlas con tecnologías como ROS o Acceleo para ampliar el mercado objetivo a clientes como las pequeñas o medianas empresas.

IRArduino-SPL: Modelo de dominio

En esta sección, se exponen los modelos de características con alto nivel de abstracción en el dominio de los SR industriales con Arduino para la propuesta de reutilización basada en la ingeniería de línea de productos. Estos modelos recogen las características del producto necesarias para conseguir un sistema funcional y satisfacer las necesidades de los alumnos. Es importante indicar que IRArduino-SPL ha tenido dos marcadas evoluciones para que, de esta forma, el enfoque propuesto pueda ser mejorado en base a la experiencia de los usuarios y expertos en los dominios subyacentes.

En la Figura 42, expone el primer modelo de características propuesto para IRArduino-SPL, que utilizaba un enfoque orientado principalmente a generar código a partir de las abstracciones realizadas en el dominio; en esta primera versión, el objetivo era encontrar la viabilidad de la propuesta y su aplicación. Esta visión se centró en el hardware de la plataforma Arduino, donde los actuadores, sensores y efectores finales podían ser seleccionados de una lista de ramificaciones que tiene Arduino. La reutilización en esta versión se logró generando código para cada elemento, pero no se realizaron abstracciones de alto nivel, sólo en la capa H.A.L. del embebido.

El análisis semántico y sintáctico realizado con la herramienta S.P.L.O.T. para esta primera versión, expuso una gran cantidad de productos posibles para IRArduino-SPL, lo que podría dificultar la mantenibilidad de la SPL; en concreto fueron 58.590

posibilidades, con 9 activos principales bien marcados, y una profundidad de 4 niveles, con 36 características totales y 6 restricciones.

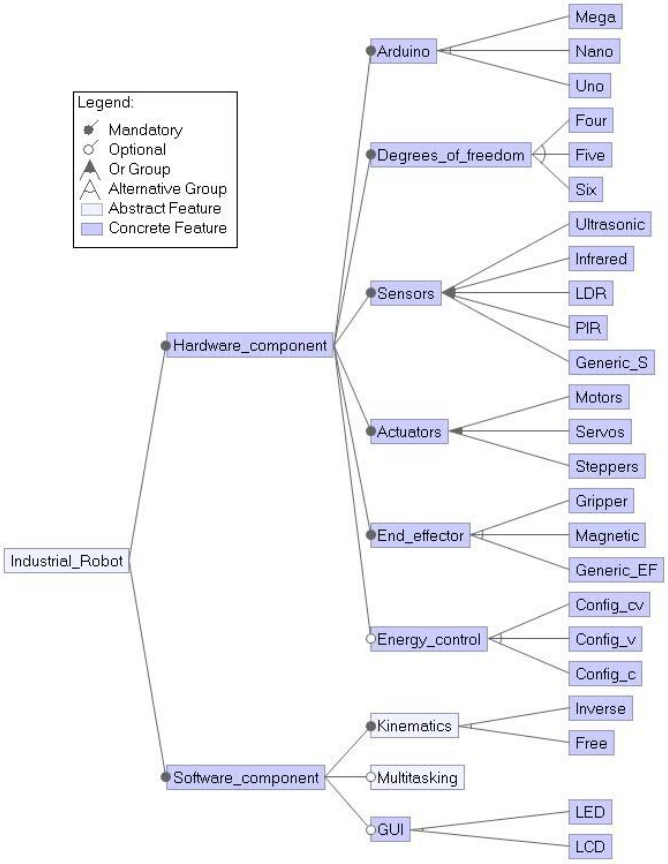


Figura 42. Primer modelo de características para IRArduino-SPL.
Fuente: Elaboración propia.

El segundo modelo de características propuesto (Figura 43) se basó en las opiniones de los usuarios y expertos que utilizaron la primera versión de IRArduino-SPL, permitiendo llevar a cabo la trazabilidad de la propuesta. Respecto a la mejora de la SPL en su segunda versión, se hizo especial hincapié en los procesos de abstracción de alto nivel en el dominio, reorientando la propuesta hacia la reutilización de software basada en activos reutilizables (activos núcleo), dejando de lado el enfoque de generación de código. Elementos como las funcionalidades habituales de los robots (tareas o movimientos repetitivos), el lenguaje de desarrollo estándar en el dominio y un mayor nivel de abstracción fueron tres de las principales mejoras introducidas en la nueva propuesta.

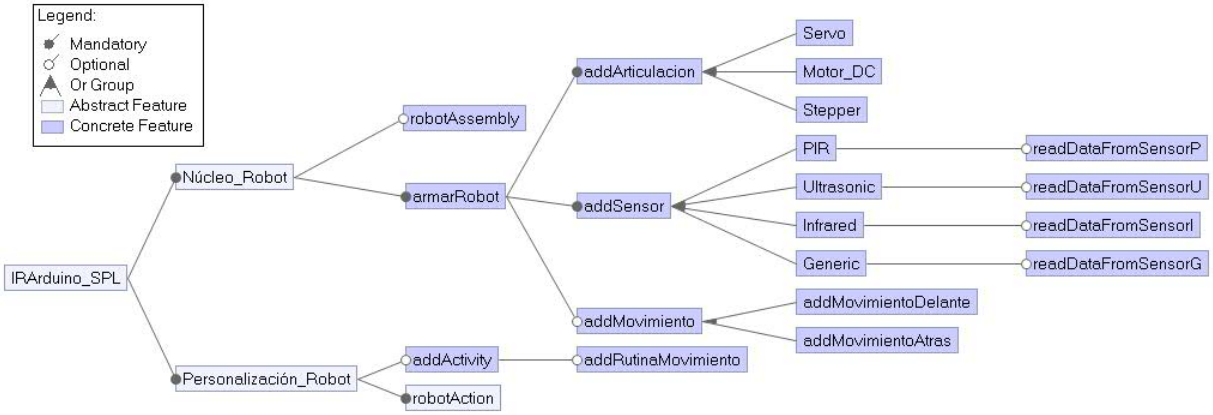


Figura 43. Segundo modelo de características para IRArduino-SPL.
Fuente: Elaboración propia.

El análisis semántico y sintáctico en la segunda versión de IRArduino-SPL con S.P.L.O.T., se observa que la variabilidad de los productos se redujo lo que permite que la mantenibilidad de la propuesta sea más sencilla. Puntualmente, se calcularon 13.440 configuraciones posibles, con 2 activos principales bien marcados, y una profundidad de 5 niveles, con 24 características totales y sin restricciones.

IRArduino-SPL: Modelo de diseño del dominio

Modelo de caso de uso

La Figura 44 ofrece una visión general de los casos de uso de la línea de productos IRArduino-SPL. En ella se muestran las principales actividades para generar un producto a partir de la propuesta. Existen dos actores principales para la SPL, el desarrollador, que es quien determina los requisitos y los accesorios que se añadirán al SR industrial, mientras que el segundo es el IDE de Arduino, que actúa como una tecnología que permite transcribir el código de alto nivel a lenguaje de máquina.

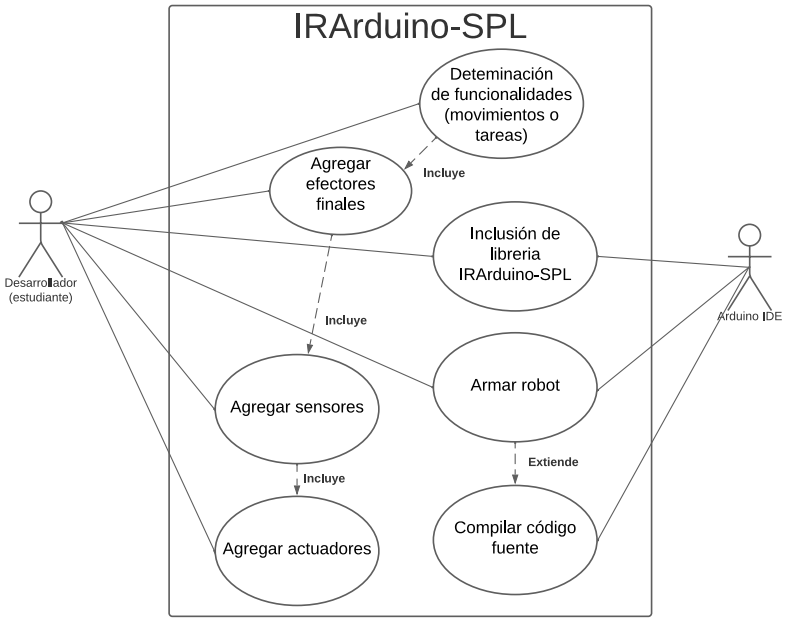


Figura 44. Modelo de caso de uso para IRArduino-SPL.
Fuente: Elaboración propia.

Modelado de dominio

Dado que la propuesta se implementó en un dominio poco explorado en el que la evidencia de reutilización de software empírica es limitada, se realizó un análisis del dominio para identificar los conceptos esenciales de la propuesta, sus principales abstracciones y sus oportunidades de mejora. Estos conceptos constituyen el vocabulario básico que describe el dominio del robot industrial con Arduino y permite determinar la homogeneidad y variabilidad de la propuesta. La Figura 45 representa el modelado del dominio en UML. Aquí, las principales clases de la propuesta están divididas por la capa framework (activos núcleo) y la capa de usuario. Es fundamental indicar que para el caso concreto, el modelo de dominio que se visualiza es el de la segunda versión de la propuesta (expresado mediante diagrama de clases), donde existen elementos como articulaciones, movimientos, sensores y actividades. Lo anterior son abstracciones en el dominio para IRArduino-SPL.

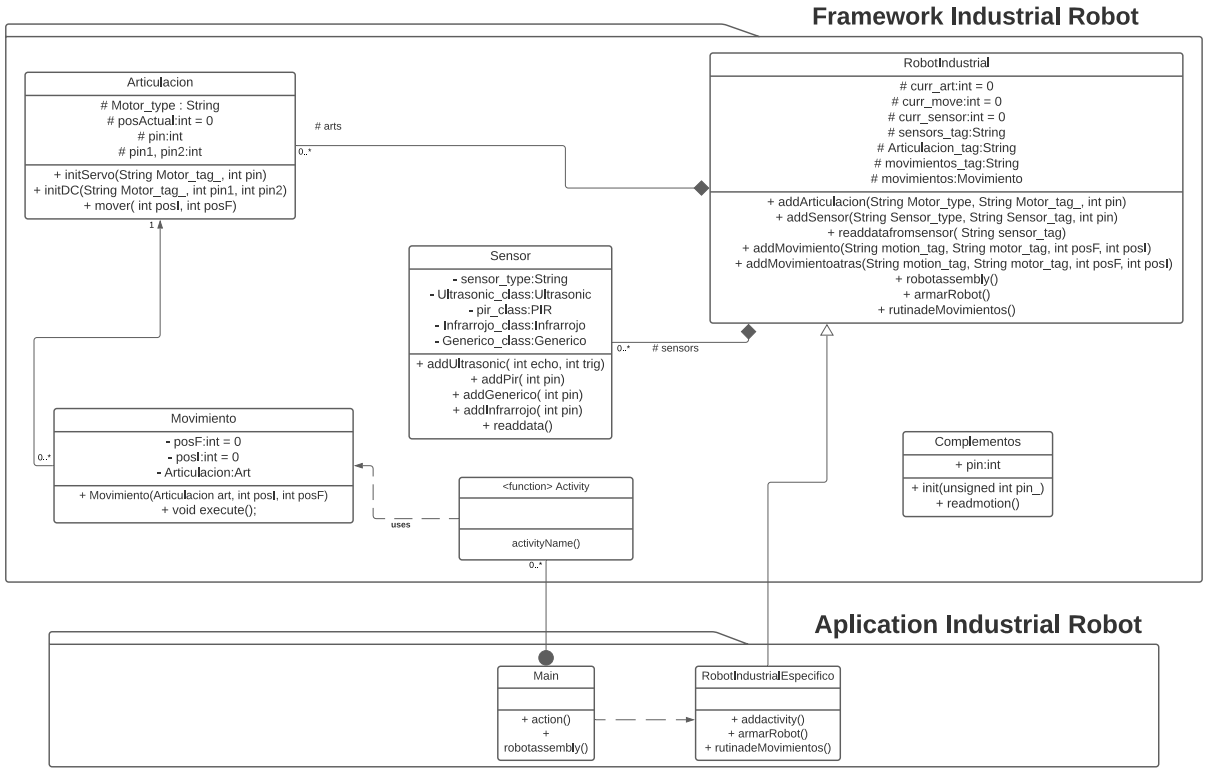


Figura 45. Diagrama de clases de la librería desarrollada seccionada en la capa framework (activos núcleo) y la capa aplicación (usuario).
Fuente: Elaboración propia.

Análisis de homogeneidad y variabilidad

En una SPL, el análisis de homogeneidad y variabilidad abarca un conjunto de productos y cómo éstos pueden generalizarse a partir de activos reutilizables. Esta sección documenta lo que es común y variable entre los productos de IRArduino-SPL.

Puntos comunes

- Los productos se enmarca dentro la misma plataforma (arquitectura) de desarrollo.
- Los productos derivados de la SPL, tienen la misma naturaleza y por lo tanto tienen elementos parecidos como sensores, actuadores y efectores finales.
- Los productos utilizan un lenguaje común de desarrollo en IRArduino-SPL.

Puntos variables

- La variabilidad más significativa de la propuesta está relacionada con las funcionalidades subyacentes que pueden cumplir los SR industriales; tareas como agarrar un objeto de un lado a otro o realizar una tarea repetitiva forman parte de la diversificación de trabajos que cumplen estos sistemas en diversos contextos particulares.
- La variabilidad de la plataforma Arduino es un reto al que IRArduino-SPL debe enfrentarse porque existen múltiples opciones entre sensores, actuadores y elementos opcionales que estos microcontroladores pueden adoptar. Por lo tanto, es un elemento que debe ser revisado, ya sea para limitar el número de dispositivos o para definir un método de extensibilidad que cubra cada uno de ellos.

Especificación de los requisitos del software

Rendimiento

La plataforma debe ser capaz de soportar las abstracciones realizadas en el dominio; aunque existen algunas propuestas, los mecanismos para hacer reutilización de software en la plataforma son limitados, aunque, por el momento, IRArduino-SPL es funcional y viable. El IDE de Arduino presenta algunos problemas de usabilidad y escasez de funcionalidades; las primeras pruebas indican que la plataforma puede soportar las abstracciones y funcionalidades de la propuesta en su segunda versión.

Adaptabilidad

La plataforma debe soportar el diferente hardware que tiene Arduino, que puede ser arquitecturas Atmel AVR, chips embebidos, y arquitectura x86. Por ahora, no se prevé la posibilidad de incompatibilidad ya que, gracias a la capa HAL, se estima la interoperabilidad entre la plataforma y la propuesta.

Requisitos de desarrollo del sistema

Evolucionabilidad

Los activos utilizados en ambos incrementos de IRArduino-SPL serán la base de los productos de las siguientes versiones de la librería. Para el tercer incremento, se adaptarán tecnologías específicas del dominio, como ROS o el simulador Gazebo,

pero los activos núcleo desarrollados deberán ser compatibles con estas nuevas tecnologías.

Extensibilidad

Debido a la naturaleza intrínseca de la plataforma Arduino y su variabilidad en cuanto a los elementos que se pueden interconectar, se establece un método que permite a los usuarios de SPL poder añadir estos dispositivos de forma individual, de manera que no se limita a los añadidos por los desarrolladores, sino que se puede emplear la propuesta en la mayoría de los casos y ser utilizada en diversos contextos de la robótica con Arduino.

IRArduino-SPL: Ingeniería de aplicación

La ingeniería de aplicación implementada para IRArduino-SPL se utilizan dos orientaciones, la primera relacionada con generar productos rápidos y observar la valides en el dominio de la propuesta utilizando la OOP como base para su desarrollo. En cambio, la segunda reorienta la generación de productos hacia el desarrollo de activos núcleo y su utilización, donde lo más importante son las abstracciones de alto nivel y las relaciones con el dominio.

Para la primera versión de IRArduino-SPL, se utilizó el modelo de dominio propuesto para identificar y construir los componentes de software reutilizables. IRArduino-SPL se basa en las abstracciones, y el modelo de dominio pensado, por lo que, para implementarlo, se realizó un generador de código como herramienta que soporta las abstracciones y el desarrollo de productos. Este generador fue programado en el lenguaje de programación Python debido a su versatilidad para el manejo de archivos, facilitando el desarrollo del generador. En esta línea, el generador de código puede crear software para sensores de ultrasonido, sensores infrarrojos y fotorresistencias, así como una plantilla genérica o exoesqueleto para sensores genéricos, independientemente del número de pines que ocupen en la placa de desarrollo. También admite la generación de código para actuadores como servos y motores, así como para efectores finales como pinzas robóticas o electroimanes. Finalmente, en la Figura 46 se puede observar la arquitectura para la primera versión de la propuesta.

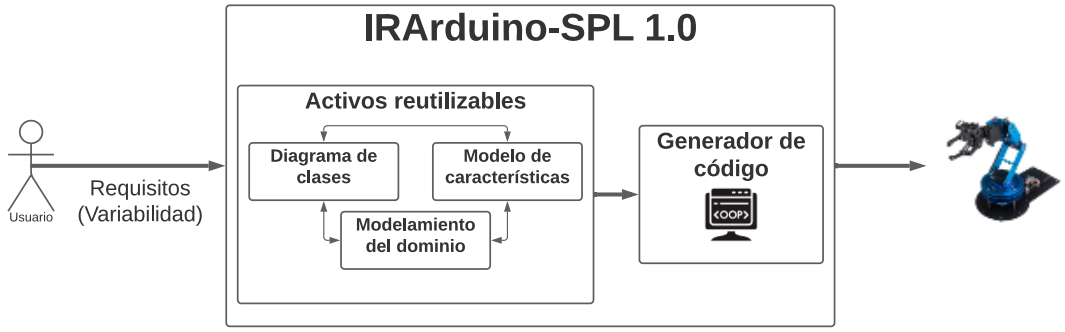


Figura 46. Arquitectura para IRArduino-SPL en su primera versión.
Fuente: Elaboración propia.

La segunda versión de IRArduino-SPL utilizó los activos reutilizables construidos en la primera versión permitiendo la implementación de abstracciones de alto nivel en la plataforma, dejando un poco de lado las preocupaciones del hardware y atendiendo principalmente a las funcionalidades del dominio de los robots. Para la Ingeniería de Aplicación de esta versión, se realizó un ajuste en la tecnología que la implementa, desarrollándose una librería específica para Arduino, que contiene los activos reutilizables y las abstracciones de alto nivel. Es importante señalar que, aunque efectivamente se trata de una librería como mecanismo de reutilización directa, también puede interpretarse como un framework que guía la construcción del software para los SR industriales en el dominio de Arduino. En la siguiente figura, se puede observar la arquitectura de la segunda versión de IRArduino-SPL.

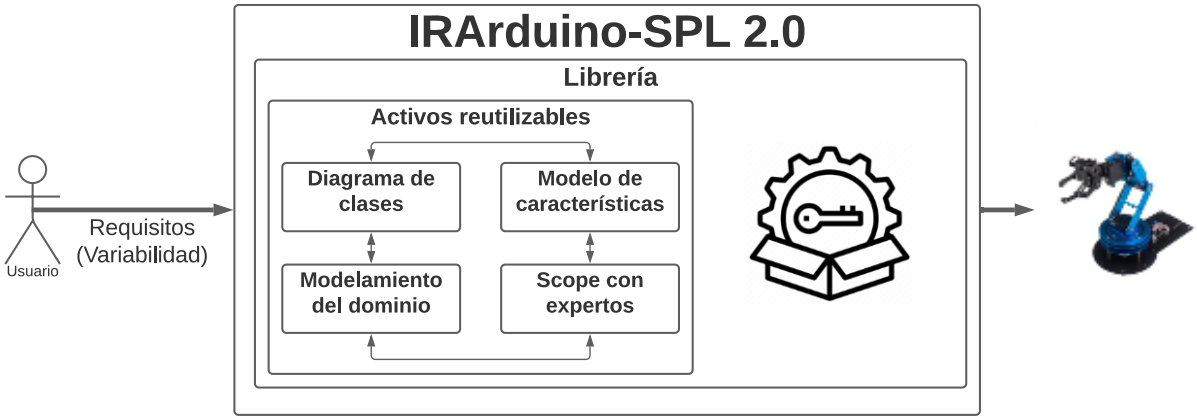


Figura 47. Arquitectura para IRArduino-SPL segunda versión.
Fuente: Elaboración propia.

Vita

Andrés Felipe Solis Pino, nació en la ciudad de Popayán, Cauca, Colombia, el 15 de septiembre de 1994, hijo del señor Guido Reinaldo Solis Pino y de la señora María Alejandra Pino Hurtado.

Terminó sus estudios en el Colegio Técnico Comfacauca, como bachiller técnico en electrónica, y sus estudios de pregrado en ingeniería mecatrónica en la Corporación Universitaria Comfacauca - Unicomfacauca. Entre sus logros destacados se encuentran ser el mejor bachiller de su cohorte en 2011, egresado destacado del programa de tecnología electrónica 2017, egresado destacado del programa de ingeniería mecatrónica 2019, y egresado destacado en investigación 2021.

Sus áreas de interés son la ingeniería de software, las líneas de productos de software, la agricultura de precisión y la Interacción Persona-Ordenador en el contexto de la usabilidad y la colaboración.

Contacto
felipe1594@hotmail.es