EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Identificación y uso de herramientas de ingeniería dirigida por modelos (MDE), para casos de estudio de desarrollo de aplicaciones embebidas y/o ciber-físicas

presentada por

ING. Julio Cesar Bernabé Analco

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Luis Gerardo Vela Valdés

Cuernavaca, Morelos, México. Enero de 2023.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

Cuernavaca, Mor., 14/diciembre/2022

OFICIO No. DCC/128/2022
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del C. JULIO CESAR BERNABE ANALCO, con número de control M19CE011, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"IDENTIFICACIÓN Y USO DE HERRAMIENTAS DE INGENIERÍA DIRIGIDA POR MODELOS (MDE), PARA CASOS DE ESTUDIO DE DESARROLLO DE APLICACIONES EMBEBIDAS Y/O CIBER-FÍSICAS"**, y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
15 DIC 2022
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO

DR. LUIS GERARDO VELA VALDÉS
Director de tesis

DR. RENÉ SANTAOLAYA SALGADO
Revisor 1

DRA. OLIVIA GRACIELA FRAGOSO DÍAZ
Revisor 2

SEP TecNM CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
RECIBIDO
15 DIC 2022
SUBDIRECCIÓN ACADÉMICA

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante
JGGS/ibm

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación y Desarrollo Tecnológico
Subdirección Académica

Cuernavaca, Mor., 14/diciembre/2022
No. De Oficio: SAC/184/2022
Asunto: Autorización de impresión de tesis

JULIO CESAR BERNABE ANALCO
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"IDENTIFICACIÓN Y USO DE HERRAMIENTAS DE INGENIERÍA DIRIGIDA POR MODELOS (MDE), PARA CASOS DE ESTUDIO DE DESARROLLO DE APLICACIONES EMBEBIDAS Y/O CIBER-FÍSICAS",** ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Educación Tecnológica al Servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
15 DIC 2022
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO
EBN

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/RMA

Dedicatoria

A mis abuelos, Francisco Analco Guerrero y Raymundo Bernabé Quintana, que en paz descansen, no les dedicaré solamente este trabajo de posgrado, también les quiero dedicar cada uno de mis logros, tanto académicos como personales. Gracias a ellos pude emprender este camino y realizar este sueño; gracias a todo su amor, a todo su esfuerzo y su empeño por hacerme una buena persona. Les estaré eternamente agradecido por sus años de dedicación, sus noches de desvelo, y su cariño siempre incondicional. Están conmigo siempre, en cada momento y en cada instante. Los amo y los amaré eternamente.

A mis abuelitas, Jacinta Vázquez Nava y Amelia Cipriano Mora, por estar conmigo y cuidar desde que estaba pequeño, por sus comidas únicas a su estilo de preparar, por seguir creyendo en mi como profesionista, como persona, como su nieto que soy. Gracias las amo y las quiero mucho.

A mis padres, Haydee Analco Cipriano y Martín Bernabé Vázquez, por enseñarme a enfrentar las adversidades de la vida, a continuar luchando, aunque todo se complique a nuestro alrededor, por darme el ejemplo de nunca darme por vencido. Por su apoyo incondicional desde que me vieron nacer hasta este momento de mi carrera. No hay con que pueda pagarles todo el cariño, amor y dedicación que han depositado en mí. Los amo papa y mama.

A mis tíos, por ser esa fortaleza en momentos de soledad, tristeza y hacerme ver que hay muchas cosas sencillas por las cuales uno como persona puede ser feliz. Por esos momentos de alegrías, y otras cosas más, Gracias.

A mi hija Ashley Juliana Bernabé Flores, por estar junto a mí y eres por quien lucho todos los días de mi vida. Eres mi tesoro.

A mi esposa Betsaida Flores Guzmán, por creer que podía lograrlo, por sentir esa compañía cuando estaba solo, ser mi compañera de aventuras únicas y alentarme a perseguir esta estrella fugaz. Gracias.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por proporcionarme a mí y a muchos otros estudiantes el apoyo necesario para potencializar nuestros sueños y habilidades.

Al Tecnológico Nacional de México / Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), por su gran servicio como Institución Académica, empeñada en formar e impulsar el conocimiento científico.

Especialmente le quiero agradecer a mi director de tesis jubilado, el Dr. Moisés González García, por el tiempo, esfuerzo, paciencia y valiosos aportes invertidos en este proyecto de tesis.

A mi director de tesis actual, el Dr. Luis Gerardo Vela, por sus consejos, aportaciones para realizar y terminar este proyecto satisfactoriamente. Un buen doctor, como amigo y como maestro. Gracias.

A mis revisores: Dr. René Santaolaya Salgado y Dra. Olivia Graciela Fragoso Diaz, por su invaluable intervención y sugerencias para realizar un trabajo de calidad.

A mis compañeros y amigos de generación: Ricardo, Regino, Xavi, Felipe, Marisol, Iván, por acompañarme en esta aventura de dos años y siempre valoraré el apoyo que me han brindado a lo largo de estos años, gracias por siempre estar ahí cuando los necesité, gracias por estos años juntos.

Resumen

En la línea de Ingeniería de Software trabajamos la generación del conocimiento de Ambientes Integrados de Modelado de Software. Estos ambientes se orientan a subir el nivel de abstracción de sistemas físicos, y mejorar la productividad, mediante el uso de lenguajes que incluyen modelos (que representan a un sistema del mundo real), metamodelos (para describir la estructura de modelos) y semántica de lenguajes, junto con transformaciones entre lenguajes, combinados con lenguajes de dominio específico. Este tipo de desarrollo permite tratar la complejidad de la plataforma y requiere un proceso que utiliza lenguajes de modelado y las transformaciones entre modelos visuales y textuales. Con el objetivo de factorizar la complejidad mediante niveles de abstracción e interés diferentes, desde modelos conceptuales de alto nivel se disminuye la abstracción hacia aspectos individuales de plataformas objetivo. En este documento se presentan los resultados de un estudio de identificación y uso de herramientas de ingeniería dirigida por modelos, realizado para reunir la evidencia publicada de las distintas herramientas MDE del tipo código abierto. A partir de una búsqueda inicial, donde se encontraron 37 herramientas MDE, posteriormente se aplicaron ciertos criterios de exclusión e inclusión, para seleccionar 2 herramientas y realizar un desarrollo asistido y colaborativo de un sistema embebido o ciber-físico (Sistema IFE).

Índice General

LISTA DE IMAGENES

LISTA DE TABLAS

ACRÓNIMOS

ARCADIA: Enfoque integrado de análisis y diseño de arquitectura.

CDB: Diagrama de clase.

CDI: Diagrama de interfaces contextuales detallas en el sistema.

CEI: Diagrama de interfaz contextual externa en el sistema.

CENIDET: Centro Nacional de Investigación y Desarrollo Tecnológico.

CIIL: Diagrama de interfaz contextual externa en componente del sistema lógico.

CIIP: Diagrama de interfaz contextual externa en el componente del sistema físico.

CIM: Modelo Independiente de Computo.

CPS: Sistemas Ciber-físicos.

DSL: Lenguajes de dominio específico.

ES: Diagrama de escenario de intercambio.

FS: Diagrama de escenario funcional.

IFE: Sistema de entretenimiento en un vuelo.

IS: Diagrama de escenario de interfaz.

JPL: Laboratorio de Jets de propulsión.

LAB: Diagrama de arquitectura lógica.

LDFB: Diagrama de flujo de datos lógicos funcionales.

LFBD: Diagrama de desglose funcional lógico.

MBSE: Ingeniería de Software Basada en Modelos.

MCB: Diagrama de capacidades de misión del sistema.

MDA: Arquitectura Dirigida por Modelos.

MDD: Desarrollo Dirigido por Modelos.

MDE: Ingeniería Dirigida por Modelos.

MSM: Diagrama de máquina de modo y estado.

M2M: Transformación de modelo a modelo.

M2T: Transformación de modelo a texto.

OAB: Diagrama de arquitectura operativa.

OABD: Diagrama de desglose de la actividad operativa.

OAIB: Diagrama de interacción de actividad operativa.

OAS: Diagrama de escenario de actividad operativa.

OCB: Diagrama de capacidades operativas.

OEBD: Diagrama de desglose de la entidad operativa.

OES: Diagrama de escenario de entidad operativa.

OOSEM: Método de ingeniería de sistemas orientados a objetos

OPD: Diagrama de Objeto – Proceso.

OPM: Metodología Objeto - Proceso.

PAB: Diagrama de arquitectura física.

PBSE: Ingeniería de sistemas basada en patrones

PDFB: Diagrama de flujo de datos físicos funcionales.

PFBD: Diagrama de desglose funcional físico.

PIM: Modelo Independiente de la Plataforma.

PMHE: Diagrama de procesos, metodologías, herramientas y entornos.

PSM: Modelo Especifico de la Plataforma.

QSOS: Calificación y selección de software de código abierto

ReMoDD: Repositorio de Desarrollo Dirigido por Modelos.

SAB: Diagrama de arquitectura del sistema.

SDFB: Diagrama de flujo de datos funcionales del sistema.

SE: Ingeniería de sistemas.

SFBD: Diagrama de desglose funcional del sistema.

SysML: Lenguaje de especificación de sistemas.

SYSMOD: Caja de herramientas de modelado de sistemas.

UML: Lenguaje de modelado unificado.

PRESENTACIÓN

La base de la presente investigación está situada en el enfoque de Arquitectura Dirigida por Modelos (MDA), que a continuación se explica.

La MDA es un enfoque para el desarrollo de software, definido por la OMG. La clave de la MDA es la importancia de los modelos en el proceso de desarrollo de software. Dentro de MDA el proceso de desarrollo de software se dirige por la actividad del modelado de los sistemas de software.

Por tanto, MDA se dirige por modelos debido a que los usa como entradas y salidas en el proceso de conceptualización, diseño, construcción, despliegue, operación, mantenimiento y modificación. Hasta el momento se ha estado hablando de que MDA se basa fundamentalmente en los modelos, pero ¿Qué es un modelo?

Un modelo es una representación abstracta de todos o algunos de los aspectos relevantes de un sistema. A menudo se representa gráficamente mediante uno o más diagramas y se escribe en un lenguaje. Un modelo tiene un significado preciso y bien entendido. Por la naturaleza del proceso de desarrollo que va de lo general a lo detallado, un modelo implica la consideración de diferentes elementos a distintos niveles de abstracción, donde cada uno de los elementos, a su vez, se organiza mediante nuevos modelos.

MDA separa la especificación funcional de la especificación de implementación y provee un enfoque abierto independiente del proveedor de tecnología, para enfrentar el reto de cambios en el negocio y en la tecnología. MDA está compuesto por tres modelos: Modelo Independiente de Computo (CIM), Modelo Independiente de Plataforma (PIM) y Modelo de Plataforma Especifica (PSM), los cuales representan el proceso de desarrollo con MDA y a continuación se describen:

- Primero, los requerimientos para el sistema se presentan en un modelo CIM, que describe la situación en la que el sistema se usará.
- Posteriormente, este modelo se transforma en un modelo PIM que describe el sistema, pero no muestra los detalles de uso, en una plataforma tecnológica particular.
- Después de obtener el modelo PIM, se realiza otra transformación hacia un modelo PSM, el cual contiene el detalle necesario para utilizar la plataforma tecnológica en la que el sistema funcionara.
- Por último, teniendo el modelo PSM se realiza una transformación que resulta en la generación de código para lograr una solución o modelo de implementación (IM).

Una vez especificado el sistema a través de los modelos, desde ellos se obtiene el producto final en base a la realización sucesiva de transformaciones. Donde una transformación es un conjunto de reglas que describen como un modelo en un lenguaje origen puede ser convertido en otro modelo de lenguaje objetivo, donde una regla es una descripción de como una o más construcciones en el lenguaje origen pueden convertirse en una o más construcciones del lenguaje objetivo. Existen, en MDA, dos tipos de transformaciones, las cuales son:

- Transformaciones verticales: son realizadas entre modelos de diferente nivel de abstracción. Por ejemplo, de los modelos CIM a PIM, de PIM a PSM y de PSM a IM.
- Transformaciones horizontales: son realizadas entre modelos del mismo nivel de abstracción. Por ejemplo, de los modelos CIM a CIM, de PIM a PIM, de PSM a PSM.

Ampliando la descripción de cada uno de los modelos que integran a MDA para su mejor compresión a continuación se detalla cada uno de ellos.

El primer modelo a definir con el que se inicia el proceso de MDA, es el CIM. Este describe la situación en la cual el sistema se usará y los requerimientos a modelarse. Tal modelo es con frecuencia llamado modelo de dominio o modelo de negocio. Este modelo es independiente de cómo se implementa el sistema.

Por otra parte, el CIM es un modelo que muestra el ambiente y funcionamiento de un sistema, de tal manera que ayuda a la exacta representación de lo que se espera que realice el sistema. Es útil, no solo para ayudar al entendimiento de un problema sino también como una fuente de vocabulario compartido para el uso en otros modelos.

Del modelo CIM se pasa al modelo PIM que describe el sistema, pero no muestra los detalles de uso en una plataforma o tecnología. La estructura de la información en este modelo podría ser bastante diferente de la estructura de la información desde el punto de vista del modelo CIM.

Para la empresa el modelo PIM podría representar información y datos específicos desde el punto de vista computacional, el cual será satisfactorio para uno o varios estilos arquitectónicos particulares.

Por último, se pasa al modelo PSM que es un modelo de un subsistema que incluye información sobre la tecnología que se usa en su realización, sobre una plataforma especifica y que hace posible contener los elementos específicos a la plataforma.

Cuando se habló de modelos en los puntos anteriores, se mencionó que un modelo siempre se escribe en un lenguaje. Ese lenguaje debe ser bien definido, debido a que tiene una gran importancia para MDA, pues el modelo debe tener asociada una sintaxis (forma) y semántica (significado) bien establecidas (definidas). Lo que permite la interpretación automática por medio de herramientas de transformación o compiladores de los modelos, fundamentales en MDA.

Este documento de tesis se divide en seis capítulos:

Capítulo 1. Introducción: Se presenta el contexto del problema, antecedentes, trabajos relacionados, que sirven como punto de referencia del trabajo propuesto, junto con el objetivo general, objetivos específicos, justificación, alcances y limitaciones.

Capítulo 2. Metodología: Se presentan las formas, metodologías, y fases de diseño para llevar a cabo el desarrollo del sistema IFE en las herramientas MDE candidatas.

Capítulo 3. Caso de estudio, descripción: Se describe el sistema de entretenimiento en un vuelo (IFE) sobre sus requerimientos, casos de uso, alcances.

Capítulo 4. Caso de estudio, implementación: Se resume de forma detallada el desarrollo del sistema IFE en las 2 herramientas MDE, en colaboración, para demostrar las actividades principales de MBSE.

Capítulo 5. Resultados y Discusión: Se presentan los principales hallazgos que se identificaron por los resultados obtenidos sobre el uso de herramientas MDE orientados a sistemas ciber-físicos.

Capítulo 6. Conclusión y Trabajos Futuros: Se incluyen las conclusiones obtenidas del análisis de evidencia existente y los trabajos futuros que se derivan de los resultados de este uso de herramientas MDE.

1. INTRODUCCION

1.1. Antecedentes

1.1.1. Ciclo de vida de ingeniería de sistemas

El ciclo de vida de la ingeniería de sistemas es un término utilizado en SE para describir el proceso de evolución de un sistema a través de diferentes fases, comenzando desde su concepción y terminando con su desecho. “Todo sistema hecho por el hombre tiene un ciclo de vida. El ciclo de vida de cualquier sistema debe abarcar no solo el desarrollo, etapas de producción, utilización y soporte, sino que también proporcionan un enfoque temprano en el retiro cuando ocurre el desmantelamiento y eliminación del sistema” (Shortell, 2015).

El propósito principal de SE es administrar el sistema en cada etapa de su ciclo de vida. Se han definido varios modelos de desarrollo del ciclo de vida para proporcionar un marco que asegure que el sistema cumpla con los objetivos deseados a lo largo de su vida. Los modelos de ciclo de vida difieren de una industria a otra y de un producto a otro, y pueden diferir en términos de los procesos y los métodos involucrados. Sin embargo, se puede deducir un modelo de ciclo de vida genérico tal que abarque todos los procesos involucrados mientras satisface los propósitos de cada una de esas etapas. Una de esas generalizaciones ha sido realizada por el estándar ISO/IEC/IEEE 15288:2015 como se muestra en la imagen 1.1 (Shortell, 2015).

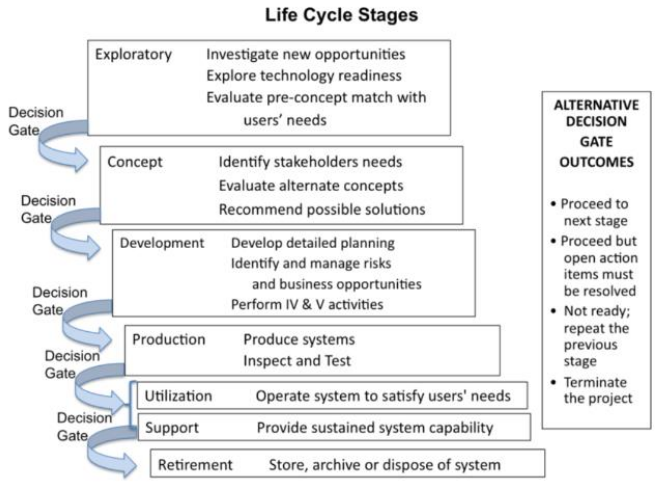


Imagen 1.1. Representación de SEBoK de los procesos del ciclo de vida del sistema, definidos por la norma ISO/IEC/IEEE 15288 (Pyster, et al., 2012).

Consta principalmente de 6 etapas: La etapa de concepto se ocupa de una amplia gama de actividades creativas como la definición del espacio del problema que implica actividades de investigación exploratoria, identificación de las necesidades de las partes interesadas y definición de las características de solución, exploración de nuevas ideas y tecnologías, necesitan refinamiento, exploración de conceptos factibles y soluciones viables y, finalmente, selección de conceptos. La etapa de desarrollo se enfoca en desarrollar el sistema de manera que cumpla con los requisitos de las partes interesadas. Un propósito clave de esta etapa es definir los requisitos del sistema, crear una descripción de la solución, implementar

e integrar los sistemas iniciales, verificar y validar los sistemas para que puedan producirse. La etapa de producción se enfoca en la fabricación y producción de los sistemas junto con su inspección y verificación. La etapa de utilización ocurre cuando el sistema es operado por el usuario final, donde el objetivo principal es brindar los servicios que el usuario final espera del sistema. La etapa de soporte ocurre cuando el sistema recibe el soporte requerido para su operación continua. Finalmente, la etapa de retiro ocurre cuando el sistema se desintegra y se retira de sus operaciones. Según el tipo de retiro deseado, el sistema se almacena, archiva o desecha.

A medida que el sistema avanza a través de sus fases del ciclo de vida, tiene que cumplir con las puertas de decisión del proyecto específicas en función de las cuales, el sistema pasa a la siguiente fase con o sin elementos de acción, continúa en su etapa actual, retrocede o incluso da como resultado la terminación del proyecto. Los modelos del ciclo de vida se han mapeado en varias ilustraciones graficas que describen el flujo secuencial del ciclo de vida del sistema. La ilustración más ampliamente aceptada es el modelo de ingeniería de sistemas en "V", en el que el tiempo y la madurez del sistema proceden de izquierda a derecha a lo largo del flujo de trabajo en forma de V (Shortell, 2015).

1.1.2. Desafíos en la implementación de la ingeniería de sistemas basada en modelos

El documento 'INCOSE SE Vision 2020' define acertadamente MBSE como la aplicación formalizada de modelado para respaldar las actividades de SE a lo largo de las fases del ciclo de vida (INCOSE, 2019). Para una implementación exitosa de un enfoque MBSE, es imperativo que las actividades de modelado respalden los procesos de SE que ocurren en las etapas del ciclo de vida descritas en la subsección anterior. Para lograr dicha implementación, es esencial una metodología de modelado sólida que consista en varios "procesos", "métodos" y "herramientas" de modo que pueda respaldar la SE en un contexto basado en modelos. El uso de las terminologías mencionadas anteriormente no ha sido preciso durante mucho tiempo y estas discrepancias semánticas tienen implicaciones en la habitación de cualquier tecnología en un entorno de ingeniería. Un estudio de encuesta mostro que la comprensión de los "términos" entre los ingenieros de sistemas en la academia y la industria sigue siendo muy heterogénea (Shortell, 2015). Aunque el estudio no analizó el uso de los términos mencionados anteriormente, es fundamental definir estos términos para evitar ambigüedades.

En (Estefan et al., 2007), utiliza las siguientes definiciones para distinguir estas terminologías clave:

- "Un Proceso (P) puede definirse como una secuencia lógica de tareas realizadas para lograr un objetivo particular. Define 'QUÉ' se debe hacer, sin especificar 'CÓMO' se realiza cada tarea" (Estefan et al., 2007). El proceso SE puede verse como modelo de proceso que define las actividades principales que se deben realizar para implementar SE. El estándar ISO/IEC/IEEE 15288 proporciona uno de esos modelos de procesos requeridos para SE.

- “Un Método (M) involucra técnicas para realizar una tarea particular. Define el ‘CÓMO’ de cada tarea. En cualquier nivel dado, las tareas del proceso se realizan utilizando métodos. Sin embargo, cada método es también un proceso en sí mismo que incluye una secuencia de tareas a realizar para ese método en particular. En otras palabras, el ‘CÓMO’ en un nivel de abstracción puede convertirse en el ‘QUÉ’ en el siguiente nivel inferior” (Estefan et al., 2007). Por ejemplo, el desarrollo de la arquitectura del sistema se puede ejecutar utilizando un método particular que incluye una serie de tareas como el análisis funcional, la definición de la arquitectura lógica y física, etc.

- “Una Herramienta (H) se puede definir como un instrumento que, cuando se aplica a un método particular, puede mejorar la eficiencia de la tarea, siempre que se aplique correctamente y por alguien con las habilidades y la capacitación adecuadas. El propósito de una herramienta es facilitar la realización de los ‘CÓMO’” (Estefan et al., 2007). El ejemplo más común de una herramienta es el diseño asistido por computadora (CAD). Sin embargo, una herramienta no necesariamente tiene que estar respaldada por computadora y también puede ser algo así como un instrumento teórico de resolución de problemas.

Una Metodología es una combinación de procesos, métodos y herramientas que se aplican a una clase de problemas comunes. El entorno de apoyo también está asociado con estas definiciones. Un Entorno (E) consiste en el entorno, los objetos externos, condiciones o factores que influyen en las acciones de un objeto, persona individual o grupo. El entorno permite el ‘QUÉ’ y el ‘CÓMO’ (Estefan et al., 2007). Varios factores ambientales como sociales, culturales, políticos o económicos tienen implicaciones significativas en la adopción de cualquier tecnología. También se aplican desafíos similares para la adopción exitosa de MBSE (Sampson, 2001).

El diagrama PMHE en la imagen 1.2 muestra la relación entre el proceso, los métodos, las herramientas y el entorno (elementos ‘PMHE’) y los efectos de la tecnología y las personas en esos elementos (Martin, 1996). En cualquier proyecto de desarrollo de SE, se deben tener en cuenta las capacidades y limitaciones de la tecnología. Además, se deben tener en cuenta los conocimientos, habilidades y capacidades de las personas que utilizan la tecnología y se deben emprender iniciativas de mejora de los empleados. Este argumento también se puede extender a MBSE.

El objetivo principal de un entorno MBSE es poder integrar las herramientas y los métodos utilizados en un proyecto. Varios factores ambientales afectaran la adopción de nuevas herramientas MBSE. Para realizar una transición fluida a MBSE, se deben superar los obstáculos culturales de gestión e implementación de herramientas junto con los desafíos de ingeniería existentes (Bone & Cloutier, 2010). En (Sampson, 2001) señala los obstáculos de gestión que suelen dificultar la adopción de la herramienta SE. Uno de los principales desafíos es lograr que la gerencia de la organización y el cliente se comprometan a permitir que los ingenieros de sistemas usen con éxito las herramientas junto con el soporte, el mantenimiento y la capacitación adecuados.

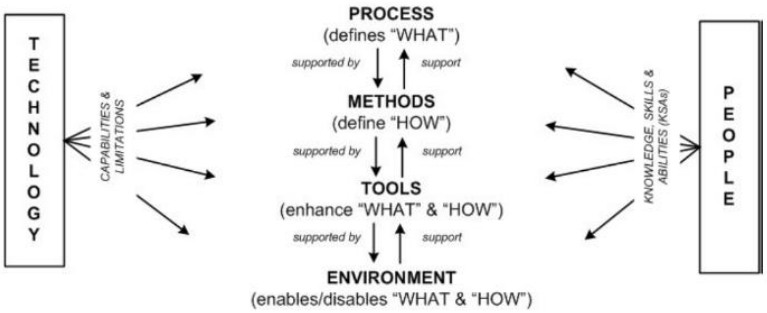


Imagen 1.2. Elementos "PMHE" y efectos de 'Tecnología' y 'Personas' (Martin, 1996).

Asimismo, la aplicación oportuna de las herramientas durante los procesos impacta enormemente en su evaluación. No solo los obstáculos de gestión, sino también varios obstáculos culturales pueden obstaculizar el proceso de adopción de herramientas. Los ingenieros de sistemas tienen una fuerte predisposición a "hacer rodar su propia" herramienta (Sampson, 2001). Esto hace que sea aún más difícil convencer a los profesionales actuales de SE de que adopten un enfoque basado en modelos. Además, la ergonomía del uso de la herramienta contribuye a los factores humanos que pueden impedir la adopción de MBSE. Por último, pero no menos importante, los ingenieros de sistemas muestran una resistencia significativa al cambio en el uso de nuevas herramientas SE. Se puede suponer un patrón similar para las herramientas MBSE.

1.2. Motivación

El trabajo de tesis está motivado por los desafíos planteados por la cuarta revolución industrial (también denominada "Industria 4.0"), los beneficios potenciales de tener una guía metodológica dentro de las herramientas de modelado con participación de un caso de estudio desarrollado en 2 herramientas de MBSE.

1.2.1. Desafíos de la industria 4.0

La complejidad de los productos modernos está creciendo rápidamente en todas las industrias. Administrar la complejidad de estos productos ha sido una tarea abrumadora, ya que las empresas se esfuerzan por crear mejores productos en un tiempo de comercialización cada vez más corto. Casi todos los productos nuevos son una evolución de las versiones anteriores con un número creciente de disciplinas involucradas (Siemens, 2011). El impacto de la nueva tecnología viene con un aumento dramático en las preocupaciones de seguridad a medida que avanzamos hacia el control del manejo de máquinas inteligentes. Esta creciente complejidad, junto con la competitividad global, exige que se incorporen enfoques de desarrollo de vanguardia en las industrias. SE, para gestionar sistemas complejos, se ha vuelto necesario a lo largo de sus ciclos de vida (Schmidt, 2006). Con la inminencia de la llamada "Industria 4.0", MBSE ha sido identificado como un facilitador clave de la ingeniería de sistemas complejos (Wortmann et al., 2017). Aunque muchas industrias han comenzado a darse cuenta de los beneficios de MBSE, las herramientas y técnicas utilizadas para su implementación deben evaluarse y evolucionar rápidamente para evitar otro aislamiento entre los equipos de desarrollo.

1.2.2. Beneficios de la orientación metodológica dentro de las herramientas

Diseñar un método SE en sí mismo, es un desafío. Adoptar una solución MBSE que satisfaga las necesidades de un método especifico es complicado, especialmente cuando la solución no se desarrolla teniendo en cuenta el método particular que se aplicará. Por lo tanto, uno de los principales desafíos a los que se enfrentan los ingenieros de sistemas durante el modelado de arquitecturas de sistemas es seguir correctamente un enfoque de modelado bien estructurado que permita un modelado preciso al tiempo que garantiza la integridad de la semántica. Se debe desarrollar una solución de modelado para proporcionar una base sólida para el método que se pretende implementar, que a su vez debe poder personalizarse según las necesidades específicas. Además, la herramienta desarrollada debe proporcionar los principios rectores del método para mejorar las tareas diarias de modelado del modelador (Rashid et al., 2015). Proporcionar una guía metodológica permite al usuario asegurarse de que se está siguiendo un enfoque correcto que se ajusta a las reglas establecidas por el método. Incluir un enfoque basado en métodos proporciona varios beneficios principalmente:

- Menor tiempo para modelar: tener una guía metódica junto con capacidades de verificación continua del modelo ahorra al modelador el tiempo para pensar en las estrategias de modelado y la precisión del modelo. Además, dicho enfoque evita la inversión de tiempo y costo que se debe realizar para definir nuevos métodos de modelado.

- Consistencia del modelo: una herramienta aumentada con un método guiado puede aumentar significativamente la consistencia de los modelos. Seguir un enfoque consistente en el modelado conduce a una representación consistente de los datos, lo que lleva a una comprensión compartida entre las partes interesadas (Reichwein & Paredis, 2011). Después de todo, se supone que poder comunicarse de manera efectiva es uno de los principales beneficios de MBSE.

- Mayor reutilización: uno de los territorios más inexplorados dentro de MBSE es el concepto de reutilización de modelos en todos los proyectos. Con un número cada vez mayor de configuraciones de productos que se ofrecen en las industrias de productos de consumo, la reutilización de modelos puede facilitar la gestión de la información de diseño de productos heredados en todos los proyectos (Cook & Schindel, 2017). La reutilización entre modelos puede ayudar significativamente a compartir datos de diseño de sistemas entre modelos.

1.3. Revisión de literatura

Esta sección revisa la literatura sobre las diversas metodologías MBSE existentes, y trabajos previos relacionados con la evaluación de métodos y técnicas basados en modelos. La sección concluye identificando la brecha que esta tesis ayudara a abordar.

1.3.1. Metodologías de ingeniería de sistemas basadas en modelos existentes

MBSE ha alcanzado un ritmo significativo en la última década. Las empresas de diversas industrias han comenzado a darse cuenta de la importancia de incorporar un enfoque basado

en modelos para el desarrollo de productos modernos. Actualmente, hay varios ejemplos de implementaciones de MBSE en un amplio espectro de industrias de fabricación de productos, cada una de las cuales intenta adoptar un modelo de ciclo de vida de SE estandarizado como el estándar ISO/IEC/IEEE 15288:2015 o desarrollar una versión personalizada propia. Independientemente, un elemento importante que influye en estas implementaciones de MBSE es la metodología de modelado que se sigue para implementar la estrategia de MBSE. Varias metodologías MBSE han sido definidas por varios proveedores de herramientas, empresas y lideres de pensamiento independientes por igual. Algunas de las metodologías populares se revisan brevemente en esta subsección.

IBM Harmony para SE

IBM Harmony-SE fue desarrollado por I-Logix Inc. que finalmente se convirtió en IBM Corporation. IBM Harmony para SE es un subconjunto de la metodología IBM Harmony para ingeniería de software (Hoffmann, 2011). Harmony para SE es un enfoque de desarrollo de arriba hacia abajo que admite arquitecturas de modelado a través de tres procesos de nivel superior,

- Análisis de requerimientos
- Análisis funcional a nivel de sistema
- Síntesis de diseño (Hoffman, 2011)

La metodología utiliza un enfoque basado en solicitudes de servicio que utiliza artefactos SysML y enfatiza la identificación y asignación de una funcionalidad requerida y un comportamiento basado en el estado en lugar de detallar su comportamiento funcional. La metodología se desarrolló para ser independiente del proveedor, pero los elementos de la metodología generalmente los aplica IBM Rhapsody.

Ingeniería de Sistemas Basada en Patrones (Metodología sistemática)

La ingeniería de sistemas basada en patrones (PBSE), se describe en (Peterson & Schindel, 2015), es una metodología MBSE que puede abordar “sistemas diez veces más complejos con una reducción de diez veces en el esfuerzo de modelado, utilizando personas de una comunidad diez veces más grande que la de un grupo de “expertos en sistemas”, produciendo antes modelos más consistentes y completos”. PBSE promueve el uso de patrones sistemáticos que podrían proporcionar un “salto inicial en la curva de aprendizaje” a partir de los patrones ya existentes y usuarios utilizando rápidamente su contenido y mejorando el patrón con el conocimiento adquirido para los futuros usuarios del patrón. La metodología sistemática utiliza el Metamodelo, que es un modelo de información relacional/objeto que se utiliza para describir requisitos, diseños y otra información en modelos, como verificación, análisis de fallas, entre otras. Un patrón es configurable y reutilizable en el modelo de una familia de sistemas formando un patrón. PBSE proporciona un modelo de datos holístico, compacto y un marco para MBSE que es adecuado para abordar los desafíos en el diseño de sistemas ciber-físicos. PBSE se puede implementar utilizando cualquier conjunto de herramientas o lenguaje de modelado de sistemas.

La caja de herramientas de modelado de sistemas (SYSMOD+)

La caja de herramientas de modelado de sistemas de (Weilkiens et al., 2016) (SYSMOD), es un enfoque orientado al usuario para la ingeniería de requisitos y el desarrollo de arquitecturas de sistemas. La metodología SYSMOD se aplica usando SysML como lenguaje preferido y se puede usar con cualquier herramienta de modelado. SYSMOD sigue un enfoque de desarrollo de arriba hacia abajo y se compone de tres artefactos principales:

1. Métodos, que dictan las mejores prácticas para crear un 'producto'.
2. Productos, que son los artefactos cruciales para el desarrollo del sistema, como requisitos o descripciones de arquitectura.
3. Roles, que son descripciones de trabajo de una persona o un operador.

Además de SYSMOD, (Weilkiens et al., 2016) también proporciona un método para el modelado de variantes (VAMOS) y la creación de arquitecturas funcionales mediante SysML (FAS) que complementan SYSMOD pero que también se pueden aplicar de forma independiente a los proyectos.

Metodología Vitech MBSE (STRATA)

La metodología STRATA MBSE fue inicialmente desarrollada en Vitech Corp (Long & Scott, 2011). STRATA se basa en el principio central subyacente de las capas estratégicas. El método se basa en analizar y resolver problemas de diseño de sistemas en capas de granularidad creciente. Comenzando en el nivel más abstracto, el enunciado del problema se analiza y se traduce en comportamientos funcionales que el sistema debe realizar para cumplir con los requisitos. Estos comportamientos se asignan a los componentes físicos que proporcionan los medios para el desempeño. Luego, la arquitectura desarrollada se prueba para ver si su rendimiento responde a los requisitos, proporcionando una trazabilidad de extremo a extremo. La metodología STRATA se puede implementar utilizando un conjunto de herramientas desarrolladas pro Vitech Corporation Inc.

Metodología de análisis de estado del JPL de la NASA

Análisis de estado es una metodología MBSE desarrollada por NASA Jet Propulsion Laboratory (JPL). Al aprovechar una arquitectura de control basada en el estado, el análisis de estado de JPL tiene como objetivo producir requisitos en el diseño del sistema y el software como modelos explícitos del comportamiento del sistema y al definir una arquitectura basada en el estado para el sistema de control (Ingham et al., 2005). El análisis de estado proporciona un marco único para el modelado de sistemas en el sentido de que,

a) se basa en el principio de que el control incluye todos los aspectos de la operación del sistema considerando una clara distinción entre el sistema de control y el sistema bajo control,
b) que los modelos del sistema bajo control deben identificarse explícitamente y usarse para lograr el consenso entre los ingenieros en sistemas,
c) comprender los estados del sistema es el aspecto fundamental para un modelado exitoso y,

d) la especificación y el diseño del comportamiento del sistema aumentan para parecerse a medida que aumenta la complejidad.

El análisis de estados proporciona un enfoque metódico y riguroso para las tres actividades principales, a saber, modelado de comportamiento basado en estado, diseño de software basado en estado e ingeniería de operaciones dirigidas a objetivos (Estefan et al., 2007).

1.3.2. Implementaciones específicas de la metodología

Durante la última década, se han realizado varias implementaciones de MBSE en industrias que van desde la aeroespacial y la defensa hasta la automotriz y la atención médica. (Carroll & Malins, 2016) proporcionan una revisión bibliográfica sistemática detallada de varias implementaciones de MBSE y tratan de identificar cómo se justifica MBSE. En su revisión sistemática de la literatura, brindan una serie de requisitos previos para que cualquier empresa en su personal y procesos junto con el compromiso de emplear un enfoque MBSE de manera bien definida. Varias implementaciones específicas de la metodología también fueron revisadas para este estudio. (Friedenthal et al., 2007) implementó el método de ingeniería de sistemas orientados a objetos (OOSEM). En resultados preliminares, pudieron modelar la arquitectura empresarial y obtuvieron una arquitectura ejecutable para soportar el análisis y la simulación del comportamiento. El análisis y la simulación sirvieron como un medio para permitir que el proyecto general validara los requisitos del cliente y de la misión y que desarrollara y demostrara soluciones de arquitectura de sistemas de sistemas (SoS). En otro estudio, (Quoc & Cook, 2012) aplicaron OOSEM para el sistema de defensa antimisiles aéreos basado en tierra e identificaron desafíos de investigación clave para implementar MBSE, es decir, ingeniería de requisitos basada en modelos, diseño y análisis de MBSE, integración de modelos y desarrollo de entornos de herramientas integradas. Una implementación similar utilizando ARCADIA se realizó en Thales Alenia Space utilizando la herramienta Capella (Calio et al., 2016). Concluyeron destacando los beneficios de un enfoque basado en el análisis funcional utilizando la herramienta Capella. Los estudios mencionados anteriormente sobre las implementaciones de MBSE indican la creciente aplicación del desarrollo basado en modelos a los sistemas de la vida real.

1.3.3. Actividades principales de la ingeniería de sistemas basada en modelos

La ingeniería de sistemas basada en modelos (MBSE) es un enfoque sistemático de modelado que se utiliza con frecuencia para apoyar las actividades de especificación, diseño, verificación y validación de requisitos del desarrollo del sistema. Sin embargo, es difícil personalizar el enfoque MBSE para el desarrollo de sistemas integrados debido a sus diversos aspectos conductuales. Además, la selección de herramientas apropiadas para realizar actividades MBSE particulares siempre es desafiante (Rashid et al., 2015). La MBSE es un enfoque bien conocido para el desarrollo de sistemas complejos y tiene características para reducir la complejidad del desarrollo, mejorar la productividad, eficiente gestión del cambio y mejora del tiempo de comercialización. La principal motivación de MBSE es utilizar el modelo como elemento fundamental en la ingeniería de sistema y asegurar que el modelo se convierta en el sistema real (Saurabh & Jose, 2013). En la imagen 1.3 se presenta el flujo MBSE, donde se inicia desde definición de requerimientos y el modelado, donde se utilizan

lenguajes como UML, SysML y MARTE. Para obtener los modelos que intervendrán en las transformaciones de modelo a modelo (M2M) o de modelo a texto (M2T), para producir modelos más detallados cada vez.

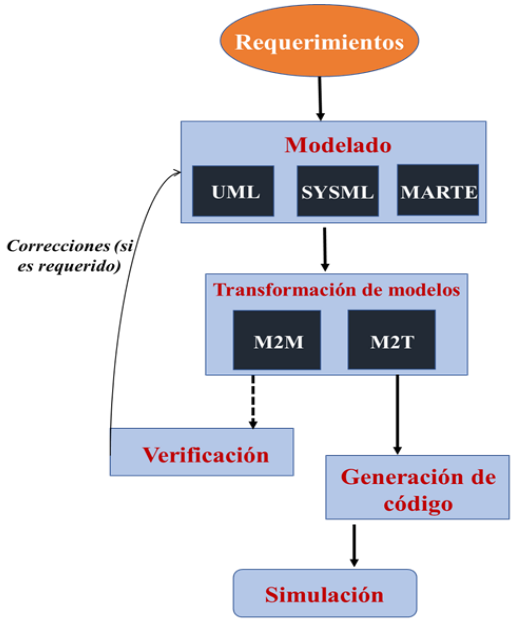


Imagen 1.3. Actividades principales de MBSE (Rashid et al., 2015).

Los modelos se verifican y se hacen correcciones (si son requeridas) en el modelado y cuando la verificación de los modelos es exitosa, se desarrollan las transformaciones siguientes requeridas, para abarcar los tres niveles de abstracción hasta generar el código (Mario et al., 2011). Posteriormente se simula, para cerciorarse que se cumple con los requerimientos especificados al inicio del proceso.

1.3.4. Estudios comparativos anteriores

También se revisó la literatura previa basada en la comparación de metodologías MBSE. Estefan, J. A. (2008) proporciona una revisión exhaustiva de las metodologías MBSE populares que sirvieron como un buen punto de partida para comprender las especificaciones, los puntos en común y las diferencias entre los diversos métodos. Amstrong, J. R. (1993) comparó la SE clásica utilizada para el análisis funcional y proporcionó varias ideas sobre parte de los métodos de SE al darse cuenta de que el alcance se limitaba al análisis funcional. Garcia, R. A. realizó un estudio de evaluación de procesos MBSE para su integración en programas de adquisición rápida utilizando un caso de estudio. En este estudio se identificó un conjunto de criterios de evaluación para evaluar los procesos. Grobshtein et. hizo una comparación uno a uno de OPM y SysML, proporcionó similitudes y diferencias en los dos, junto con desarrollos potenciales en la alineación de ambos enfoques. Finalmente, Weilkiens et. proporciona un criterio de evaluación para evaluar las metodologías MBSE en general y proporciona la comparación de las metodologías SYSMOD y MDDM como un caso de estudio.

1.4. Lagunas/huecos de la literatura

Existen principalmente dos lagunas en la literatura. En primer lugar, la literatura proporcionó varias revisiones de metodologías MBSE. Una revisión completa está disponible para las metodologías populares revisadas en la sección 1.3. Sin embargo, faltan estudios de evaluación rigurosos que evalúen las metodologías candidatas en función de criterios bien definidos. Para satisfacer las necesidades de la industria 4.0, las industrias deben tener una referencia para elegir el conjunto de herramientas adecuado para sus entornos. Por lo tanto, existe la necesidad de un catálogo de herramientas con criterios de evaluación y calidad para evaluar dichos entornos de desarrollo y los diferentes enfoques. En segundo lugar, ARCADIA / Capella y AutoFocus3 son enfoques y herramientas de modelado relativamente nuevos en comparación con las implementaciones SE tradicionales. No se ha encontrado literatura sobre una colaboración mutua entre ARCADIA / Capella y AutoFocus3. Esta tesis contribuirá a ambos vacíos de la literatura.

1.5. Objetivos de la tesis

La tesis tiene como objetivo realizar una búsqueda exhaustiva de herramientas tipo MDE, cuya búsqueda servirá para la toma de decisiones para que los arquitectos de sistemas decidan sobre la herramienta adecuada para su actividad de definición / desarrollo de arquitectura de sistemas. En base a lo anterior, construir una taxonomía de herramientas disponibles de código libre, con características y atributos generales que ofrezcan, para posteriormente realizar una comparación y evaluación de calidad de las herramientas candidatas por medio del atributo “madurez”. Por lo consecuente, se lleva a cabo, en las dos herramientas seleccionadas, el desarrollo mutuo del caso de estudio conceptual de un sistema de entretenimiento en un vuelo (IFE). El sistema se modela implementado una metodología SE llamada ARCADIA, utilizando la herramienta Capella junto con la herramienta AutoFocus3. Se identifica un conjunto general de necesidades de las partes interesadas y se introduce en el flujo de trabajo de modelado de la arquitectura. Se modela una arquitectura de concepto inicial utilizando la metodología Objeto-Proceso que luego tiene dos propósitos;

a) Proporcionar una referencia sintáctica para modelar la arquitectura candidata,
b) Servir como línea base para ir construyendo el sistema como bloques de rompecabezas.

Por consecuente, se llega a la meta final generando el código c, por medio del modelo final resultante entre las dos herramientas de modelado.

1.6. Suposiciones de la tesis

El estudio se realiza en base a algunos supuestos que es necesario aclarar antes de la evaluación. En primer lugar, el estudio excluye la parametrización del modelo de arquitectura, solo se toma como referencia una parte funcional del sistema para demostrar por medio de las herramientas de modelado junto con las actividades principales de MBSE, se supone que se deben cumplir todas las actividades, por medio de transformaciones de modelos, llegando al objetivo principal que es generar el código de más bajo nivel, que es el código C. En segundo lugar, en la búsqueda de herramientas, se supone que solo se toman en consideración las que son gratuitas y de código libre, en base a lo anterior se realiza la

evaluación y selección por medio del atributo de calidad llamado “madurez”, para que el arquitecto en específico, se dé cuenta cuales son las herramientas que están lo suficientemente funcionales y no fracase al momento de estar realizando las actividades principales de MBSE. En tercer lugar, se supone que la complejidad de la arquitectura no tiene efecto significativo en la evaluación y los resultados obtenidos de los módulos desarrollados puedan demostrar los beneficios y falencias de desarrollar un sistema a base de puro modelo junto con la metodología ARCADIA y el enfoque MBSE.

1.7. Organización de la tesis

La tesis está organizada en seis capítulos. El capítulo 1 es la introducción de la tesis, incluidos los antecedentes, la motivación, la revisión de la literatura, los objetivos de la tesis y los supuestos. La metodología utilizada para lograr los objetivos de la tesis, junto con las herramientas de modelado se explican en el capítulo 2. El capítulo 3 describe el modelo conceptual de un sistema de entretenimiento en un vuelo (IFE), utilizando la metodología Objeto-Proceso que sirve como arquitectura de referencia para modelar el sistema de interés utilizando las dos herramientas candidatas en el capítulo 4. El capítulo 5 incluye los resultados de la evaluación usando la metodología ARCADIA con las dos herramientas de modelado (Capella, AutoFocus3). También se describe aspectos destacados y diferencias clave entre las dos herramientas y analiza la importancia de MBSE en el ciclo de vida general del producto. Finalmente, el capítulo 6 concluye con el resumen de la tesis e identifica futuras áreas de investigación que pueden aumentar sustancialmente los hallazgos de este estudio.

2. METODOLOGIA

En este capítulo se describe el enfoque metodológico seguido en esta tesis. La evaluación temprana de las dos herramientas de modelado se realiza utilizando una metodología de 3 pasos:

1. Generar una taxonomía de herramientas de modelado a partir de la búsqueda exhaustiva, teniendo un panorama amplio de necesidades básicas de instalación, manejo y manuales de uso de dichas herramientas para posteriormente hacer una evaluación por medio del atributo de calidad de "madurez".

2. Generar una arquitectura conceptual de un ejemplo de sistema de interés e identificación de los módulos a desarrollar y demostrar su desarrollo en las herramientas de modelado teniendo como base las actividades principales de MBSE.

3. Después de implementar el caso de estudio y el desarrollo en las dos herramientas de modelado, se realizó una lista de los beneficios y falencias de utilizar el enfoque de las actividades de MBSE, cuyos resultados se utilizan para respaldar los hallazgos de la tesis.

Las siguientes secciones de este capítulo brindan una descripción general de los procesos de SE estándar asignados a un flujo de trabajo de desarrollo de arquitectura genérica y una descripción de las metodologías de SE utilizadas en el caso de estudio. El siguiente capitulo está dedicado a presentar la arquitectura de referencia utilizando la Metodología Objeto-Proceso (OPM). En la imagen 2.1 muestra un gráfico que ilustra la metodología seguida para la tesis.

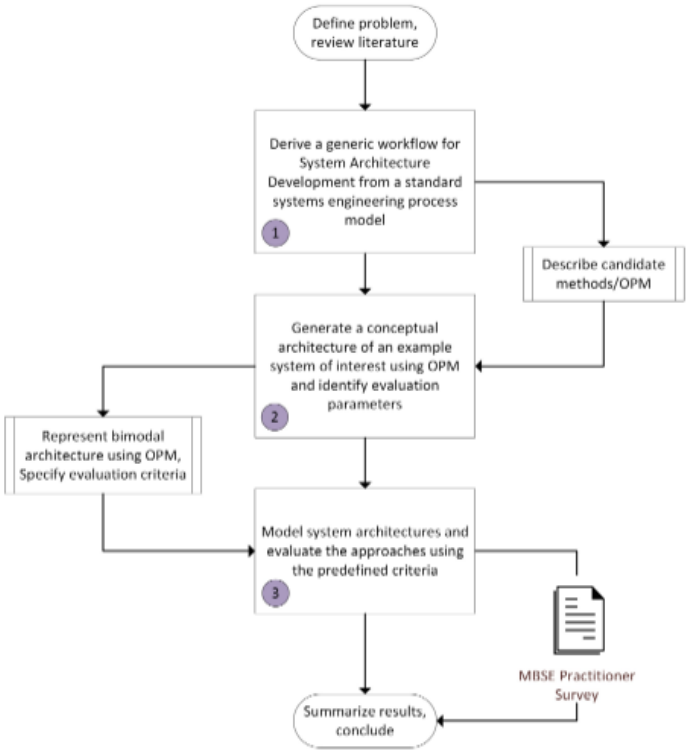


Imagen 2.1. Cuadro de metodología de investigación.

2.1. Procesos técnicos de ingeniería en sistemas

El estándar ISO/IEC//IEEE 15288 identifica cuatro grupos de procesos para admitir SE. Estos incluyen los procesos técnicos, los procesos de gestión técnica, los procesos de acuerdos y los procesos organizativos de proyectos. Los procesos técnicos que se muestran en la imagen 2.2 se utilizan para diseñar el sistema desde la definición de los requisitos hasta su disposición, permitiendo a los ingenieros de sistemas coordinarse entre los especialistas en ingeniería.

Entre los 14 procesos técnicos, los primeros cuatro procesos se ocupan directa o indirectamente de la actividad de desarrollo de la arquitectura del sistema. Podría decirse que los procesos de definición de requisitos pueden o no considerarse parte de la actividad de desarrollo de la arquitectura del sistema. Los requisitos y los análisis de casos de uso no forman parte de las actividades del arquitecto del sistema (Weilkiens, 2016). La definición de requisitos forma un subconjunto de las actividades de ingeniería de requisitos que se ocupan de describir, desarrollar, rastrear, analizar, calificar, comunicar y gestionar los requisitos que ayudan a definir el sistema en varios niveles de abstracción (Dick et al., 2017). Un aspecto importante de la ingeniería de requisitos es la verificación de los requisitos del sistema mediante la trazabilidad. Para lograr esta trazabilidad en un contexto basado en modelos, se debe asociar un requisito con los artefactos de arquitectura que satisfacen o verifican el requisito. Además, los requisitos funcionales impulsan el desarrollo de la arquitectura funcional de un sistema en base a la cual se desarrolla la arquitectura técnica de un sistema.

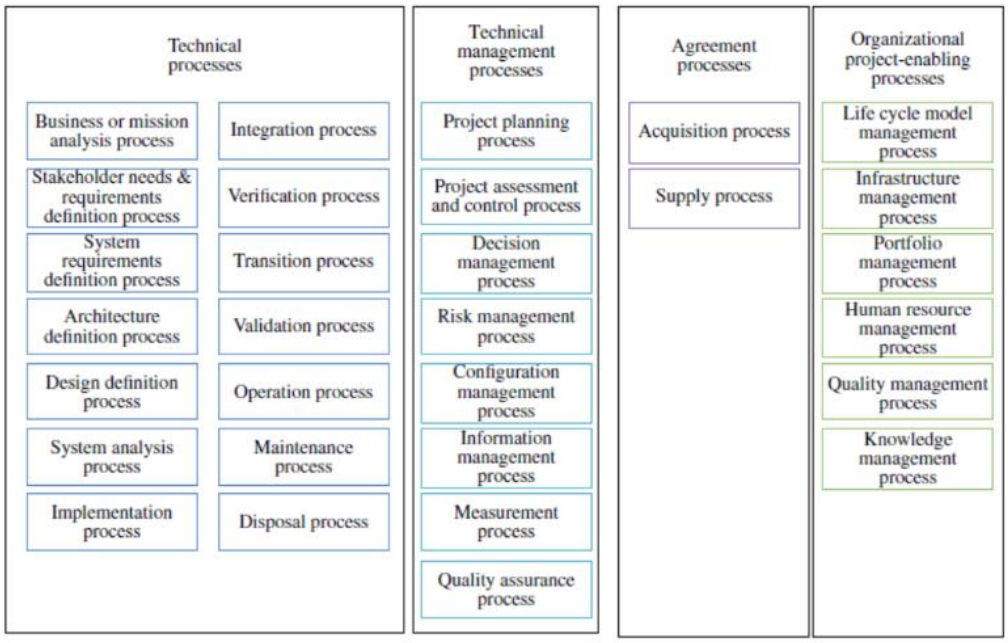


Imagen 2.2. Procesos del ciclo de vida del sistema según ISO/IEC/IEEE 15288 caracterizados en proceso (Shortell, 2015).

Realizar la definición y el análisis de requisitos de forma aislada del desarrollo de la arquitectura conduciría a un obstáculo innecesario en la consistencia del proceso. Por lo tanto, resulta lógico incluir los procesos de definición de requisitos como parte de la actividad general de desarrollo de la arquitectura. Una implementación típica de SE comienza con el contexto del proyecto, considerando el sistema y su entorno para evaluar mejor las necesidades del usuario junto con la identificación de amenazas potenciales para el sistema. En esta etapa, el sistema se considera como una entidad de caja negra que es el nivel más alto de abstracción para el sistema y se identifica su entorno operativo. Esto nos permite identificar "Qué" necesita el usuario del sistema a desarrollar y luego "Qué" debe lograr el sistema para satisfacer las necesidades del usuario. El término 'usuario' se puede extender a una 'parte interesada' y se hace así favorablemente en la mayoría de los proyectos. El 'Qué' en los procesos de SE, estos caracterizan como el 'espacio del problema'. Sucesivamente, el sistema se considera como una caja blanca y se detalla en niveles inferiores de abstracción para lograr una arquitectura de sistema final que se puede utilizar para desarrollar especificaciones y análisis de diseño más detallados. Esto forma parte del 'Cómo' de los procesos de SE, caracterizado como el 'espacio de solución'. Los primeros cuatro procesos técnicos descritos en el Manual INCOSE SE son discutido brevemente a continuación:

- **Proceso de análisis de negocio o misión:** El proceso de análisis de negocio o misión se ocupa de generar la especificación de requisitos de negocio. Este es un proceso a nivel comercial, organizacional o empresarial que comienza con la visión comercial, el concepto de operaciones y las metas y objetivos organizacionales y estratégicos a partir de los cuales se pueden definir las necesidades de la misión/negocio.

- **Proceso de definición de necesidades y requisitos de las partes interesadas:** Este proceso se ocupa de obtener las necesidades de las partes interesadas que, por lo general, son necesidades abstractas de alto nivel. Los ingenieros de requisitos luego traducen las necesidades de las partes interesadas en requisitos formales que se obtienen en un documento de especificación de requisitos de las partes interesadas.

- **Proceso de definición de requisitos del sistema:** Los requisitos de las partes interesadas definen lo que las partes interesadas del sistema quieren lograr del sistema. Estos requisitos se traducen en requisitos del sistema que establecen lo que el sistema debe lograr para satisfacer los requisitos de las partes interesadas. Por lo tanto, en esta fase se crea un documento de especificación de requisitos del sistema que impulsa las actividades de diseño de la arquitectura del sistema.

- **Proceso de definición de la arquitectura:** En función de los requisitos del sistema, se define la arquitectura del sistema, comenzando desde el sistema como una entidad de caja negra hasta el nivel deseado de detalle y descomposición. Una arquitectura de sistema, una vez definida, permite especificar la especificación de diseño para desarrollar el diseño detallado de los diversos componentes, definir planes de verificación y validación del sistema en una etapa temprana, entre otras.

El objetivo más importante de MBSE debe ser apoyar el desarrollo de la arquitectura del sistema requerida a través de los 4 procesos mencionados anteriormente. Esto, de ninguna manera enfatiza que MBSE está limitado a estos cuatro procesos. Aunque la mayor parte de la actividad de modelado de la arquitectura se aplica en estas fases, la arquitectura del sistema desarrollado podrá facilitar la extracción de información a través de varios puntos de vista relacionados con los requisitos del dominio descendente. Por ejemplo, un ingeniero de verificación podrá definir planes de verificación del sistema en función de los escenarios operativos y del sistema definidos en el modelo, o un ingeniero de diseño mecánico podrá diseñar la geometría CAD de un componente en función de la información de diseño extraída de los artefactos de modelo. La medida en que se puede extraer este tipo de información y los medios para hacerlo es un tema interesante en sí mismo que se discutirá brevemente más adelante en esta tesis.

2.2. Flujo de trabajo de desarrollo de arquitectura

Como se mencionó en la sección anterior, el objetivo principal de MBSE es apoyar SE a través del modelado de los procesos establecidos en los procesos técnicos de la norma ISO/IEC/IEEE 15288:2015. Se hace referencia a los procesos iniciales del grupo de procesos técnicos del estándar para derivar un flujo de trabajo de modelado de arquitectura de sistema genérico para facilitar un modelado de sistema coherente y, en ocasiones, ágil. Este flujo de trabajo genérico cumple con el estándar ISO 15288. La definición del término arquitectura del sistema ha sido algo ambigua en los dominios de ingeniería y los ingenieros de esos dominios la interpretan de manera diferente. Para el contexto de esta tesis, consideraremos

la siguiente definición de términos: “La arquitectura es una descripción abstracta de las entidades de un sistema y la relación entre esas entidades” (Crawley et al., 2015).

“La arquitectura del sistema es la encarnación del concepto, la asignación de la función física/informativa al elemento de forma y la definición de las relaciones entre los elementos y con el contexto circundante” (Crawley et al., 2015). La arquitectura del sistema puede consistir en una arquitectura funcional que describe una vista de su comportamiento, arquitectura lógica o arquitectura física dependiendo del nivel de abstracción y los elementos contextuales. Estos términos se definen en la siguiente sección.

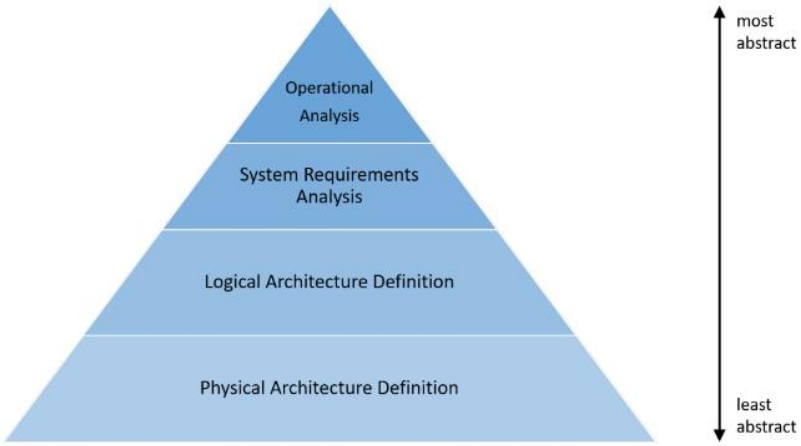


Imagen 2.3. Flujo de trabajo de desarrollo de arquitectura genérica derivado de procesos técnicos ISO 15288.

2.2.1. Análisis operacional

La fase de análisis operacional se centra en analizar las necesidades/inquietudes de las partes interesadas y traducirlas en especificaciones de requisitos. Antes de obtener los requisitos de las partes interesadas, la gerencia define las necesidades a nivel de negocio o misión que se basan en la visión comercial de la organización o empresa, el concepto de las operaciones, las metas y los objetivos estratégicos de la organización. Usando estos como guía, se obtienen las necesidades de las partes interesadas, ya sea utilizando modelos o bases de datos textuales, y luego se traducen en requisitos que se pueden vincular a los artefactos del modelo del sistema dentro de un entorno de modelado. Con base en las necesidades de las partes interesadas, se identifican objetivos de alto nivel del negocio o misión y se modelan con artefactos específicos para crear la formulación más abstracta de los requisitos, llamados casos de uso y, en algunos contextos, capacidades. Los actores y entidades que interactúan con los sistemas de interés están vinculados a la capacidad respectiva identificando así a los beneficiarios externos del sistema. Se identifica un contexto operativo junto con varios actores y entidades externas que interactúan con el sistema durante su operación. SE dicta que cada semana está destinado a proporcionar una función primaria relacionada con el valor entregada externamente a sus usuarios previstos. La función luego se descompone en funciones internas que el sistema debe realizar para lograr las funciones primarias. La funcionalidad de un sistema es la suma de sus funciones externas e internas que el sistema proporciona o en las que consiste. La funcionalidad del sistema se describe por su caso de

uso o capacidad en términos de cómo se utiliza para lograr los objetivos de sus diversos usuarios o proporcionar una capacidad específica. La identificación de una funcionalidad del sistema de alto nivel se realiza en esta fase (Voirin, 2013).

2.2.2. Análisis de requisitos del sistema

Los requisitos de las partes interesadas se utilizan para derivar los requisitos funcionales y no funcionales del sistema. Los requisitos funcionales del sistema se utilizan para identificar las funciones internas que el sistema debe realizar. Una arquitectura funcional es la descripción del sistema en términos funcionales independientemente de su tecnología. La arquitectura funcional pretende mostrar las interacciones funcionales de los componentes del sistema sin identificar los componentes reales. La capacidad de un sistema puede elaborarse mediante un conjunto de representaciones de comportamiento, como diagramas de bloques de flujo funcional, flujo de datos y secuencia de actividades o el comportamiento basado en el estado del sistema. No existe una regla sobre qué tipos de representaciones de comportamiento deben realizarse para el análisis funcional y uno puede optar por modelarlas de cierta manera en función de una determinada metodología que se sigue en el contexto apropiado. En esta fase, se asegura que todos los requisitos funcionales estén representados por las capacidades y se pueden refinar o definir nuevos requisitos si es necesario. A veces denominado análisis funcional del sistema, este es un paso clave en la definición de los requisitos funcionales. El análisis funcional (FA) es una de las técnicas más importantes en SE y es una fase distinta en muchas metodologías de SE. Aunque el análisis funcional a nivel de sistema es una fase importante, no debe limitarse a un nivel de sistema y, por lo tanto, a veces es engañoso incluir FA como una fase distinta.

2.2.3. Definición de la arquitectura lógica

El resultado principal de la fase de análisis de requisitos es identificar la arquitectura funcional del sistema. La fase de definición de la arquitectura lógica es donde el modelado cambia del “espacio del problema” al “espacio de la solución”. “la arquitectura lógica de un sistema se compone de un conjunto de conceptos y principios técnicos relacionados que soportan la operación lógica del sistema” (Pyster et al., 2012). Una arquitectura lógica es una representación abstracta de los componentes del sistema, independientemente de sus soluciones técnicas, de manera que cada función del sistema pueda ser realizada por un componente lógico correspondiente. La descomposición de funciones de nivel lógico (funciones de subsistemas/componentes) puede ser el próximo paso necesario para identificar las subfunciones que se deben realizar para satisfacer las funciones de nivel superior. Con base en esta descomposición, un sistema puede descomponerse en sus elementos de manera que cada función o subfunción debe ser realizada por un determinado elemento (forma), que puede ser un subsistema o un componente. Los elementos identificados, denominados elementos lógicos, definen la arquitectura lógica del sistema que implementa la arquitectura funcional del sistema. Durante el curso de la definición de la arquitectura lógica, se pueden identificar más funciones lógicas que conducen a la creación de los elementos lógicos correspondientes. Sobre la base de la arquitectura lógica, se desarrollan y perfeccionan varios requisitos de interfaz y componentes lógicos que deben documentarse adecuadamente.

2.2.4. Definición de la arquitectura física

La siguiente fase trata de definir la arquitectura física del sistema. “Una arquitectura física es una disposición de elementos físicos (elementos del sistema e interfaces físicas) que proporciona la solución de diseño para un producto, servicio o empresa, y está destinada a satisfacer los elementos de la arquitectura lógica y los requisitos del sistema” (Pyster et al., 2012). Es implementable a través de tecnologías. El propósito de una arquitectura física es desarrollar una solución técnica a la arquitectura lógica. En esta fase, los componentes lógicos se atribuyen a los componentes del sistema físico que realmente realizarán las funciones internas. Una función del sistema puede ser realizada por un componente lógico, pero puede ser ejecutada por uno o varios componentes físicos que están representados en la arquitectura física. Sobre la base de la arquitectura física, los requisitos de los componentes se desarrollan, perfeccionan y documentan. En un proceso de desarrollo de arquitectura, se pueden sintetizar arquitecturas físicas alternativas utilizando diversas técnicas para seleccionar la mejor solución posible que satisfaga los requisitos del sistema.

Una buena herramienta de modelado de sistemas debe ser capaz de satisfacer todas las necesidades de modelado que puedan surgir en las fases antes mencionadas. Este es un flujo de trabajo muy genérico y puede variar entre industrias y dominios de aplicación, por lo que una metodología de SE debe ser independiente del dominio que se pueda adaptar a las necesidades específicas de un proyecto si es necesario. ARCADIA (Voirin, 2017) es un método de modelado que proporciona conceptos de modelado que, cuando se aplica con una herramienta de autor como Capella, brinda orientación metodológica a los ingenieros de sistemas para diseñar las arquitecturas de sus sistemas. Desde la perspectiva de un modelador, ARCADIA/Capella es un enfoque de Método primero, Segundo idioma. Sin embargo, ARCADIA proporciona conceptos básicos de modelado que son exclusivos del método y, por lo tanto, la declaración no debe generalizarse para la metodología general.

2.3. Metodología ARCADIA

El enfoque integrado de análisis y diseño de arquitectura (ARCADIA) es un método de ingeniería de arquitectura estructurada para definir y validar sistemas multidominio, basado en actividades de ingeniería centradas en la arquitectura y basadas en modelos (Voirin, 2017). ARCADIA es un método basado en el análisis funcional y se centra en el desarrollo del sistema a partir del análisis de necesidades y el desarrollo de soluciones hasta la verificación y validación integradas. ARCADIA es muy flexible y se puede implementar utilizando un enfoque de desarrollo de arriba hacia abajo, de abajo hacia arriba o de medio hacia afuera, según se desee. Las fases clave del método ARCADIA son:

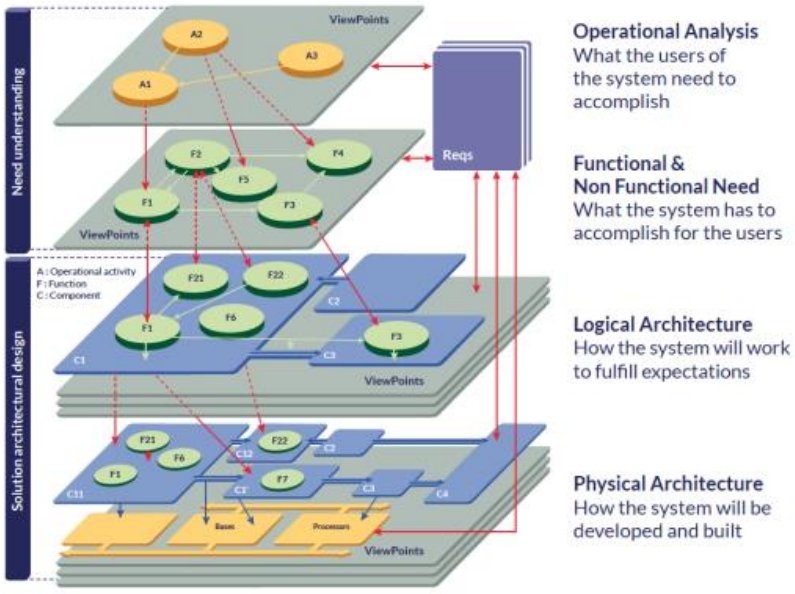


Imagen 2.4. Diferentes fases/capas del método ARCADIA (Capella, 2019).

- *Análisis de necesidades operativas del cliente*, que define las necesidades de los usuarios del sistema a cumplir,

- *Análisis de necesidades del sistema*, que define lo que el sistema necesita lograr para sus usuarios,

- *Diseño de arquitectura lógica*, que define cómo el sistema va a satisfacer las necesidades del sistema y,

- *Diseño de arquitectura física*, que define cómo se construirá y desarrollará el sistema.

Estas fases implementadas en la herramienta de modelado Capella se analizan en detalle durante la implementación en el Capítulo 4. La guía metodológica de ARCADIA se ejecuta utilizando una serie de diagramas con elementos de modelo interrelacionados que definen la estructura y el comportamiento del sistema. Además, de un método, ARCADIA proporciona conceptos de metamodelo para respaldar su enfoque metódico.

2.3.1. Capella Polarsys

Capella es una solución de software de código abierto para MBSE que proporciona un proceso y una herramienta de soporte para modelar arquitecturas de sistemas multidominio, de conformidad con el método ARCADIA (Roques, 2017). Es una herramienta de modelado híbrido que proporciona notaciones para los conceptos de metamodelo (ARCADIA ML) y está muy inspirada en SysML. Capella se desarrolla como una iniciativa del grupo Polarsys, uno de los varios grupos de trabajo de la Fundación Eclipse (Polarsys, 2019). Es fundamental destacar que Capella es la única herramienta que soporta el modelado mediante el método ARCADIA al proporcionar siete tipos de diagramas caracterizados que son: diagramas de flujo de datos, diagramas de escenarios, diagramas de arquitectura, diagramas de modo y

estado, diagramas de descomposición, diagramas de clase y diagramas de capacidad. Estos diagramas se proporcionan en todas las perspectivas de ARCADIA con distinciones especificas entre los elementos del diagrama en cada nivel.

2.4. Metodología QSOS

La metodología de selección y calificación de software de código abierto (QSOS), es concebida para calificar, seleccionar y comparar el software de código libre y abierto de forma objetiva, trazable y argumentada. Cuando se planea implementar una solución de software es necesario definir una metodología de calificación y selección objetiva, estructurada y funcional, basada en criterios técnicos. También es necesario contar con un método de calificación para establecer de manera cuantitativa las diferencias existentes entre los distintos candidatos, tanto en aspectos técnicos como funcionales y estratégicos. Por lo tanto, para poder cuantificar y medir las posibilidades reales de implantación de alguna solución de software, es necesario establecer un método de gestión de riesgos y calificación de software, que nos ofrezca la posibilidad de hacer comparaciones de software según los requisitos de las necesidades específicas que se generan dentro del contexto pedagógicos, en función de establecer criterios ponderados, en base a los cuales calificar el software y hacer una selección final de la manera más objetiva y beneficiosa.

El proceso general diseñado en la metodología QSOS se compone de cuatro etapas definidos en ciclos iterativos, como se observa en la imagen 2.5.

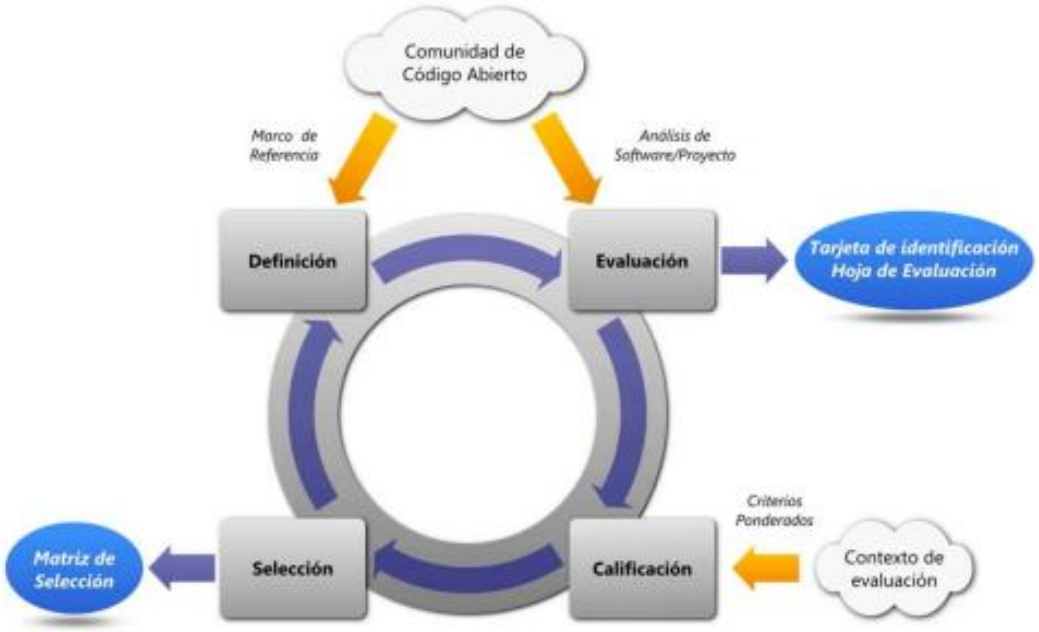


Imagen 2.5. Etapas de la metodología QSOS.

El proceso general de la metodología QSOS se puede aplicar con diferentes niveles de granularidad, lo que permite establecer el nivel de detalle del proceso en función de los requerimientos individuales de cada organización. Las cuatro etapas son:

- Definir – organizar lo que será evaluado (criterios)
- Evaluar – el software contra los criterios evaluados
- Calificar – establecer pesos relacionados al contexto
- Seleccionar – la herramienta apropiada

Mas adelante en el capítulo 5, se aborda la metodología QSOS con el atributo de calidad de madurez para justificar la selección de herramientas MBSE para llevar a cabo el caso de estudio IFE.

2.5. Metodología de trabajo AutoFocus3

AutoFocus3 es una herramienta basada en modelos y una plataforma de investigación para sistemas integrados para la seguridad. Esta construido sobre un marco de herramientas genérico basado en Eclipse, es de código abierto (licencia Apache 2.0) y admite el diseño, desarrollo y validación de sistemas embebidos y ciber-físicos en muchas fases de desarrollo, incluido el diseño de la arquitectura, la implementación, la integración de hardware/software y el razonamiento de seguridad basado en modelos formales desde los siguientes puntos de vista.

- Requisitos
- Arquitectura lógica y comportamiento
- Arquitectura técnica (hardware y software)
- Distribución (mapeo de software/hardware, programación)
- Argumento de seguridad

AutoFocus3 es un software de código abierto para MBSE que proporciona una herramienta de soporte para modelar e incrustar arquitecturas de sistemas embebidos y ciber-físicos. Como se observa en la imagen 2.6 ofrece una metodología básica sobre la descripción de análisis del sistema a desarrollar, como ejemplo, los requerimientos funcionales del sistema y la definición de la arquitectura que se va a crear. Por otra parte, la herramienta, a partir de la creación de la arquitectura del sistema, se empieza a involucrar las transformaciones de modelos hasta llegar a la generación de código para poder incrustar en una arquitectura de plataforma, como son Arduino, Raspherry, entre otros.

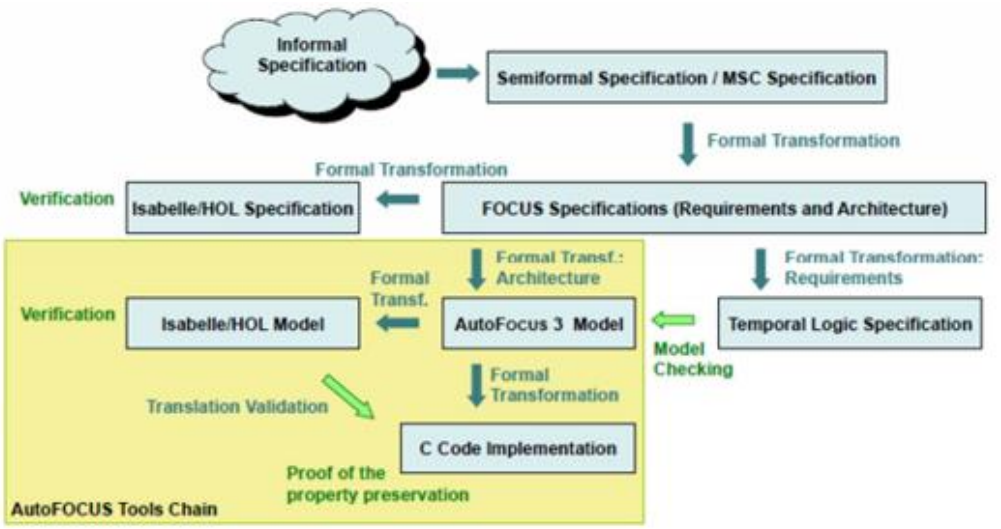


Imagen 2.6. Metodología de trabajo de la herramienta Autofocus3.

2.6. Metodología Objeto – Proceso

La metodología Objeto-Proceso (OPM) es una metodología MBSE inventada por el Dr. Dov Dori en el Instituto Tecnológico de Massachusetts. OPM se define sobre la premisa de que todo en el universo es, en última instancia, un objeto o un proceso. Por lo tanto, cualquier sistema en este universo puede representarse conceptualmente en un paradigma objeto-proceso a través de varios tipos de relaciones entre los objetos y los procesos. "Un objeto es algo que existe o puede existir física o informáticamente. Un proceso es una cosa que transforma un objeto al generarlo o consumirlo o afectarlo. Los objetos y procesos se denominan colectivamente cosas". OPM también considera la naturaleza con estado de los objetos, lo que significa que los objetos poseen varios estados y un proceso puede transformar un objeto de un estado a otro. En OPM, un sistema se puede modelar usando un solo tipo de diagrama llamado Diagrama de Objeto-Proceso (OPD). El OPD de nivel superior consiste en el proceso primario extremo (función) que el sistema necesita realizar junto con el objeto (forma) que es el sistema en sí mismo, que puede descomponerse aún más niveles inferiores utilizando los OPD de nivel inferior.

3. CASO DE ESTUDIO: DESCRIPCION

Este capítulo presenta el sistema de ejemplo para el estudio usando OPM, el sistema de entretenimiento en vuelo (IFE). El caso de estudio utilizado para comparar las herramientas de modelado es una característica de IFE. El propósito de la comparación o colaboración de las herramientas de modelado es centrarse en los artefactos de modelado que se pueden usar para desarrollar la arquitectura del sistema y otras capacidades de modelado en ambas herramientas. Este modelo de arquitectura del sistema IFE se ha desarrollado como el sistema de interés junto con los pasajeros, porque son los usuarios que interactúan directamente con dicho sistema IFE. La precisión y la lógica de la arquitectura del sistema no es una preocupación principal en este estudio. La segunda sección de este capítulo describe las perspectivas de evaluación, los criterios y las métricas utilizadas para la selección de herramientas de modelado.

3.1. Arquitectura del Sistema de entretenimiento en vuelo (IFE)

Los sistemas IFE son un tipo de sistemas de entretenimiento para el pasajero a bordo de un avión, mientras que el avión realiza el vuelo, el pasajero utiliza el sistema IFE para entretenerse y no terminar aburriéndose durante su viaje a bordo del avión. Se conocen mejor como las pantallas que están alojadas en cada asiento, donde muestran contenido de servicios de películas, series, música o información sobre el vuelo que se está efectuando. La forma habitual es que cada asiento tenga su propia pantalla individual y el pasajero controle lo que aparece con un mando o una pantalla táctil. La idea es que cada pasajero pueda ver lo que desea en la pantalla durante su viaje. En la imagen 3.1 se muestra un mapa de un avión con sus diferentes tipos de asientos, tomas de corriente eléctrica, cafetería entre otros.

Imagen 3.1. Mapa de asientos en aviones.

Este sistema está basado en un servidor ubicado en el avión y se llama Unidad reemplazable en línea (LRU), y es el responsable de controlar todas las pantallas basado en conexiones Ethernet, donde las series y películas se alojan en otros LRU con discos duros. Cada sistema IFE puede ser diferente en cuestión de software, porque cada aerolínea suele crear un software exclusivo que funcione en sus aviones.

La estructura de control del sistema IFE se muestra en la imagen 3.2, consta de dos modos de funcionamiento, “control del servicio” y “control del tipo de señal”. El control del sistema IFE proporcionará el control del servicio basado en la señal de audio y video, formato y el tipo de boleto establecida por el pasajero, donde se determina el tipo de servicio que solicitó.

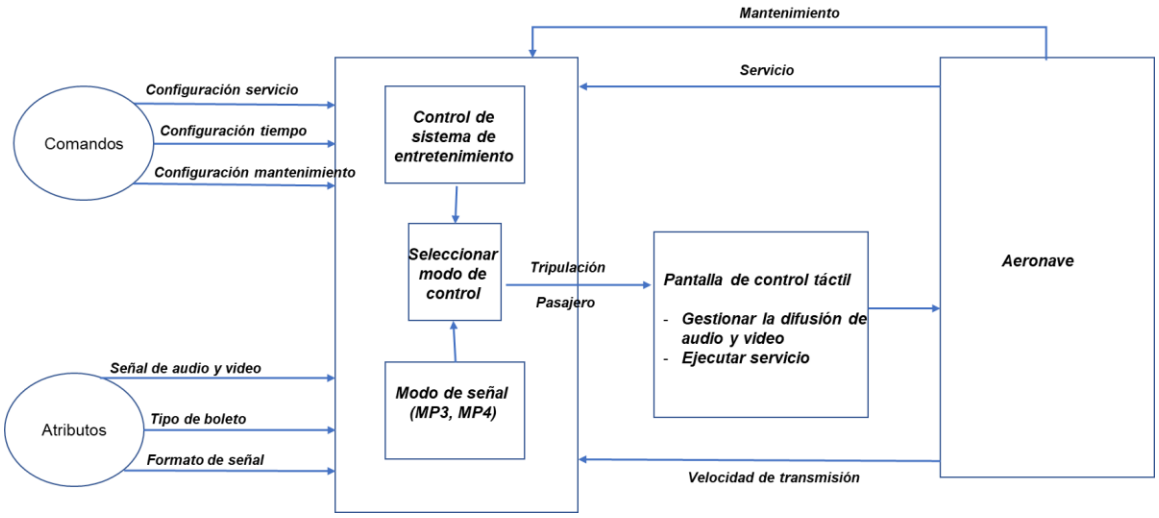


Imagen 3.2. Estructura de control del sistema IFE.

Necesidades de las partes interesadas del sistema IFE.

Antes de comenzar el proceso de modelado, en la tabla 1, se identificó un conjunto de necesidades de las partes importantes que servirán como base para el proceso de modelado.

Tabla 3.1. Necesidades importantes para el proceso de modelado del sistema IFE.

Nombre	Descripción
Encendido / Apagado	El usuario debe poder manejar de forma segura el sistema ON / OFF mientras realiza su viaje.
Control del servicio	El usuario debe poder manejar el servicio seleccionado, de tal manera que tenga un listado (menú) de las propiedades que se maneja en cada uno de los servicios ofertados por la aerolínea.
Control de la señal	El usuario debe poder seleccionar el tipo de formato del contenido multimedia para reproducir en la pantalla individual de su asiento
Mantenimiento	El usuario debe poder actualizar el contenido de los servicios con mejoras en cuestión de películas, música, juegos, entre otros.
Desactivación de emergencia	El usuario debe poder desactivar el sistema de entretenimiento por ocasión de alguna turbulencia o mal estado del tiempo climático.
Control de anuncio promocional	El usuario debe poder promocionar la marca de la aerolínea con anuncios de publicidad en cuestión de seguridad del pasajero, servicios ofrecidos, cuidados de la unidad, entre otros.

Para poder desarrollar el sistema, primero tenemos que ver los requerimientos funcionales con su respectivo actor y poder acotarlo a un caso de uso para poder cumplir con el requerimiento que se especifica y determinar cuántas partes del sistema se pretende construir. En la imagen 3.3, se observa un diseño general de los actores del sistema de entretenimiento

en un vuelo de manera general y por consecuente, se muestran los casos de uso y la relación de actores para realizar esa parte funcional del sistema.

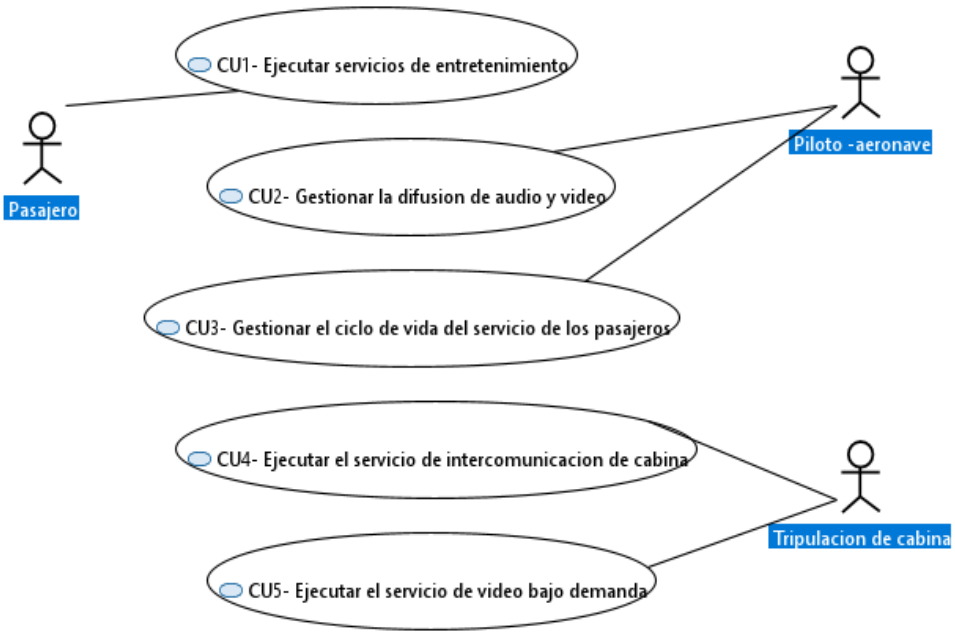


Imagen 3.3. Diseño de los casos de uso del sistema IFE.

3.1.1. Modelado IFE usando la metodología objeto – proceso

El "principio de la función como semilla" es uno de los principios básicos de OPM que establece que "el modelado de un sistema comienza definiendo, nombrando y representando la función del sistema, que es también su proceso de nivel superior". El proceso de modelado en OPM comienza con la identificación de la función principal que realiza el sistema, que es el valor que el sistema entrega a sus beneficiarios, que, en nuestro caso, seria "Alimentando IFE". El diagrama Objeto-Proceso se denomina Diagrama de sistema (SD) en todos los niveles de abstracción. El objeto y el proceso representan la forma y función del sistema respectivamente. OPM fue elegido como el método de modelado de arquitectura de referencia principalmente por dos razones:

- "OPM es fundamentalmente simple, tanto para construir una arquitectura de sistema compleja como para razonar a través del sistema", y,
- "OPM permite una representación clara de las muchas características importantes de un sistema: sus conexiones topológicas; su descomposición en elementos y subelementos; las interfaces entre los elementos y la emergencia de la función a partir de los elementos".

La comparación de las arquitecturas candidatas en las herramientas de modelado con la arquitectura OPM proporcionaría un contexto claro para la creación del sistema IFE en las herramientas de modelado y tener una guía en el contexto de crear el sistema IFE de buena forma. El modelado comenzó con la creación de la función principal del sistema. En la imagen 3.4, se describe el diagrama del sistema IFE de alto nivel que incluye la función de

nivel superior como el proceso, el objeto y los procesos ambientales que invaden el proceso principal del sistema IFE.

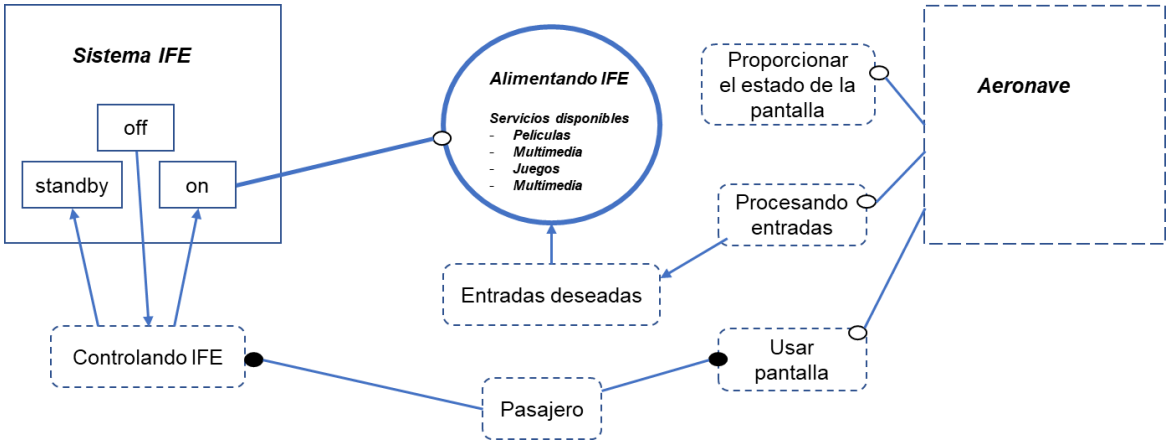


Imagen 3.4. Diagrama de alto nivel del sistema IFE.

3.2. Criterio de evaluación

La evaluación de las dos herramientas se basa en 3 parámetros. Esta sección describe los parámetros de los criterios desarrollados en parte para el estudio haciendo referencia a varias fuentes.

3.2.1. Búsqueda de herramientas MDE

Los repositorios son sistemas de información que tienen como fin organizar, preservar y difundir, en el modo de acceso abierto, recursos científicos y académicos. Deben ser sustentables a largo plazo, permiten incrementar y fortalecer el acceso a los recursos, cuentan con mecanismos para el depósito de material por el creador u otra persona y una arquitectura para manejar contenidos y metadatos, además de prestar servicios como búsquedas, recuperación, administración, control de acceso y permisos (Gloria et. al. 2019).

Existen varios tipos de repositorios: a) temáticos, los materiales que contiene o abarcan una determinada disciplina, b) nacionales, representan parte de la producción científica de un país, c) de datos, reúnen elementos que formaron parte de investigaciones, d) agregados, unen varios repositorios, y e) institucionales, contienen la producción académica y científica que genera una institución (Gloria et. al. 2019).

En la tabla 3.2, se presenta una visión general de los repositorios más representativos en el área de desarrollo por modelos. La información incluida en la tabla proviene de los sitios web (URL) incluidos en la tabla. ReMoDD es el repositorio más destacado respecto que ofrece una especie de consulta sobre Modelos, Meta-modelos y Transformaciones y lo complementan 24 herramientas de modelado que básicamente cubren desde la etapa de requisitos hasta la generación de código.

Tabla 3.2. Muestra de herramientas de modelado empleadas en repositorios colaborativos.

Repositorio	Artefactos administrados	Herramientas encontradas	Herramientas disponibles para modelado	URL
Rational Rhapsody	SysML Modelos UML	1	Rational Architec	www-03.ibm.com
Enterprise Architec	SysML Modelos UML	3	Enterprise Architec, StarUML, AndroMDA.	www.sparxsystems.com.au
Visual Paradigm	Modelos UML	1	Visual paradigm	www.visual-paradigm.com
MagicDraw	Modelos UML	Ninguna herramienta	No encuentra la página, el sitio está en mantenimiento	www.nomagic.com/getting started/solutions
Modelio	Modelos	1	Modelio	www.modelio.org
GenMyModel	Modelos UML	1	Te pide una suscripción por una pequeña cuota para descargar la herramienta	www.genmymodel.com
CDO	Modelos Meta-modelos	1 recurso	EMF	www.eclipse.org/cdo
EMFStore	Modelos Meta-modelos	1	EMFStore (se puede utilizar para instancias de EMF Model existente)	http://eclipse.org/emfstore
MORSE	Modelos	No se encuentra disponible	Ruta no encontrada	www.infosys.tuwien.ac.at/ prototyp/morse
ModelBus	Modelos, Meta-modelos Transformaciones	3 herramientas	Enterprise Architect, Rational Architect, Matlab Simulink,	www.modelbus.org
AMOR	Modelos Meta-modelos	Ninguna herramienta	Página principal de prototipos del repositorio, no menciona herramientas.	www.modelversioning.org
ReMoDD	Modelos, Meta-modelos Transformaciones	24 herramientas	Referencia de la tabla 2 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 33, 35]	www.cs.colostate.edu/remodd
GME	Modelos Meta-modelos	No se encuentra disponible el sitio	La ruta es la correcta, no se encuentra disponible por mantenimiento	www.isis.vanderbilt.edu/ projects/gme
MDEForge	Modelos, Meta-modelos, Transformacion, editores	1 herramienta	ATL (lenguaje de transformación Atlas)	www.mdeforge.org

El repositorio de ReMoDD, se tomó como base para una segunda búsqueda más a detalle de las herramientas MDE, porque fue el repositorio con más herramientas disponibles para consultar y eran las más activas en el mercado de desarrollo MDE.

La prioridad era encontrar herramientas que satisficieran las actividades de MBSE, ya sea que cumplieran con una actividad específica o herramientas que se dedicaran a soportar más de 2 actividades. Si realizaban todo el proceso de MBSE de principio a fin serian mejor, porque no se tendrían problemas de compatibilidad de modelos, al momento de realizar el proceso de MDE.

3.2.2. Taxonomía de herramientas MDE

Para realizar la taxonomía se tomaron criterios o características sobre cada herramienta: si es de código abierto (open-source), en que entorno se trabaja (Eclipse IDE), las plataformas de desarrollo (Windows, Linux, MAC), su última versión de descarga, la relación con otras herramientas de la misma taxonomía. Lo más importante es la actividad o etapa que soporta, del enfoque MBSE, pues permite determinar que herramientas se deben elegir cuando se realiza una actividad, o si existen algunas herramientas que soporten el flujo completo de desarrollo de MBSE.

En la tabla 3.3, se presenta las herramientas MDE estructuradas de acuerdo a una variante de la categoría técnica “funcionalidad específica” (Jon Whittle et. al. 2015). En este caso, funcionalidades para las realizar las actividades de la ingeniería MBSE, en el dominio de

sistemas de tiempo real: el modelado, las transformaciones de modelos (M2M, M2T), la verificación de modelos, la generación de código y la simulación. Para facilitar la elección de alguna herramienta considerando su propósito.

Tabla 3.3. Taxonomía de herramientas MDE.

No.	Herramienta	Sistema operativo /plataforma	Asociación con otras herramientas /lenguajes	Última versión	Tipo de software
Modelado (7)					
1	*AndroMDA*	Multiplataforma	Todo tipo de herramientas UML	3.5	libre
2	*Enterprise Architect*	Windows Mac IOS	ActionScript, Ada,C y C++, C#, Java, Delphi, Verilog, PHP, VHDL, Python, System C, VB.Net, Visual Basic	13.5	Versión de prueba
3	*Rational Architect*	Windows Linux Mac OS X	Todo tipo de herramientas UML	9.7	Versión de 30 días, software de paga.
4	*Visual paradigm*	Windows	UML, SysML, BPMN, SoaML	16.0	libre
5	*Modelio*	Windows Linux MAC	UML, BPMN	4.0	Software de paga, versión gratis de 30 días.
6	*StarUML*	Windows Mac Linux	UML, BPMN, C++, C#, java	3.2	libre
7	*Sirius*	Eclipse	EMF, GMF, XML	Paquete libre para instalar	Herramienta de modelado libre
Transformación de modelos (18)					
8	*ATL*	Windows, Mac, Linux, Eclipse IDE.	Motores de transformación como QVTo y QVTd. Relación EMF	4.1	Plugin, complemento de Eclipse
9	*VIATRA*	Windows, Mac, Linux, Eclipse IDE.	Objetos EMF	2.2.1	SDK de consulta de tipo de transformación para Eclipse.
10	*OCL*	Windows, Mac, Linux, Eclipse IDE.	UML, EMF, Papyrus, Xtext y enlaces Ecore	2.4	Conjunto de librerías org.eclipse.ocl. pivot/model/ OCL-2.5.oclstdlib.
11	*MWE*	Eclipse	XML	Versión tipo plugin	Traducción de modelos en lenguaje XML.
12	*QVT*	Windows, Mac, Linux, Eclipse IDE	Complementos de QVT	No especifica	Librerías de QVT
13	*EuGENia*	Windows, Mac, Linux, Eclipse IDE.	Enlaces Ecore, EMF	Versión no disponible	Editor para GMF
14	*Exced*	Windows, Mac, Linux, Eclipse IDE.	Modelos EMF. Combinación con Modelink	versión no disponible	Editor para ver características de modelos EMF.
15	*Modelink*	Windows, Mac, Linux, Eclipse IDE.	Editor Exeed	Versión no disponible	editor que consta de 2-3 editores basados en árbol EMF en paralelo.
16	*Workflow*	Windows, Mac, Linux, Eclipse IDE.	Modelos EMF	0.8.5	Transformación y validación de los modelos EMF
17	*HUTN*	Windows, Mac, Linux, Eclipse IDE.	ETL, EGL, EVL	1.0	Transformaciones de XMI
18	*Concordance*	Eclipse	EMF	No incluye versión	Un marco de gestión de la integridad del modelo
19	*Gemoc Studio*	No especifica.	Xtext, Sirius Animator	No incluye versión	proporciona un enfoque modular para complementar un metamodelo Ecore

20	*GME*	Windows, Mac, Linux	GREAT, UDM, CAPE	18.10 .exe	kit de herramientas configurable para crear entornos de modelado y de dominio.
21	*GREAT*	No especifica	GME, UDM	1.7.8 .exe	reglas de reescritura de gráficos
22	*UDM*	No especifica	GME, GREAT	3.2.15 .exe	archivo de esquema XML (. DTD o .XSD).
23	*CAPE*	plataforma de aprendizaje basada en la web	GME	2.6	representar el diseño de experiencias de aprendizaje en línea
24	*Epsilon*	MAC, Windows, linux	EMF, XMI, OCL, HUTN, Ecore, SIMULINK, XML	1.5.1	Herramientas desde la transformación – generación de código.
25	*MOFScript*	Lenguaje de transformaciones de modelos	Acceleo, JET, Xtend, EMF	No especifica	Plugin de verificación de programas, transformaciones
Generación de código (3)					
26	*Acceleo*	Windows, Mac, Linux	ATL	3.3	Generación de código basada en el estándar MOFM2T de OMG
27	*EGF*	Windows, Mac, Linux	EMF	No especifica	Plugin o complemento de EMF para la generación de código
28	*EMF*	Windows, Mac, Linux, Eclipse	Núcleos EMF	No especifica	Plugin para generar el código
Verificación de modelos (2)					
29	*EUnit*	Eclipse	EMF	No especifica	pruebas unitarias
30	*UPPAAL*	Mac, Windows, linux	Validación y verificación C++, Java	4.1.24	Requiere licencia
Simulación (2)					
31	*SIMULINK*	Mac, Windows, linux	MATLAB, C, C++, HDL	R2019	Modelado y simulación del sistema Prueba gratuita 30 días
32	*CDT*	Eclipse	C, C++	9.10.0	Compilación y depuración
Conjuntos de herramientas que soportan el proceso completo (3)					
33	*Papyrus*	Mac, Windows, Linux, Eclipse	UML, SysML, MARTE	4.6.0	Requerimientos-generación de código.
34	*AutoFocus3*	Mac, Windows, Linux, Apache	C, java	2.16	Requerimientos-simulación
35	*Capella*	Windows, MAC, Linux, Eclipse	UML, SysML, Java, C++	1.4.0	Requerimientos-generación de código

La mayoría de las herramientas trabaja con el entorno de desarrollo Eclipse, por lo que se disminuye la posibilidad de tener problemas de compatibilidad de modelos, en todo el desarrollo de las actividades de desarrollo MDE. El objetivo de esta taxonomía de herramientas de la MDE es facilitar la identificación de herramientas que nos ayuden en el desarrollo de artefactos de los elementos principales de MBSE. Para mejorar la decisión sobre qué herramientas se pueden utilizar para desarrollar un sistema embebido o ciber-físico.

3.2.3. Modelo de calidad de "Madurez"

Para la selección de herramientas MDE, se tomaron como prioridades las siguientes características:

- Que las herramientas fuera de código abierto (open-source),

- Que tuvieran una asociación entre herramientas para lograr la meta de valor (abarcar al menos un elemento de la MBSE) para el desarrollo del sistema embebido y las plataformas disponibles,
- Que la última versión esté disponible para utilizarla sin ninguna dificultad,
- Y, por último, que abarque los elementos de MBSE, para la generación de artefactos de los niveles de abstracción de MDA (CIM, PIM y PSM), desde la definición de requerimientos hasta la generación de código.

Para hacer la comparación y evaluación de las herramientas MDD se utilizó la metodología QSOS. Como parte del primer paso se elaboró un mapa mental sobre los criterios a evaluar en las diferentes herramientas FLOSS. En la Imagen 3.5, se muestra el modelo de calidad para medir la madurez de las herramientas MDE, el cual se tomaron como atributos el desarrollo, legado, actividad, industrialización, y características.

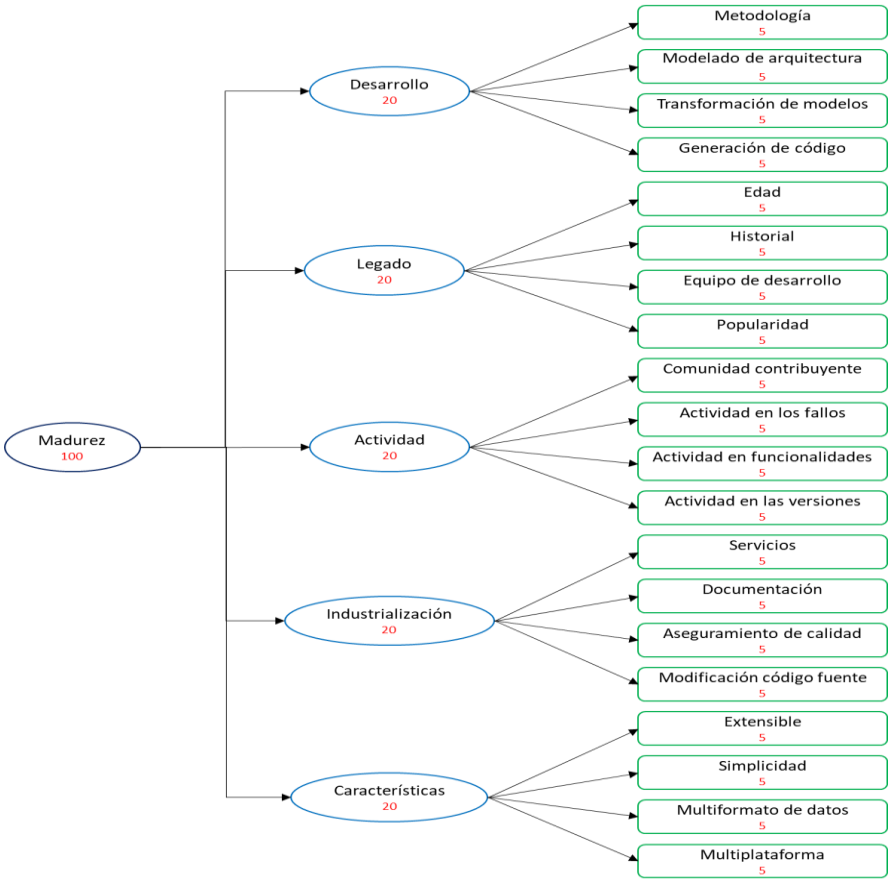


Imagen 3.5. Modelo de calidad para medir la madurez de las herramientas MDE.

Mas adelante en el capítulo 5, se determinan los valores deseados sobre una evaluación de las herramientas de modelado y la justificación de la selección de las dos herramientas candidatas para el uso y colaboración del desarrollo del sistema IFE en dichas herramientas.

4. CASO DE ESTUDIO: IMPLEMENTACION

En este capítulo se describe la implementación del caso de estudio de un sistema de entretenimiento en un vuelo utilizando ARCADIA/Capella junto con la herramienta AutoFocus3. La sección 4.1 describe el enfoque ARCADIA/Capella para modelar el caso de estudio. En la sección 4.2 describe la colaboración de la herramienta AutoFocus3, continuando el modelado y transformación de código a partir de la definición de la arquitectura física del sistema de entretenimiento en la herramienta Capella. Ambas herramientas son únicas a su manera y tienen un propósito similar al mismo tiempo. El flujo de trabajo de desarrollo de arquitectura desarrollado en el Capítulo 2 se utiliza como plantilla para mantener la coherencia a lo largo de este capítulo. En el **Anexo A**, se muestran el resto de los diagramas generados en la herramienta Capella.

4.1. Modelando IFE usando ARCADIA/Capella

Taxonomía de diagramas ARCADIA/Capella

En esta sección se proporciona una descripción general de los tipos de diagramas de ARCADIA/Capella. En la imagen 4.1, muestra la taxonomía de diagramas de ARCADIA/Capella. El lenguaje de modelado ARCADIA caracteriza los diagramas en siete tipos diferentes, donde cada tipo de diagrama incluye diagramas que se nombran de manera diferente para los diferentes niveles de ARCADIA o posen el mismo nombre en todos los niveles. A continuación, se ofrece una breve explicación de todos los tipos de diagramas:

1. *Diagramas de flujo de datos*: se proporcionan en todos los niveles (fases) en ARCADIA y se utilizan para representar la red de dependencia de información entre funciones.

2. *Diagramas de arquitectura*: se proporcionan en todos los niveles de ARCADIA y muestran principalmente la asignación de funciones a los componentes.

3. *Diagramas de escenarios*: describen el flujo secuencial de mensajes que se transmiten entre varios elementos (funciones, componentes, estados, entre otros) representados a través de líneas de vida verticales.

4. *Diagramas de modo y estado*: se utilizan para modelar las máquinas de modo y estado en varios niveles de abstracciones. Los modos o estados se pueden modelar para el sistema y sus componentes.

5. *Diagramas de desglose*: se proporcionan en todas las fases de ARCADIA y se utilizan para representar jerarquías funcionales y de componentes.

6. *Diagramas de clases*: se utilizan para modelar estructuras de datos en todos los niveles de ARCADIA y para definir varios tipos de intercambios entre elementos del modelo.

7. *Diagramas de capacidad*: se utilizan para definir capacidades y sus relaciones con los actores y entidades, principalmente durante las fases operativas y de análisis del sistema.

8. *Diagramas de interfaz*: definen las interfaces contextuales internas y externas del sistema y sus componentes.

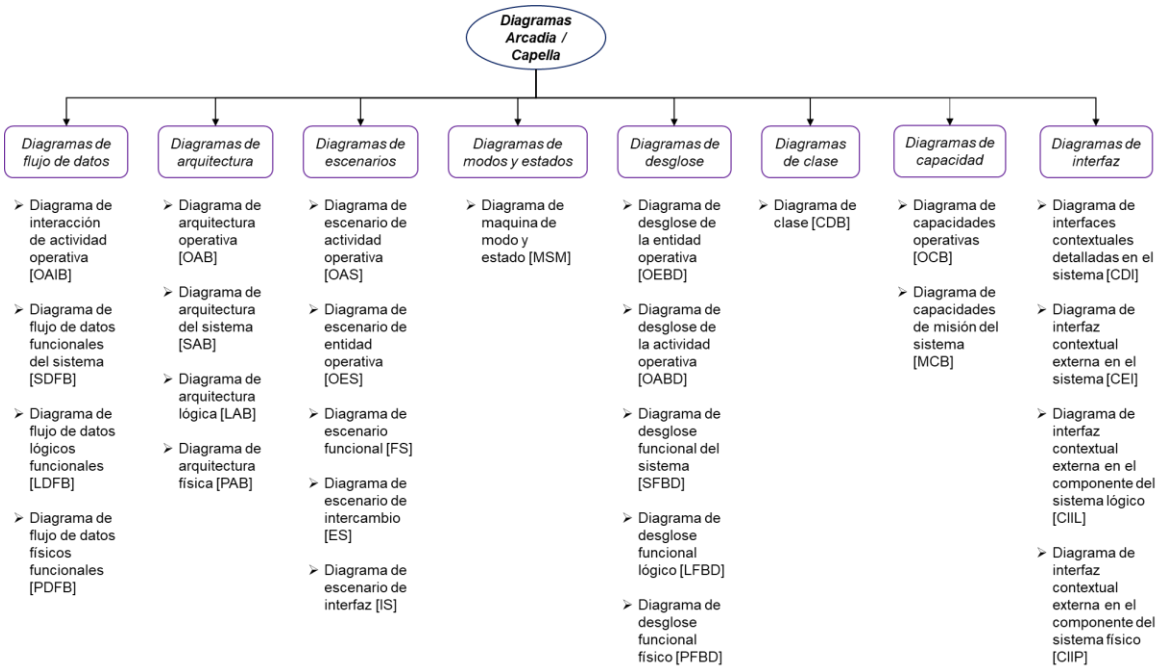


Imagen 4.1. Taxonomía de diagramas de ARCADIA/Capella.

4.1.1. Análisis operacional

El método ARCADIA se centra en el modelado basado en funciones en lugar de los requisitos y trata de vincular los requisitos funcionales a las funciones. El metamodelo de Capella permite la creación de elementos de requisitos y vincularlos a los artefactos en cualquier diagrama de Capella.

Análisis operacional de ARCADIA (OA)

Entre los cuatro niveles (*capas*) de abstracción del método ARCADIA, el primer nivel es el *Análisis Operacional* (***OA***). El análisis operacional es un medio para capturar lo que los usuarios del sistema deben lograr como parte de su trabajo o misión, y las condiciones asociadas, independientemente de cualquier solución, y particularmente de los sistemas que podrán utilizar para este propósito.

Las expectativas del usuario final, el contexto de su trabajo, las condiciones y las limitaciones a menudo no están suficientemente expresadas por los requisitos de las partes interesadas. Por lo tanto, el análisis operacional se realiza antes o junto con el análisis funcional del sistema.

El análisis operacional de la metodología ARCADIA permite al arquitecto del sistema modelar las capacidades operativas de alto nivel requeridas de la misión y realizar un mejor análisis de las necesidades operativas sin siquiera definir el sistema de interés en primer lugar. Esto ayuda a comprender lo que realmente quiere el usuario y luego decidir cuál podría ser el sistema optimo, para identificar los desafíos de las partes interesadas y comprender mejor cómo el sistema puede respaldarlos o brindar soluciones. ARCADIA proporciona un nivel adicional opcional de abstracción que permite encontrar alternativas para definir cuál podría ser realmente el sistema de interés.

El metamodelo de Capella también distingue entre una capacidad operativa y una capacidad del sistema. Una o más capacidades del sistema contribuyen a una capacidad operativa, que son dos elementos diferentes en Capella. Se hace una diferenciación similar para los elementos estructurales y de comportamiento en todos los niveles en Capella. Finalmente, los requisitos de las partes interesadas se definen, refinan y vinculan a los artefactos del modelo necesarios para garantizar la trazabilidad entre los niveles. Los principales pasos de esta fase son:

- Definir actores y entidades operativas
- Identificar capacidades operativas
- Describir actividades operativas y escenarios de capacidades
- Definir modos y estados operativos
- Asignar actividades a entidades y actores operativos

Definir actores y entidades operativas

Los pasos mencionados anteriormente no exigen un orden especifico de actividades a realizar. De hecho, la metodología ARCADIA es bastante flexible en términos del flujo de trabajo de modelado. Durante su ciclo de vida, el sistema interactuará con varios actores y entidades durante su operación. Por lo tanto, se deben identificar todas las entidades operativas y los actores que podrían interactuar con el sistema para comprender mejor los problemas e identificar los requisitos operativos del sistema.

El diagrama de desglose de entidades operativas [OEBD], enumera todos los actores y entidades que interactúan con el sistema IFE, se muestra en la imagen 4.2.

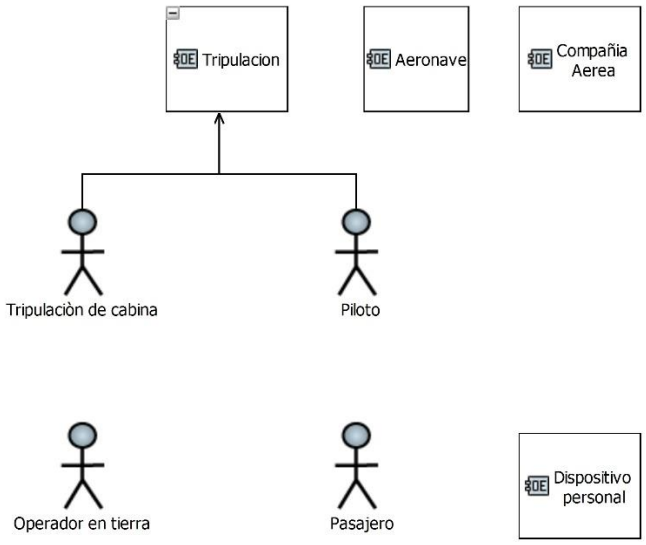


Imagen 4.2. Diagrama de desglose de las entidades operativas del "Sistema IFE" [OEBD].

Identificar capacidades operativas

En Capella, las capacidades de nivel operativo se pueden definir utilizando un elemento del modelo de capacidad operativa. Se espera que las entidades operativas proporcionen una capacidad, denominada "capacidad operativa", que brinde un servicio para cumplir una o más misiones del sistema. Una capacidad de nivel operacional refina las necesidades operacionales y poder ser descrita por varias actividades operacionales que son realizadas por un actor operacional o una entidad operacional. Los escenarios de nivel operativo representan la secuencia de flujo de actividades entre los actores y entidades de manera oportuna para describir escenario de capacidad particular.

En la imagen 4.3, muestra el diagrama en blanco de capacidades operativas [OCBD] en Capella, que considera una capacidad de alto nivel como "Entretenimiento durante el vuelo".

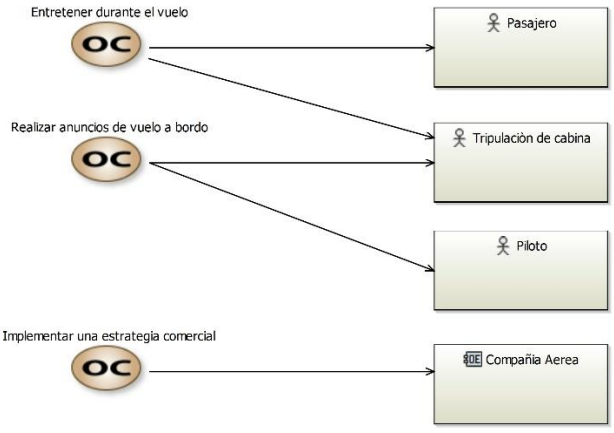


Imagen 4.3. Capacidad operativa de Capella mediante el diagrama en blanco de capacidad operativa [OCBD].

Describir actividades operativas y escenarios de capacidades

Una capacidad operativa puede describirse mediante múltiples escenarios de comportamiento. Los escenarios operativos se utilizan para definir las actividades y la secuencia de sus interacciones a realizar por cada actor y entidad durante el escenario. Alternativamente, una capacidad puede describirse mediante un diagrama en blanco de interacción de actividad operativa [OAIB] para representar las interacciones de actividad para una capacidad particular, sin considerar su asignación a los actores y entidades.

En la imagen 4.4, muestra el diagrama en blanco de interacción de actividad operativa [OAIB] para la capacidad "Ver la película" que se incluye en el nivel superior.

Imagen 4.4. Operación "Ver la película" modelada mediante el diagrama en blanco de interacción de actividad operativa [OAIB].

Definir modos y estados operativos

Una entidad operativa o un actor operativo puede tener varios estados y modos de operación que pueden describirse mediante un diagrama de máquina de modo y estado [MSM]. La imagen 4.5, muestra un diagrama [MSM] para los estados del "Vuelo de aviones".

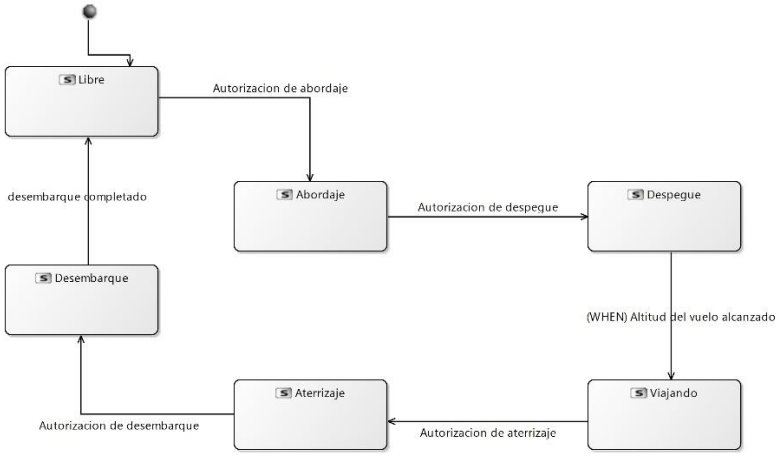


Imagen 4.5. "Estados operativos del avión" mediante el diagrama de máquina de modo y estados [MSM].

Capella proporciona modos y estados de elementos de modelo distintos para modelar modos y estados respectivamente, por consecuente, un modo es un comportamiento esperado del sistema, un componente y también un actor o entidad operativa, en algunas condiciones elegidas, mientras que el estado es un comportamiento que experimenta el sistema, un componente, un actor o entidad operativa, en unas condiciones impuestas por el entorno. En

la imagen 4.6, muestra un diagrama de máquina de modo y estado [MSM] para los modos de "funcionamiento del sistema IFE".

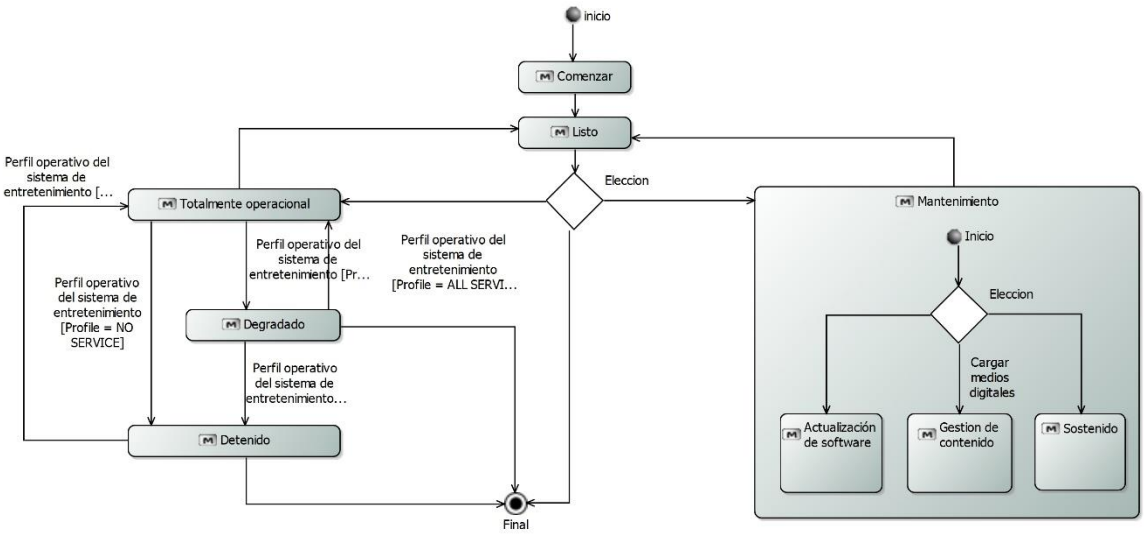


Imagen 4.6. "Modos de funcionamiento del sistema IFE" mediante el diagrama de máquina de modo y estados [MSM].

Asignar actividades a entidades y actores operativos

Uno de los principales resultados de la fase de análisis operativo, es un diagrama en blanco de la arquitectura operativa [OAB] que describe la arquitectura operativa del sistema esperado. En la imagen 4.7, se modelan las actividades previstas de alto nivel para el pasajero, el piloto y la tripulación para lograr un objetivo deseado.

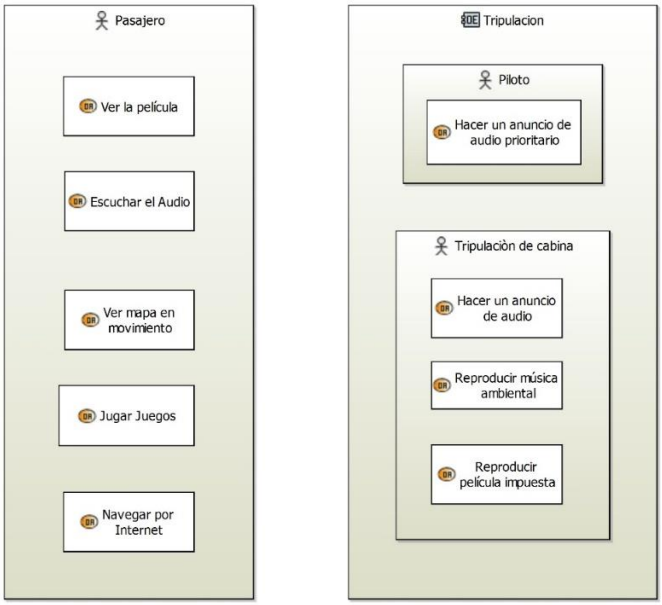


Imagen 4.7. Actividades previstas de alto nivel para los actores del sistema IFE mediante un diagrama de arquitectura operacional [OAB].

En la imagen 4.8, se modelan el pasajero, el sistema de la aeronave y sus entidades y el entorno que incluye a los actores y entidades primarios que interactuarían entre sí para lograr un objetivo deseado, donde se definen y asignan sus respectivas actividades operativas. Un análisis operativo sirve para comprender las necesidades del usuario y estas mismas necesidades se convierten en requisitos que son verificados por un equipo de ingenieros de requisitos y sistemas. Se definen objetivos específicos y se identifican nuevos requisitos durante el proceso de análisis.

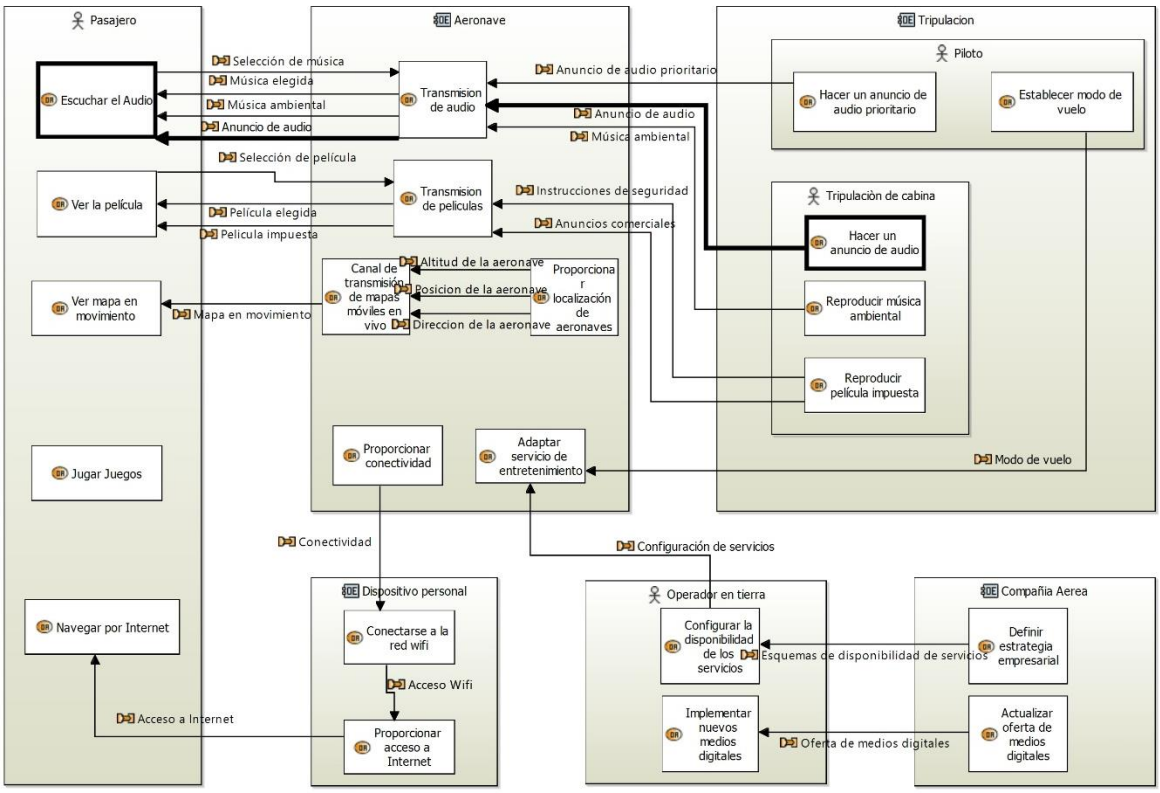


Imagen 4.8. "Arquitectura operacional" usando el diagrama de arquitectura operacional [OAB] de ARCADIA.

4.1.2. Análisis de requisitos del sistema

Como se describió anteriormente, la fase de análisis de requisitos del sistema se ocupa de analizar y refinar los requisitos del sistema.

Análisis del sistema de ARCADIA (SA)

El propósito del análisis de requisitos del sistema es "definir la contribución esperada del sistema a las necesidades de los usuarios, tal como se describen en la fase de análisis operativo y/o en forma de requisitos expresados por el cliente" [9]. La fase SA tiene como objetivo definir 'que' debe lograr el sistema para satisfacer las necesidades del usuario.

El sistema se introduce como una caja negra en esta fase y el análisis de requisitos del sistema es el nivel más bajo de abstracción, perspectiva que ayuda a identificar el contexto, el

comportamiento y la estructura de baja negra del sistema para definir las interfaces externas del sistema y sus interacciones. Los principales pasos en SA son:

- Identificar el contexto del sistema
- Definir las capacidades del sistema y perfeccionar el comportamiento mediante funciones
- Asignar funciones al sistema y los actores
- Definir modos y estados operativos a nivel del sistema
- Definir escenarios del sistema y requisitos de actualización

Identificar el contexto del sistema

En este punto se puede crear un diagrama de contexto del sistema para comprender los límites del sistema y los actores externos que interactúan con el sistema. Esto proporción la información básica necesaria para determinar los servicios solicitados del sistema cuando está integrado en su entorno (capacidades del sistema) sin proporcionar los detalles técnicos de estos servicios.

La identificación de varios actores del sistema puede conducir a discusiones muy fructíferas entre las diferentes partes interesadas del sistema. La imagen 4.9, muestra un diagrama de actores contextuales del sistema [CSA] que representa el contexto operativo del sistema.

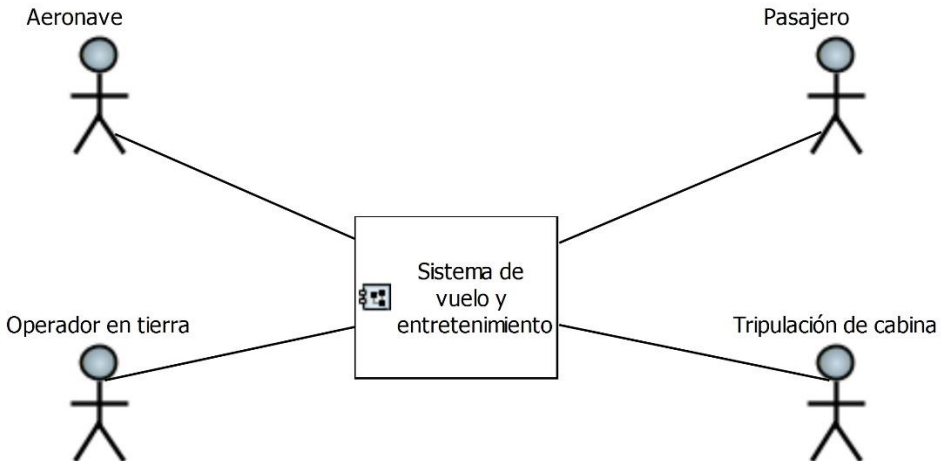


Imagen 4.9. Contexto del sistema IFE.

Definir las capacidades del sistema y perfeccionar el comportamiento mediante funciones

La imagen 4.10, muestra todas las misiones y capacidades del sistema para el sistema IFE. La misión del sistema “Proporcionar soluciones de entretenimiento” aprovecha la capacidad del sistema de “proporcionar de entretenimiento por video”. En Capella, la capacidad del

sistema se elabora mediante un diagrama de flujo de datos del sistema que describe las funciones del sistema y sus intercambios necesarios para proporcionar la capacidad.

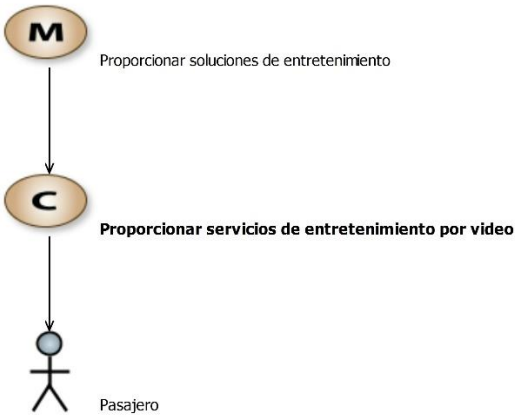


Imagen 4.10. Capacidades del sistema y de misión en Capella usando el diagrama de Capacidades y misiones [MCB].

Al identificar las capacidades del sistema, el flujo de datos de la función a nivel del sistema se puede describir para estas capacidades del sistema. En la imagen 4.11, se muestra todas las misiones y capacidades del sistema IFE, con sus respectivos actores y cuales fueron misiones fueron cubiertas para el desarrollo.

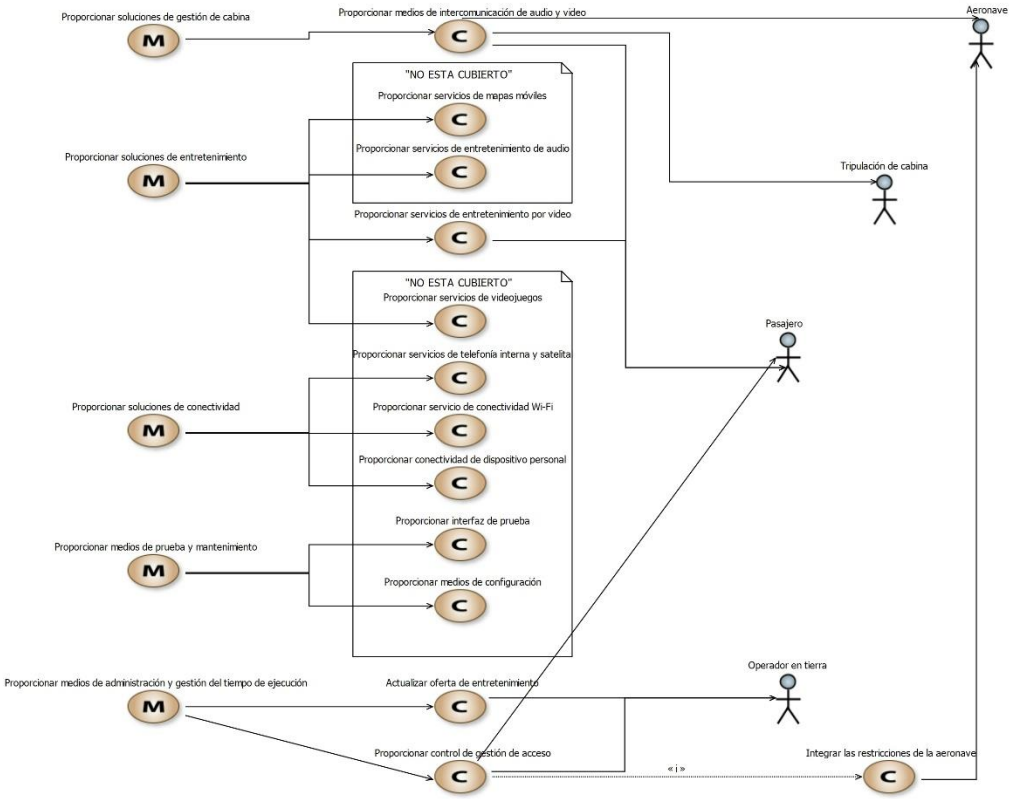


Imagen 4.11. Diagrama de misiones y capacidades del sistema IFE [MCB].

En la imagen 4.12, se muestra la descripción general funcional de alto nivel del sistema IFE donde se muestran las capacidades del sistema y como es la interacción entre ellas.

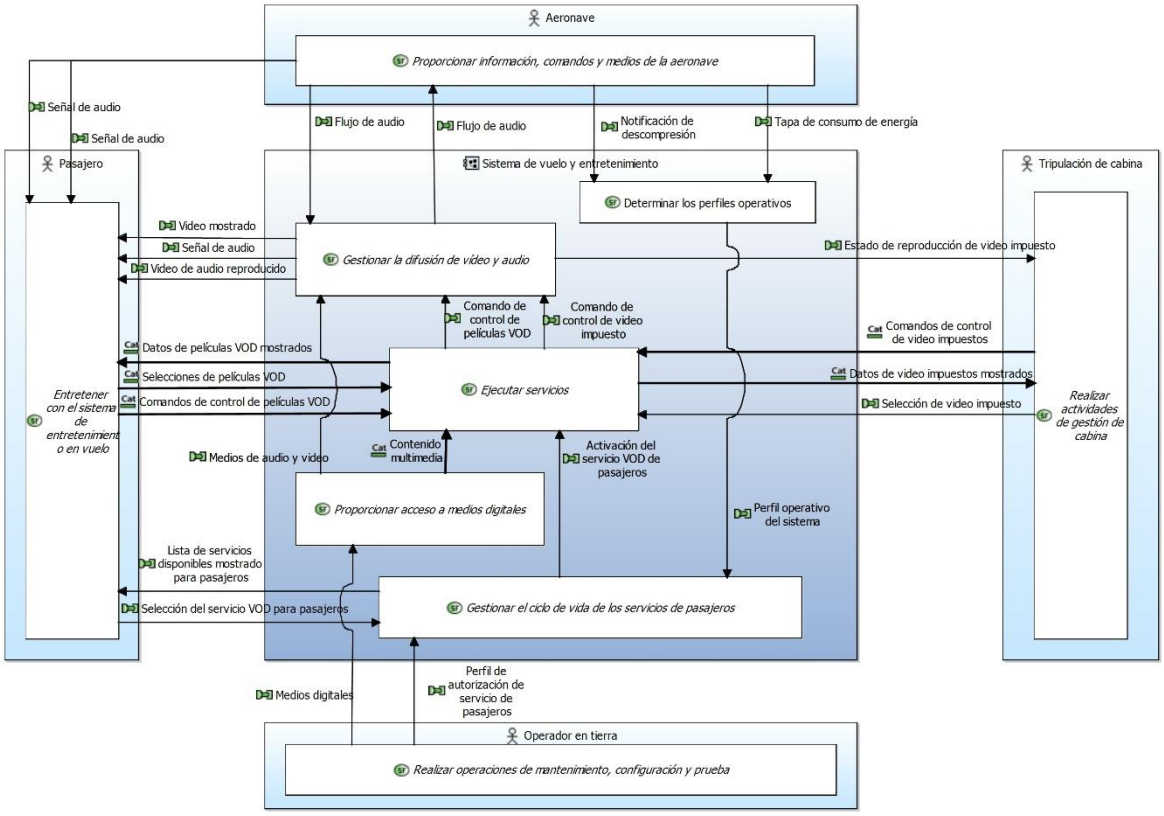


Imagen 4.12. Diagrama de arquitectura del sistema de alto nivel usando el diagrama [SAB].

Capella proporciona varios aceleradores de modelos que ayudan al modelador a reducir el tiempo de modelado al proporcionar una transición automatizada de los elementos del modelo. Uno de esos aceleradores es la transición de la actividad operativa a la función del sistema.

Debido a dicha transición, una capacidad puede definirse mediante un conjunto existente de funciones con flujos de datos que han heredado las relaciones de las actividades operativas y sus interacciones, así como se observa en la imagen 4.13. Los elementos de intercambio de diferentes tipos se pueden asignar fácilmente a intercambios funcionales y puertos de función mediante la creación de elementos de intercambio y tipos de datos en los diagramas de clases de Capella.

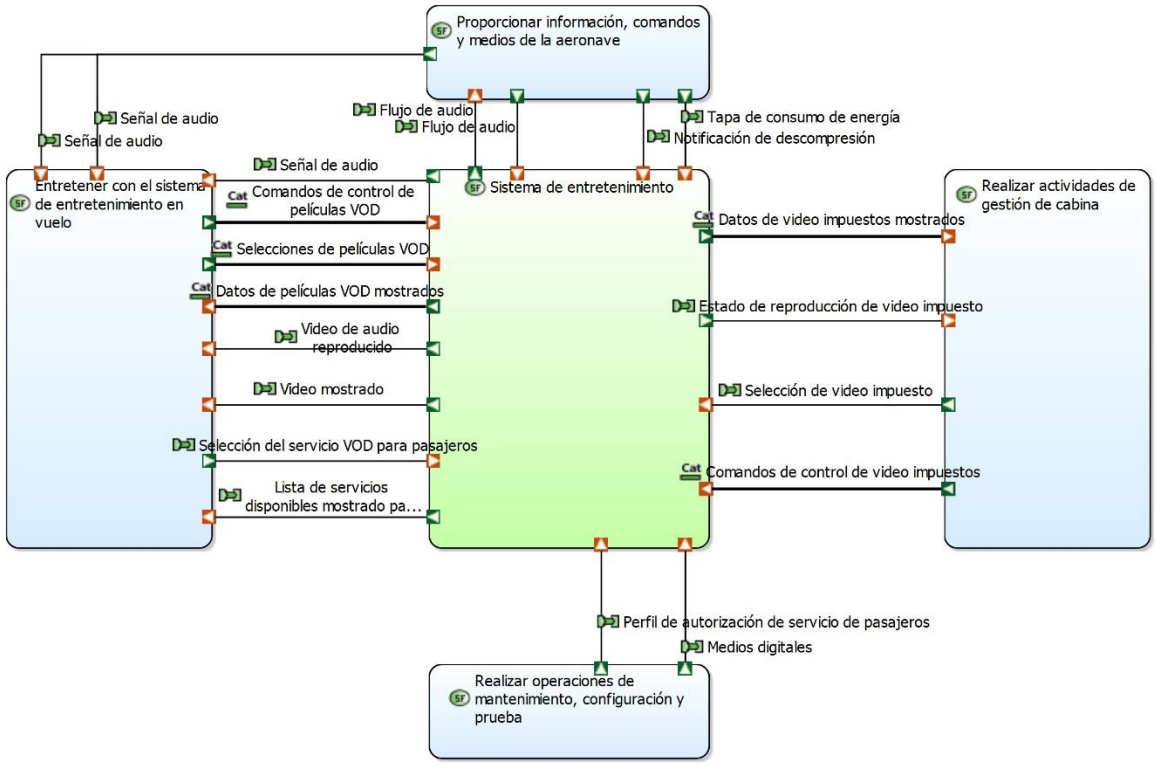


Imagen 4.13. Descripción general funcional de alto nivel usando el diagrama [SDFB].

Además, los actores y entidades operativos pueden convertirse en elementos de actores del sistema. En este punto, una función del sistema (que realiza la actividad operativa) se puede descomponer y asignar al sistema recién creado y sus actores. Si es necesario, se pueden identificar nuevos actores en función de la descomposición de la función del sistema.

Las funciones a nivel del sistema se clasifican como funciones de actor (azul) que deben realizar los actores externos y las funciones del sistema (verde). En el diagrama de flujo de datos del sistema [SDFB], las funciones del sistema deben descomponerse debido a su asignación parcial al sistema.

Por ejemplo, en la imagen 4.14, la actividad operativa “Ver una película” de la imagen 4.4, realizada por el actor operativo “Pasajero” interactúa con la actividad “Mostrar video y reproducir audio”, realizada por el sistema IFE.

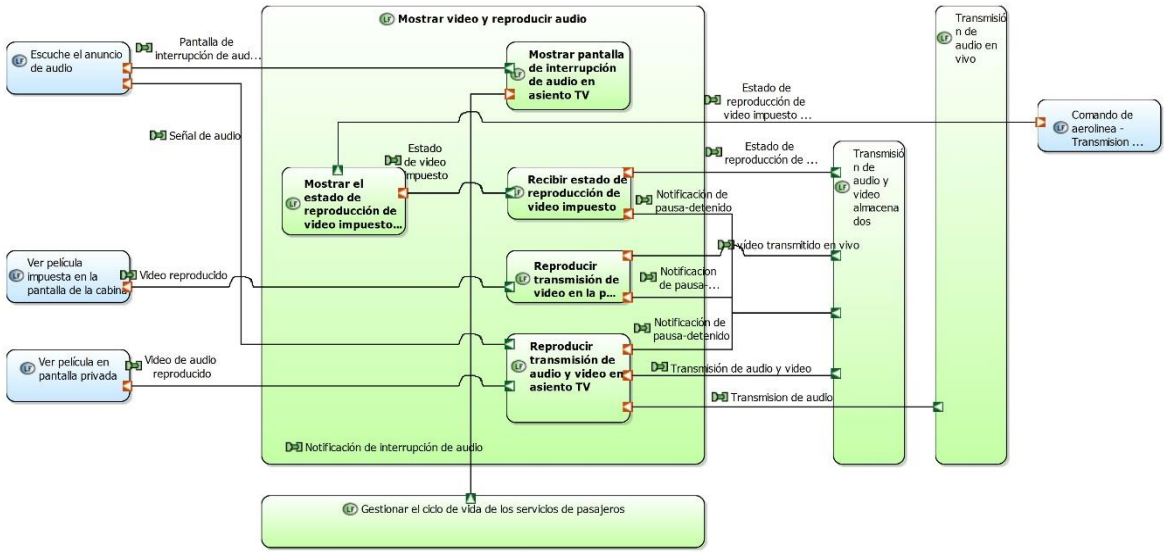


Imagen 4.14. Descomposición funcional a nivel del sistema.

Asignar funciones al sistema y los actores

Una vez identificadas todas las funciones a nivel del sistema, estas funciones se asignan al sistema y a los actores externos para generar una vista de diagrama de la arquitectura del sistema con funciones asignadas. Un arquitecto en sistemas puede preferir modelar los estados a nivel del sistema y los escenarios funcionales para comprender mejor el comportamiento del sistema antes de crear el diagrama de arquitectura del sistema [SAB]. En la imagen 4.15, se muestra el diagrama [SAB] para el sistema IFE.

En el diagrama de arquitectura del sistema [SAB] de la herramienta Capella, las funciones del sistema se pueden asignar a los actores del sistema y al sistema a través de la paleta de diagramas. Los intercambios funcionales ocurren automáticamente después de la asignación de funciones a los actores. Capella proporciona funciones para generar vistas simplificadas y contextuales del diagrama, así como funciones de clonación de diagramas. Las funciones se pueden ocultar para obtener una vista de la arquitectura de “solo componentes” por motivos de simplicidad.

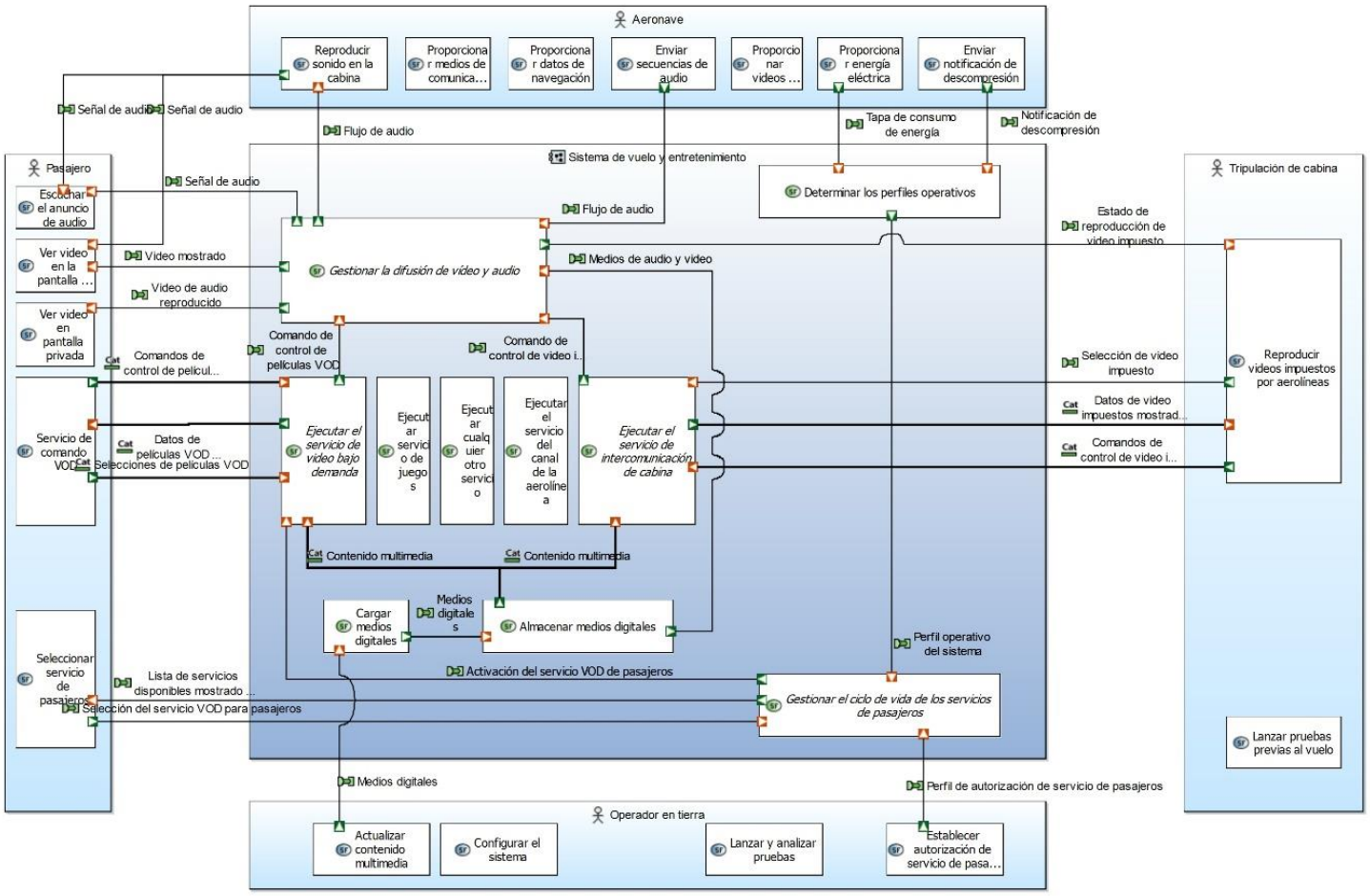


Imagen 4.15. Arquitectura del sistema IFE refinado, usando el diagrama [SAB].

Definir escenarios del sistema

En la imagen 4.16, se muestra el escenario “Iniciar Servicio VOD” que muestra los intercambios entre el sistema y actores. Un escenario puede invocar “subescenarios” que se definen en diferentes diagramas utilizando una referencia insertada entre intercambios sucesivos a lo largo del eje del tiempo. Los modos y estados intermedios también se pueden mostrar en estos diagramas.

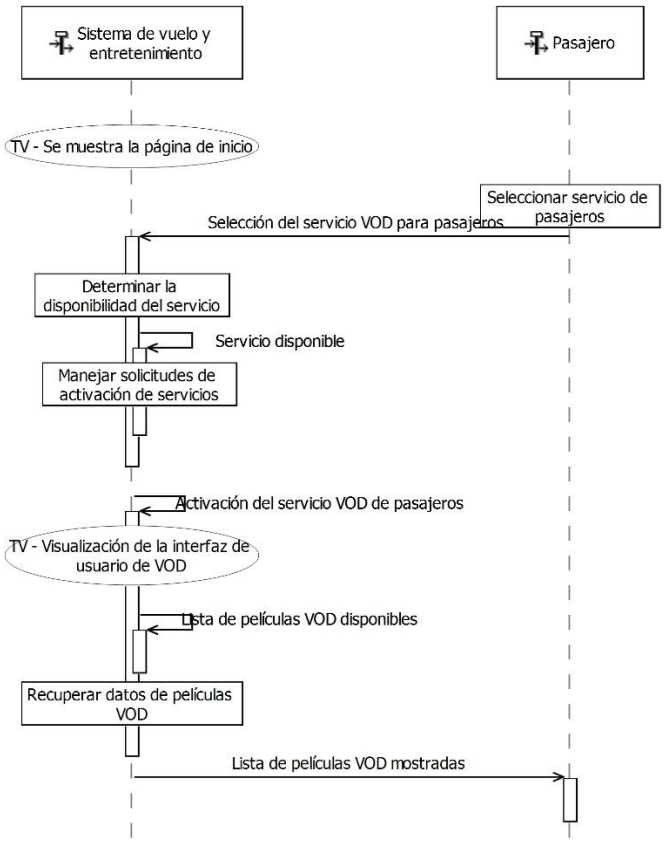


Imagen 4.16. Escenario de intercambio "Iniciar servicio VOD" usando el diagrama de escenarios [ES].

Definir modos y estados operativos a nivel del sistema

De manera similar a los estados operativos, el comportamiento esperado del sistema se puede modelar usando un diagrama de máquinas y estados, particularmente si se supone que el sistema reacciona a eventos de entrada externos. La metodología ARCADIA propone dos conceptos: estado y modo. El uso de estos elementos es una elección metodológica en el contexto de la implementación de MBSE. La imagen 4.17, muestra varios estados exhibidos por el sistema IFE. El establecimiento de estados del sistema es un proceso iterativo y, a menudo, se combina con escenarios para describir escenarios complejos del sistema.

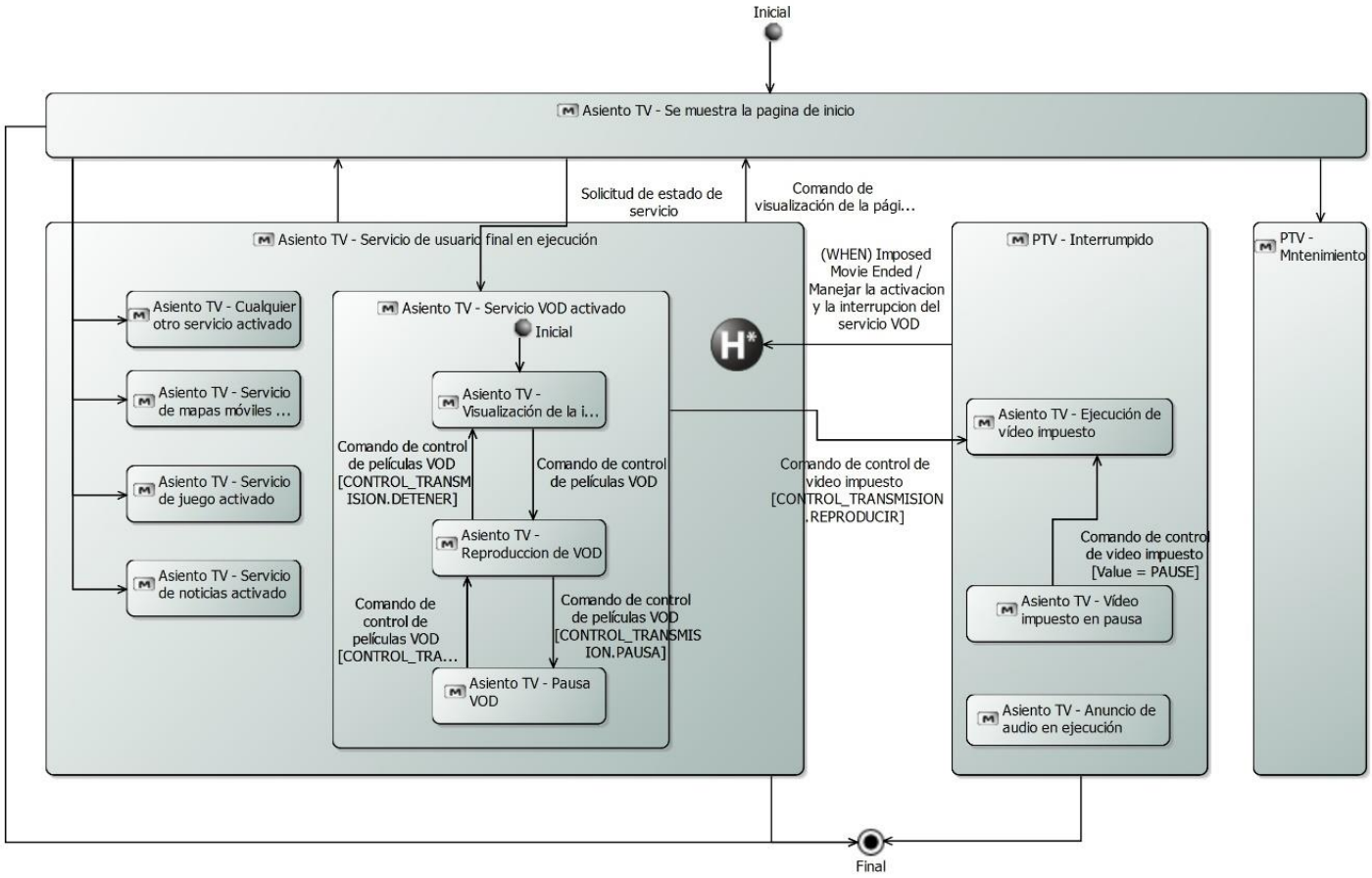


Imagen 4.17. Máquina de modo y estados del sistema IFE usando el diagrama [MSM].

4.1.3. Definición de la arquitectura lógica

La fase de definición de la arquitectura lógica tiene como objetivo capturar los principales impulsores arquitectónicos de la solución, identificando los componentes de alto nivel y su funcionalidad esperada, proporcionando una visión de cómo funciona el sistema, sin sumergirse demasiado en los detalles técnicos.

Arquitectura lógica de ARCADIA (LA)

La fase de diseño de arquitectura lógica de ARCADIA permite encontrar "como" el sistema realizará sus funciones definidas en la fase de análisis del sistema. La fase de arquitectura lógica incluye actividades importantes como definir los factores que impactan la arquitectura y los puntos de vista del análisis, definir los principios subyacentes al comportamiento del sistema, construir alternativas de arquitectura basadas en componentes y seleccionar la mejor arquitectura entre los candidatos. Los principales pasos de esta fase son:

- Identificación de componentes lógicos
- Descomposición de funciones lógicas
- Definición de escenarios y estados de componentes lógicos
- Asignar funciones lógicas a componentes

Identificación de componentes lógicos

Los componentes lógicos del sistema se identifican en función de las funciones asignadas al sistema. El enfoque para obtener una arquitectura lógica final se puede tomar varias maneras como:

1. Refinar las funciones del sistema en funciones de solución y luego agrupar estas funciones en componentes de modo que se mantenga la trazabilidad automática

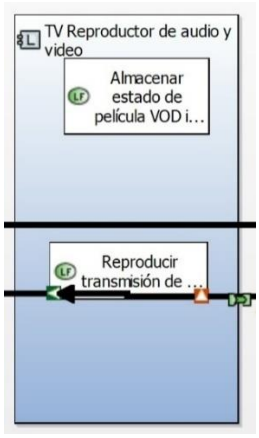
2. Definición de un desglose orientado a soluciones de funciones lógicas en función de la cual se crea la trazabilidad manual entre el sistema y las funciones lógicas

3. Asignar funciones del sistema a componentes lógicos y luego refinar las funciones hasta funciones de solución

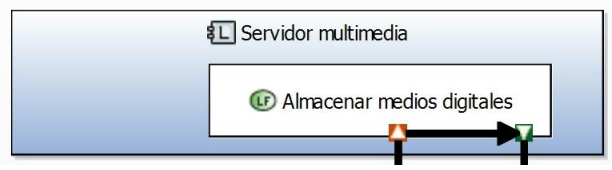
Las funciones del sistema identificadas en el diagrama de flujo de datos del sistema [SDFB] deben asignarse a varios componentes lógicos que satisfagan la realización de funciones lógicas de las funciones del sistema junto con la realización de los intercambios funcionales.

Las transiciones son aceleradores que se desarrollan para facilitar las tareas del modelador. Sin embargo, el modelador siempre tiene la opción de crear una trazabilidad manual y confiar en la validación del modelo para verificar la integridad de la trazabilidad.

Similar al análisis funcional del sistema, el análisis funcional lógico se ocupa de identificar varias funciones de nivel lógico del sistema mediante el diagrama de flujo de datos lógicos [LDFB]. Se define una arquitectura lógica basada en funciones lógicas y luego se realiza la asignación funcional. Por ejemplo, en la imagen 4.18, las funciones *´Almacenar estado de película VOD´* y *´Reproducir transmisión de película´* están asignadas al componente del sistema *´TV Reproductor de audio y video´* y la función *´Almacenar medios digitales´* está asignada a ´Servidor multimedia´. De esta forma, todas las funciones lógicas del sistema se asignan a todos los componentes lógicos identificados.



a) TV Reproductor de audio y video



b) Servidor multimedia

Imagen 4.18. Asignación funcional a componentes lógicos.

Descomposición de funciones lógicas

Una vez que se identifican las funciones lógicas, se puede realizar una descomposición funcional para identificar las funciones de los componentes internos. La descomposición funcional generalmente conduce a múltiples subfunciones que se pueden realizar. Por ejemplo, en la función ***´Transmisión de audio y video almacenados´*** se puede dividir en dos funciones, a saber, *´Transmitir la transmisión´* y *´Preparar transmisiones´*, ya que se pueden usar múltiples servicios para realizar la transmisión de audio y video. En base a esto, se pueden identificar dos nuevos componentes lógicos, a saber, ***´TV´*** y ***´Servidores´***. La imagen 4.19 muestra la descomposición funcional realizada en Capella.

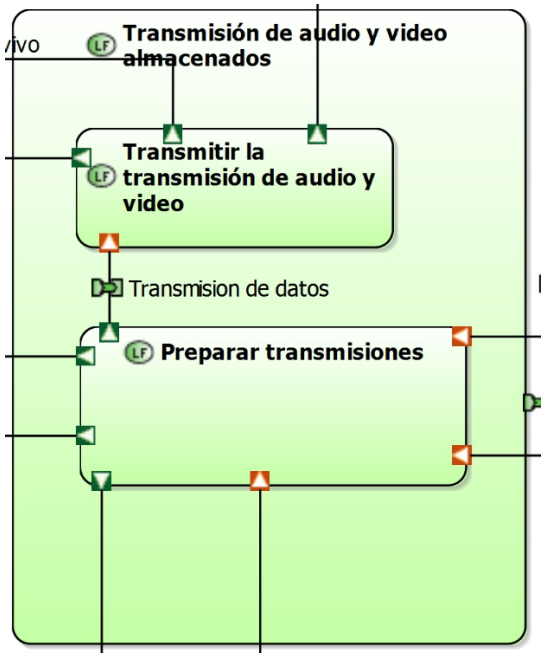


Imagen 4.19. Descomposición de funciones lógicas.

Identificar escenarios y estados de componentes lógicos

Con base en los escenarios del sistema, se pueden modelar escenarios lógicos para mostrar los intercambios secuenciales entre funciones y componentes lógicos. Nuevamente, los tres tipos de escenarios, principalmente funcional, de intercambio e interfaz, se pueden modelar en el nivel lógico. El propósito de los escenarios fue principalmente refinar los escenarios creados en la perspectiva SA y comprender el comportamiento. Los modos y estados del sistema permanecen prácticamente sin cambios, sin embargo, la asignación funcional a los modos y estados con las condiciones de transición son de interés clave.

Asignar funciones lógicas a componentes

La arquitectura lógica de Capella proporciona una vista de diagrama para mostrar las asignaciones funcionales a los componentes utilizando el espacio en blanco de arquitectura lógica [LAB]. Además, las funciones de asignación se pueden ejecutar con un simple

comando de arrastrar y soltar. Los intercambios entre funciones se pueden asignar fácilmente a los intercambios de componentes que definen las interfaces.

Permite mostrar los intercambios funcionales y de componentes simultáneamente en el mismo diagrama, la asignación entre funciones y puertos de componentes, entre otras, lo que permite la justificación de las interfaces de los componentes en función del contenido funcional. Además, se muestra una vista simplificada y contextual de los componentes y funciones utilizando varios diagramas en Capella. La imagen 4.20, muestra el diagrama de arquitectura lógica [LAB] para el sistema de entretenimiento, modelado en Capella que muestra los componentes lógicos.

Algunos de los elementos se ocultan con las funciones de Capella para resaltar solo los componentes de la arquitectura lógica del sistema. La generación de vistas simplificadas de los elementos contextuales de la arquitectura es un aspecto importante del desarrollo de la arquitectura. Capella proporciona la función para generar elementos contextuales utilizando una serie de diagramas. Sin embargo, debido al menor nivel de complejidad de la arquitectura del sistema de este estudio, solo se muestra una única vista de arquitectura de la arquitectura lógica.

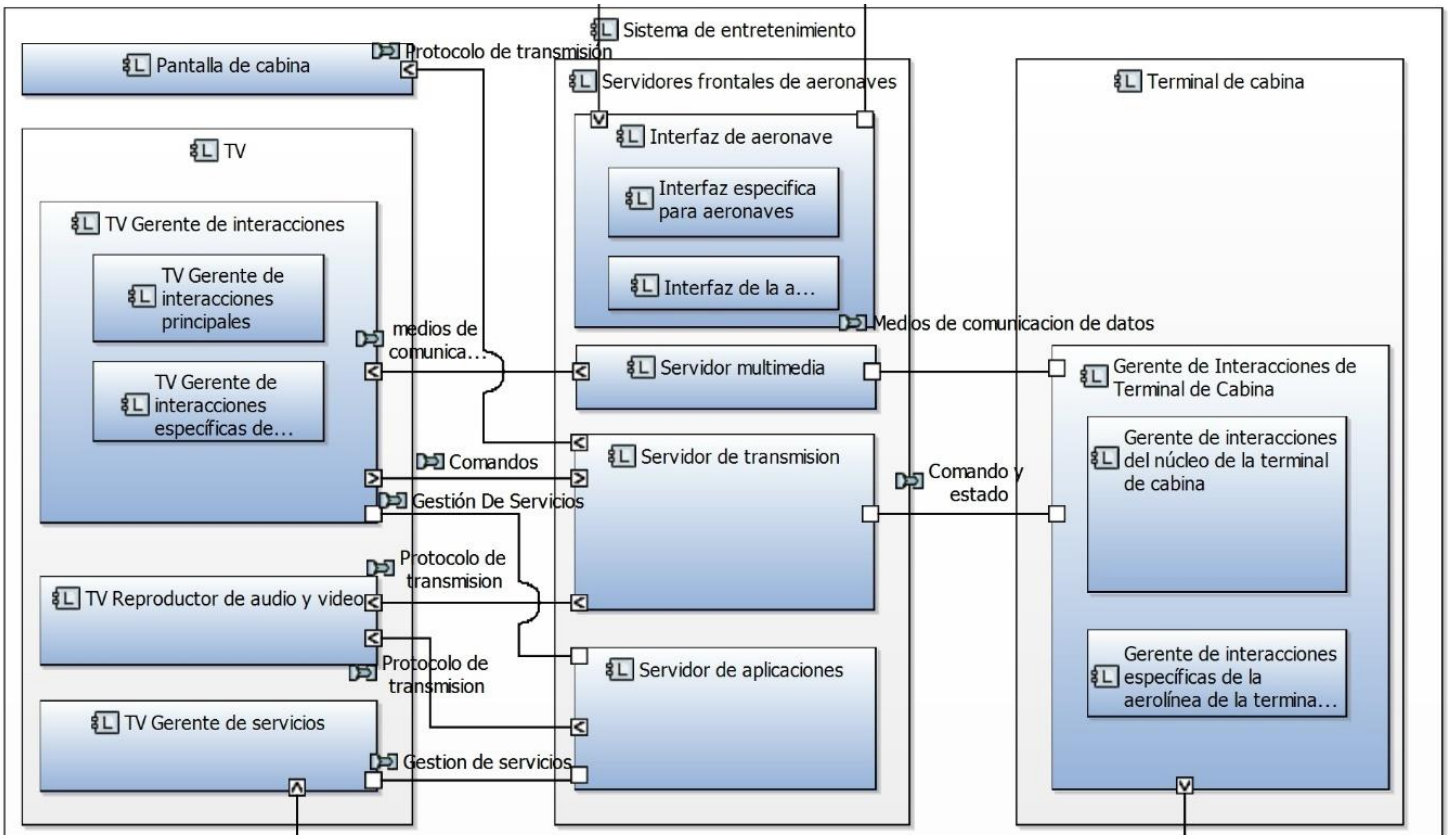


Imagen 4.20. Arquitectura lógica del sistema de entretenimiento de componentes lógicos.

4.1.4. Definición de la arquitectura física

El propósito de la fase de definición de la arquitectura física es desarrollar una arquitectura que consta de componentes físicos dependientes de la tecnología o partes necesarias para cumplir las funciones de los componentes lógicos.

Arquitectura física de ARCADIA (PA)

El flujo de modelado seguido en la fase PA es nuevamente arbitrario, similar a OA, SA y LA. Similar a la fase LA, los componentes físicos del sistema se identifican en función de las funciones lógicas asignadas a los componentes del sistema. La notación a Capella, proporciona los conceptos de componentes físicos mediante el uso de dos elementos: ***componentes de comportamiento***, que realizan las funciones delegadas al sistema, y componentes físicos de alojamiento (nodo), que albergan los componentes de comportamiento y les proporcionan los recursos necesarios para funcionar. Un comportamiento o un componente de nodo puede ser un hardware o un software. Los componentes del nodo se vinculan mediante un enlace físico, para especificar los medios de comunicación entre dos componentes para respaldar los intercambios de comportamiento. Los principales pasos de esta fase son:

- Definir los componentes físicos del nodo para admitir los componentes de comportamiento
- Asignar los componentes físicos de comportamiento a los componentes del nodo
- Definir las interfaces físicas
- Asignar funciones físicas a los componentes de comportamiento e identificar sub-funciones si es necesario.

Definición de componentes físicos del nodo (PC del nodo)

El enfoque de este paso es definir las PC del nodo para definir múltiples soluciones que reflejen los principios de estructuración. Las PC de nodo se crean en el diagrama de arquitectura física [PAB]. Las PC de nodo están conectadas por enlaces físicos que proporcionan el medio para canalizar los intercambios entre los componentes de comportamiento (por ejemplo, una red cableada o un enlace satelital). A continuación, los intercambios de componentes se asignan a los enlaces físicos respectivos entre los componentes. Como resultado, los intercambios funcionales que se asignan a los componentes se reflejan en los intercambios físicos.

Asignación de comportamiento a componentes físicos (Comportamiento PC)

Las transiciones de Capella realizan automáticamente el comportamiento de las PC a partir de los componentes lógicos. El comportamiento de los PC se define en base a componentes lógicos. Los PC de comportamiento se pueden identificar de forma análoga a la implementada para encontrar los componentes lógicos. Una PC de nodo puede contener una o más PC de comportamiento; sin embargo, de forma predeterminada, un componente de comportamiento solo se puede asignar a un componente de nodo. Antes de finalizar el comportamiento, se realiza la asignación del comportamiento dinámico utilizando

escenarios, estados y modos, entre otros, con la posibilidad de refinar los requisitos y su asignación a los componentes involucrados. Los intercambios de componentes entre los componentes de comportamiento se definen o realizan a partir de SA y se les asignan intercambios funcionales.

Asignación de funciones físicas

Finalmente, se asignaron funciones físicas a los componentes de comportamiento. La evaluación de la arquitectura se puede realizar en ARCADIA agregando restricciones a los elementos del modelo y realizando análisis de puntos de vista especializados. La imagen 4.21 muestra el diagrama de la arquitectura física del sistema IFE donde se muestra un componente de comportamiento para la ´Unidad de visualización de video privada´ con intercambios funcionales, físicos y componentes ocultos.

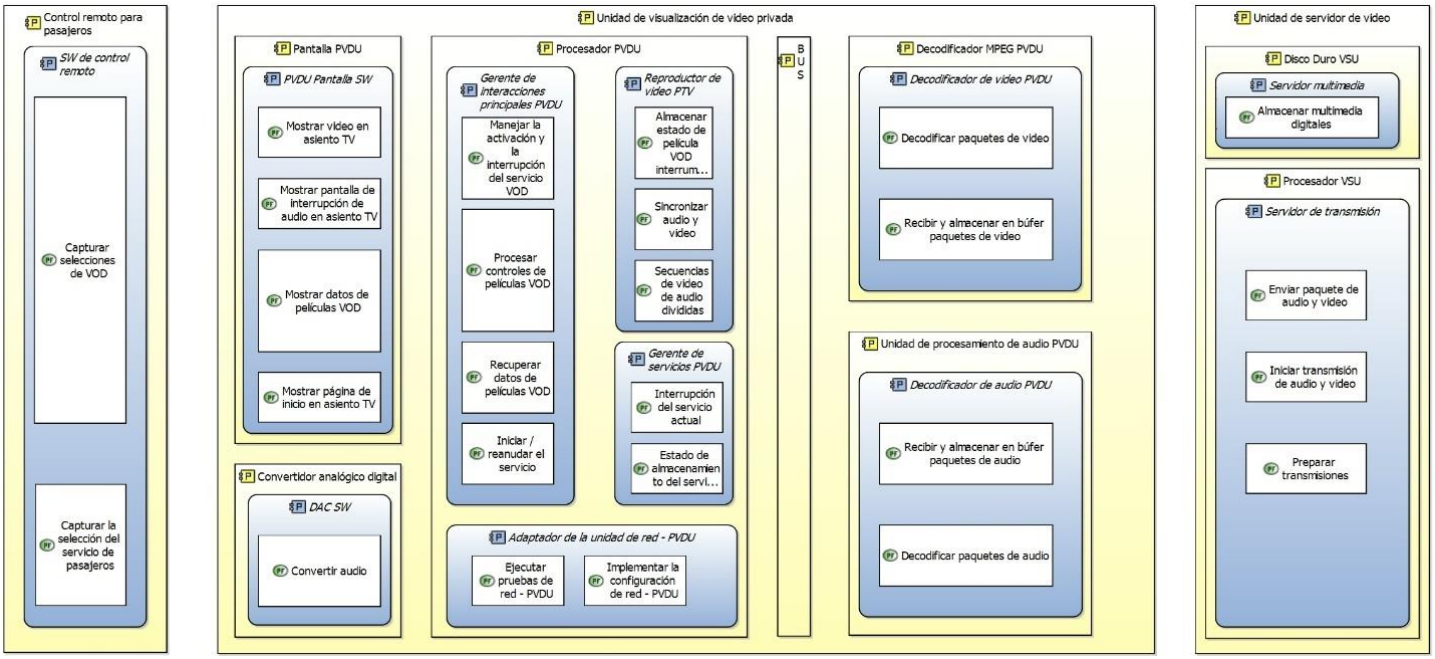


Imagen 4.21. Arquitectura Física de ARCADIA sobre el sistema IFE.

Estructura de descomposición del producto final de ARCADIA (EPBS)

ARCADIA proporciona una capa adicional de EPBS que ayuda a definir "Qué se espera del proveedor de cada componente". Este nivel es mucho más bajo, como se observa en la imagen 4.22, que el OA, SA, LA y PA, en cuanto a las actividades de modelado, conceptos y diagramas como enfoque para detallar la estructura de desglose del producto.

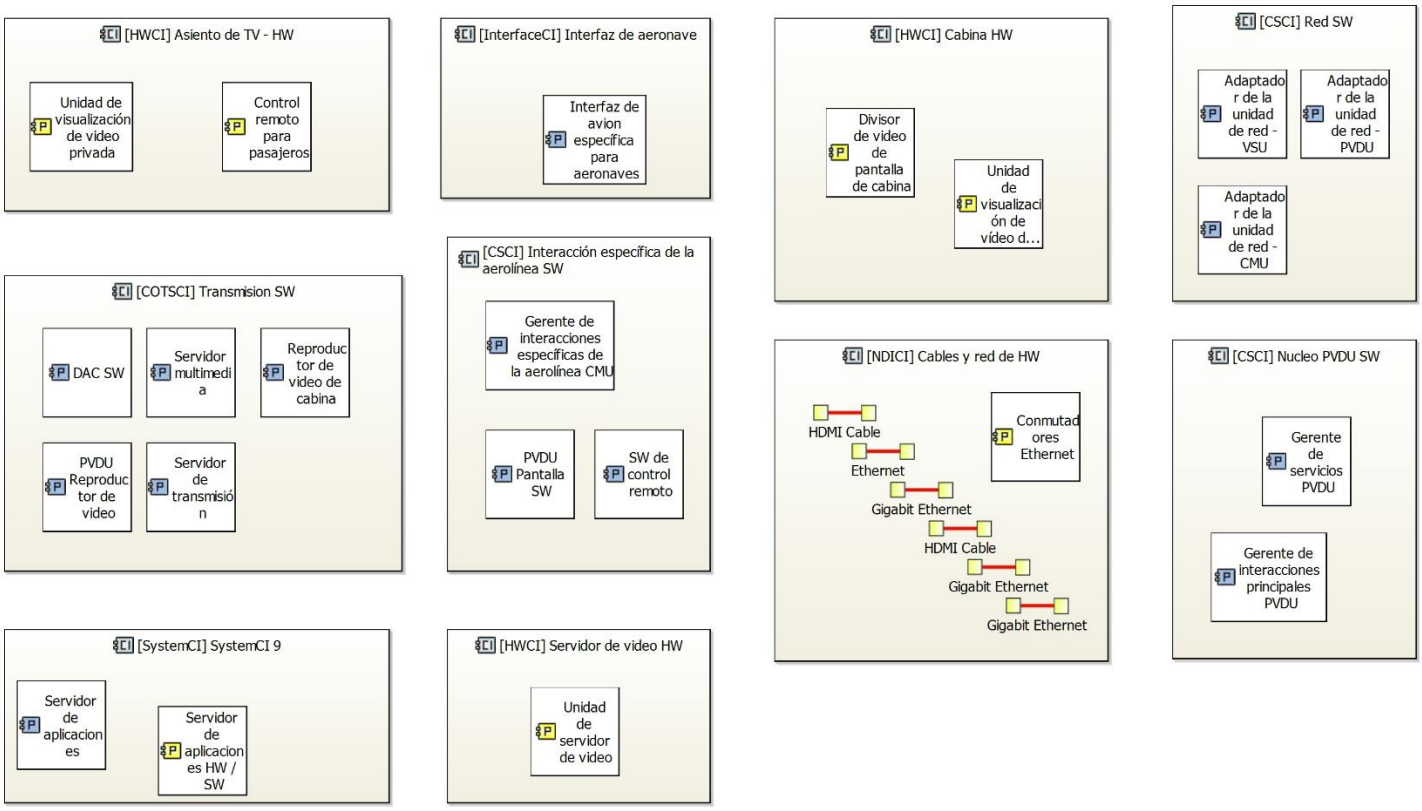


Imagen 4.22. Estructura de descomposición final del sistema IFE.

La imagen 4.23 muestra la pirámide de desarrollo de la arquitectura del sistema que representa los diagramas de arquitectura ARCADIA / Capella creados en los distintos niveles de abstracción.

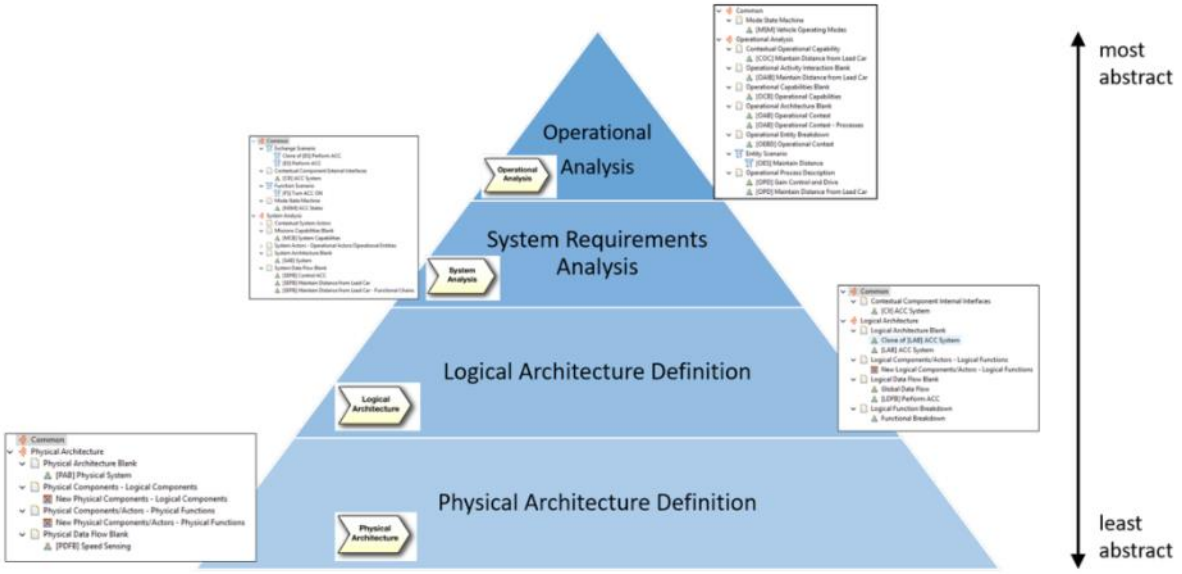


Imagen 4.23. Pirámide de desarrollo de la arquitectura: ARCADIA / Capella.

4.2. Modelando IFE usando AutoFocus3

En esta etapa de desarrollo, se va a tomar como base la estructura de descomposición final del sistema IFE, como se aprecia en la imagen 4.22, por lo que se continua la segunda parte del desarrollo a partir de este diagrama de referencia. Cabe destacar que se toma como ejemplo el caso de uso *"Ejecutar servicios de entretenimiento",* referenciando a la imagen 3.3, del Capítulo 3, justificando los artefactos seleccionados para poder cumplir con el caso de uso. En el **Anexo B**, se muestran el resto de los diagramas generados por la herramienta AutoFocus3.

En la imagen 4.24, se muestran los artefactos que se utilizaron para continuar el desarrollo en la herramienta AutoFocus3 y poder concluir las actividades principales de la MBSE.

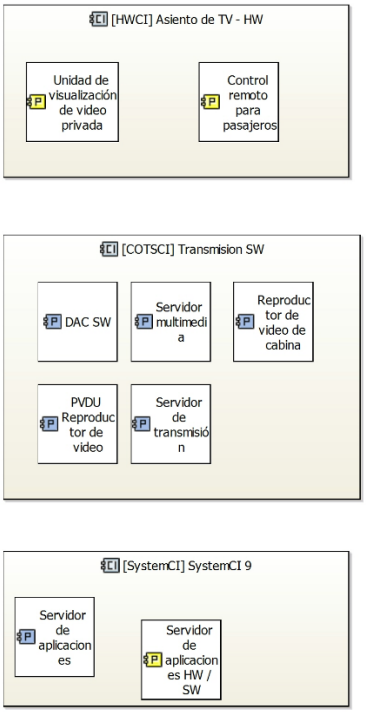


Imagen 4.24. Artefactos seleccionados para concluir el desarrollo en la herramienta AutoFocus3.

Taxonomía de diagramas AutoFocus3

En esta sección se proporciona una descripción general de los tipos de diagramas de AutoFocus3. En la imagen 4.25, muestra la taxonomía de diagramas de AutoFocus3. Se caracteriza los diagramas en cinco tipos diferentes, donde cada tipo de diagrama incluye diagramas que se nombran de manera diferente para los diferentes niveles de AutoFocus3.

1. *Diagramas de ingeniería de requisitos:* es una descripción general de los requisitos del sistema.
2. *Diagramas de modelado de arquitectura:* describe el comportamiento, tipos y funciones de los componentes creados a partir de los requisitos.

3. *Diagramas de simulación:* describe como simular el comportamiento de los componentes y subcomponentes de arquitectura por la configuración automática o manual de valores en los puertos de entrada de los componentes.
4. *Diagramas de arquitecturas técnicas:* describe la estructura y las propiedades de la plataforma de hardware / software del sistema que se está diseñando.
5. *Diagramas de generación de código:* describe el código C a partir de arquitecturas de componentes y una definición de plataforma asociada.

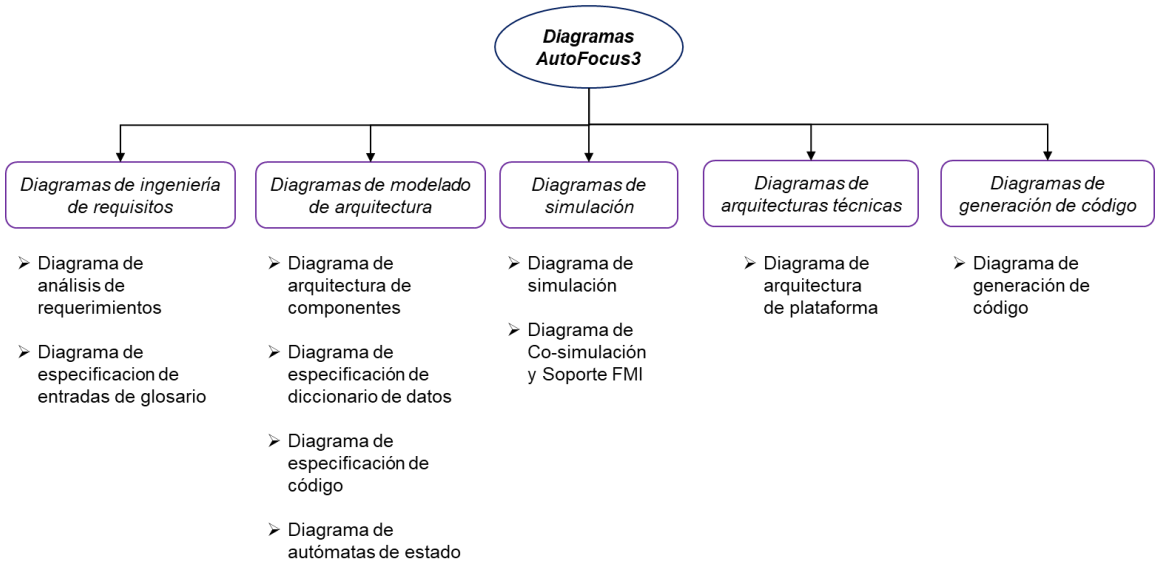


Imagen 4.25. Taxonomía de diagramas de la herramienta AutoFocus3.

4.2.1. Modelado

Arquitectura de componentes

Un componente representa una parte modular de un sistema y su comportamiento se define en términos de interfaces obligatorias y proporcionadas. Un diagrama de componentes representa como un sistema de software es dividido en componentes e incluyen archivos, cabeceras, módulos, entre otros. Debido a que los diagramas de componentes son más parecidos a los diagramas de casos de uso, estos son utilizados para modelar la vista estática y dinámica de un sistema. No es necesario que un diagrama incluya todos los componentes del sistema. En la imagen 4.26, se crearon los componentes con sus entradas y salidas respectivamente.

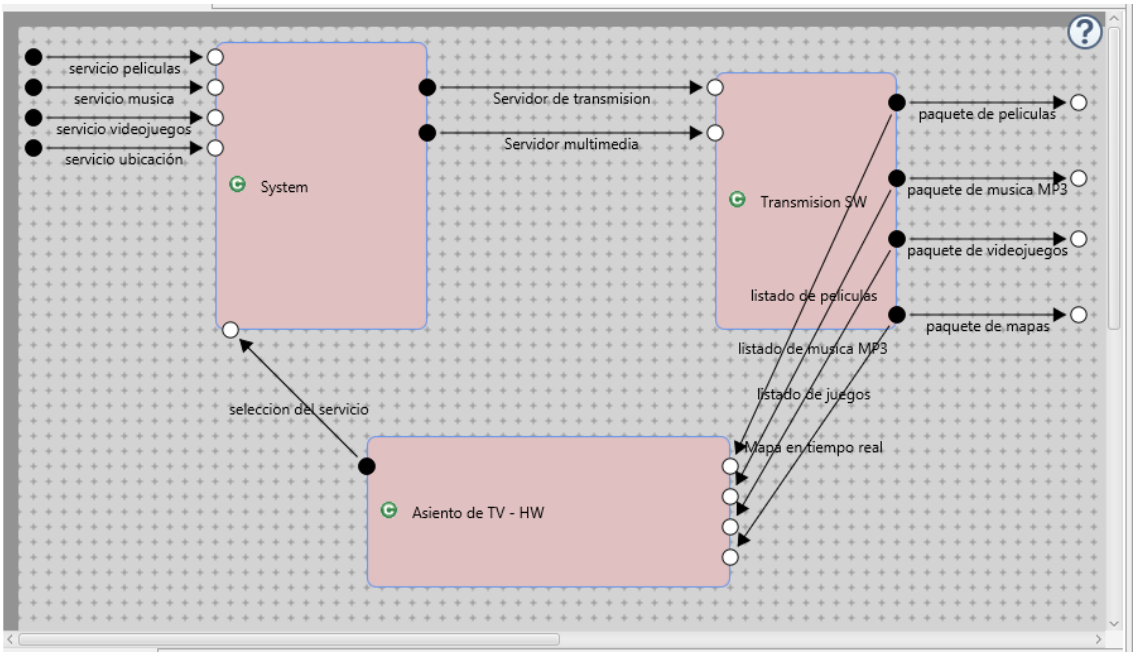


Imagen 4.26. Diagrama de componentes del caso de uso CU1.

Por consecuente se determina el componente fuertemente causal para realizar dichas entradas y salidas sobre la meta de valor que se está buscando. Es por eso que en la imagen 4.27, el componente ***"sistema"*** tiene referencia con el componente ***"Asiento TV",*** porque el componente ***"transmisión"*** está vinculado y es el encargado de provocar las peticiones al componente ***"Asiento TV",*** por consecuente es el componente fuertemente causal.

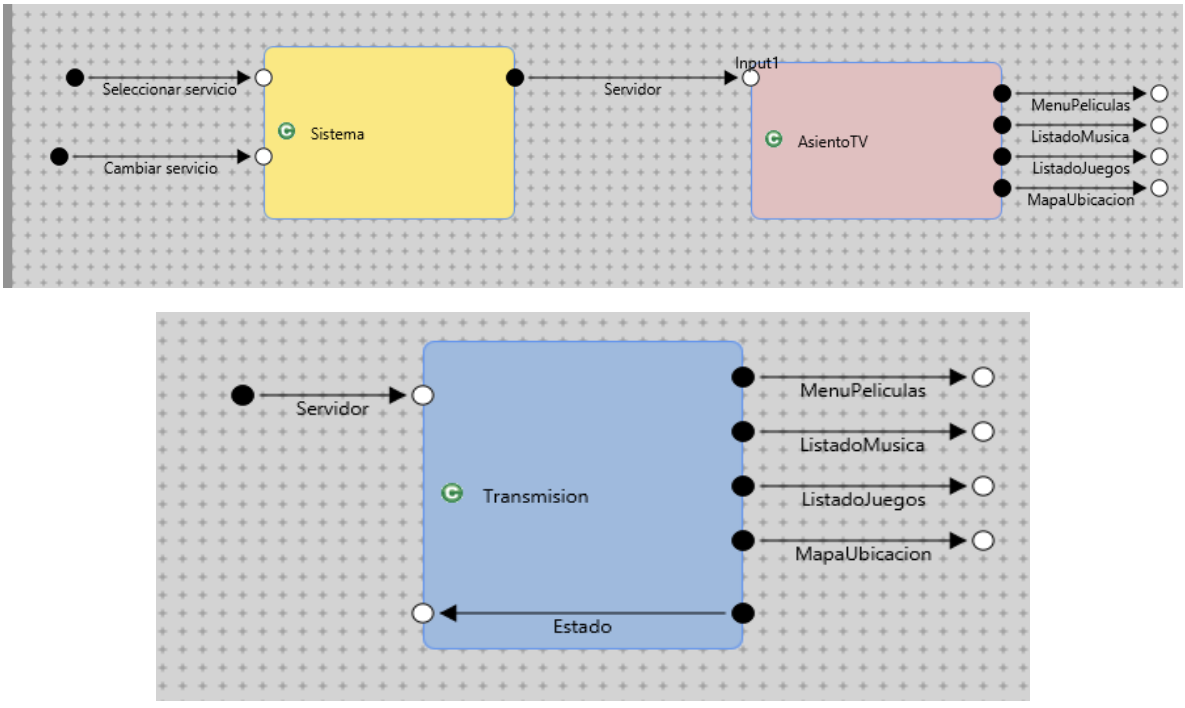
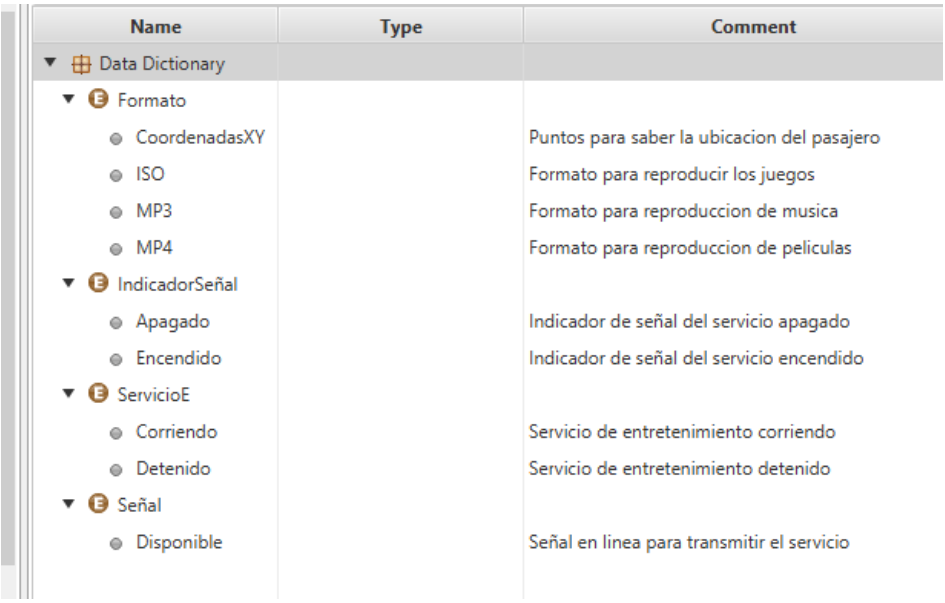


Imagen 4.27. Diagrama de componentes con referencia al Asiento TV y Transmisión.

Especificación de diccionario de datos

Un diccionario de datos es un tipo de metadato que enlista de manera organizada los nombre, definiciones y características de cada uno de los campos o atributos de un conjunto de datos. Tiene por objetivo proveer un lenguaje común entre el autor de dichos datos y sus posibles usuarios. Nos permiten entender e interpretar un conjunto de datos al proporcionar información básica sobre campos o variables que contiene. Una vez que tenemos la arquitectura de componentes dado de alta, se crea un diccionario de datos para que se vinculen a las entradas y salidas de la arquitectura de componentes. En la imagen 4.28, tenemos el nombre de los datos y una pequeña descripción para utilizar en los componentes antes mencionados.



Name	Type	Comment
Data Dictionary		
Formato		
CoordenadasXY		Puntos para saber la ubicacion del pasajero
ISO		Formato para reproducir los juegos
MP3		Formato para reproduccion de musica
MP4		Formato para reproduccion de peliculas
IndicadorSeñal		
Apagado		Indicador de señal del servicio apagado
Encendido		Indicador de señal del servicio encendido
ServicioE		
Corriendo		Servicio de entretenimiento corriendo
Detenido		Servicio de entretenimiento detenido
Señal		
Disponible		Señal en linea para transmitir el servicio

Imagen 4.28. Especificación de diccionario de datos.

Especificación de código

Una especificación de código es la forma de darle vida y lógica a los componentes creados. Una vez dado de alta el diccionario de datos, para poder ser interpretados en los componentes, se crea una especificacion de código del componente “sistema”, para poder interactuar entre si con los demás componentes creados. En la imagen 4.29, se observa la lógica del componente.

Code Specification Editor

```
sistemaOutServidor = NoVal;

if (sistemaInSeleccionar == Disponible() ){
    sistemaOutServidor = Disponible();
    return;
}

if (sistemaInCambiar == Disponible() ){
    sistemaOutServidor = Disponible();
    return;
}
```

Imagen 4.29. Especificación de código del componente "Sistema".

Por consecuente, se realizó la especificación de código al componente "transmisión", donde se utiliza el diccionario de datos e influye directamente en el componente "Asiento TV", como en la imagen 4.30.

Code Specification Editor

```
if (transmisionInEstado == 0) {
    transmisionOutFormato = MP4();
    transmisionOutServicioE = Detenido();
    transmisionOutIndicadorSenal = Apagado();
    transmisionOutEstado = 1;
    return;
}

if (transmisionInEstado == 1) {
    transmisionOutFormato = MP4();
    transmisionOutServicioE = Corriendo();
    transmisionOutIndicadorSenal = Encendido();
    transmisionOutEstado = 2;
    return;
}

if (transmisionInEstado == 2) {
    transmisionOutFormato = MP3();
    transmisionOutServicioE = Corriendo()();
    transmisionOutIndicadorSenal = Encendido();
    transmisionOutEstado = 3;
    return;
}

if (transmisionInEstado == 3) {
    transmisionOutFormato = ISO();
    transmisionOutServicioE = Corriendo();
    transmisionOutIndicadorSenal = Encendido();
    transmisionOutEstado = 4;
    return;
}

if (transmisionInEstado == 4) {
    transmisionOutFormato = CoordenadasXY();
    transmisionOutServicioE = Corriendo();
    transmisionOutIndicadorSenal = Encendido();
    transmisionOutEstado = 5;
    return;
}

if (transmisionInEstado == 5) {
    transmisionOutFormato = CoordenadasXY();
    transmisionOutServicioE = Detenido();
    transmisionOutIndicadorSenal = Apagado();
    transmisionOutEstado = 0;
    return;
}

transmisionOutFormato = NoVal;
transmisionOutServicioE = NoVal;
transmisionOutIndicadorSenal = NoVal;
transmisionOutEstado = transmisionInEstado;


return;
```

Imagen 4.30. Especificación de código del componente "Transmisión".

Como resultado de la especificación de código en todos los componentes creados, automáticamente se despliega un diagrama de componentes más refinado incluyendo el diccionario de datos junto con la especificación de código ya incrustado en los componentes. en la imagen 4.31 y 4.32, se observan los diagramas refinados de dichos componentes.

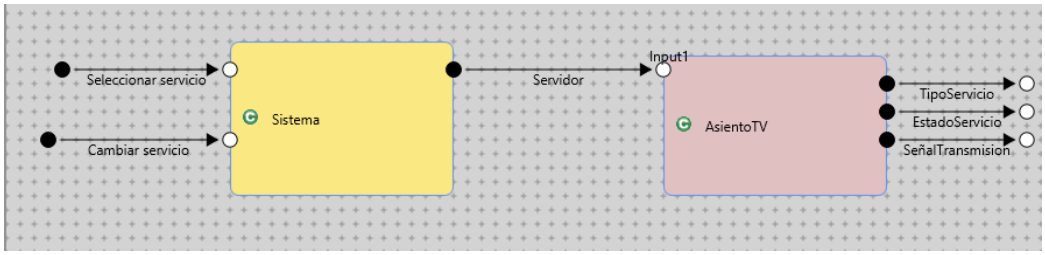


Imagen 4.31. Diagrama de componentes "Sistema" y "Asiento TV", refinado con diccionario de datos incrustado.

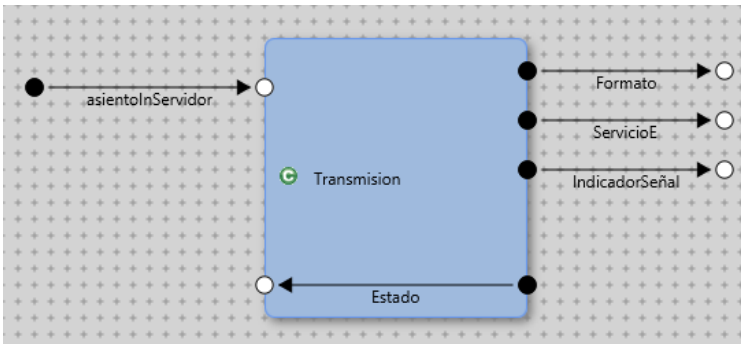


Imagen 4.32. Diagrama de componentes "Transmisión" refinado con diccionario de datos incrustado.

4.2.2. Despliegue y generación de código

Arquitectura de plataforma

Una arquitectura de plataforma describe la arquitectura de hardware y los recursos del sistema. Hay de tres tipos de plataforma que ofrece la herramienta AutoFocus3: *genérica, jerárquica y RaspberryPi.*

Para este desarrollo de tipo de arquitectura de plataforma, se utiliza la arquitectura de plataforma genérica, que es un modelo plano de la plataforma de hardware del sistema, que consiste en una red de unidades de control electrónico (ECU) que están conectadas por medio de unidades de transmisión. Una vez terminado la arquitectura de componentes, se realiza un diagrama de arquitectura de plataforma genérica para conectar los componentes entre sí, como se observa en la imagen 4.33.

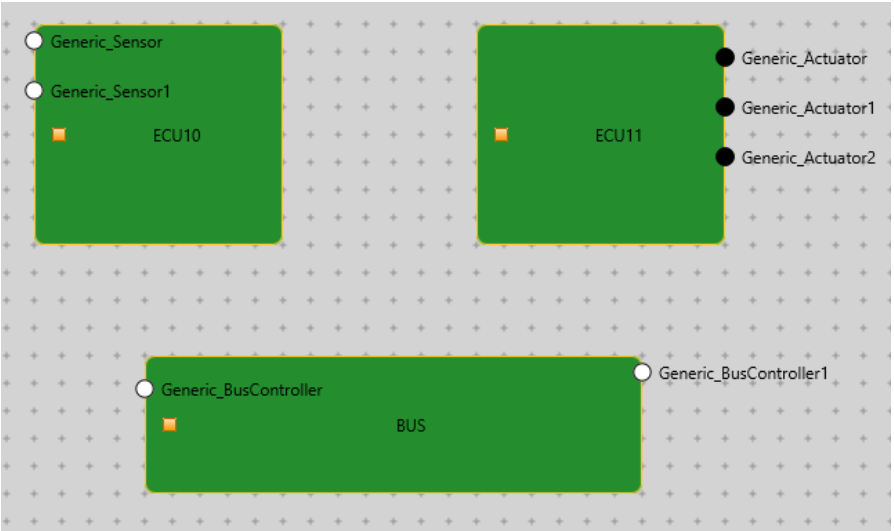


Imagen 4.33. Diagrama de arquitectura de plataforma sin asignaciones.

Lo siguiente en el proceso es realizar asignaciones de la ubicación de los componentes y en el hardware especifico donde serán alojados sus unidades de ejecución. En la imagen 4.34, se despliega los componentes anteriormente creados con las respectivas unidades de control (ECU) y poder realizar la asignación.

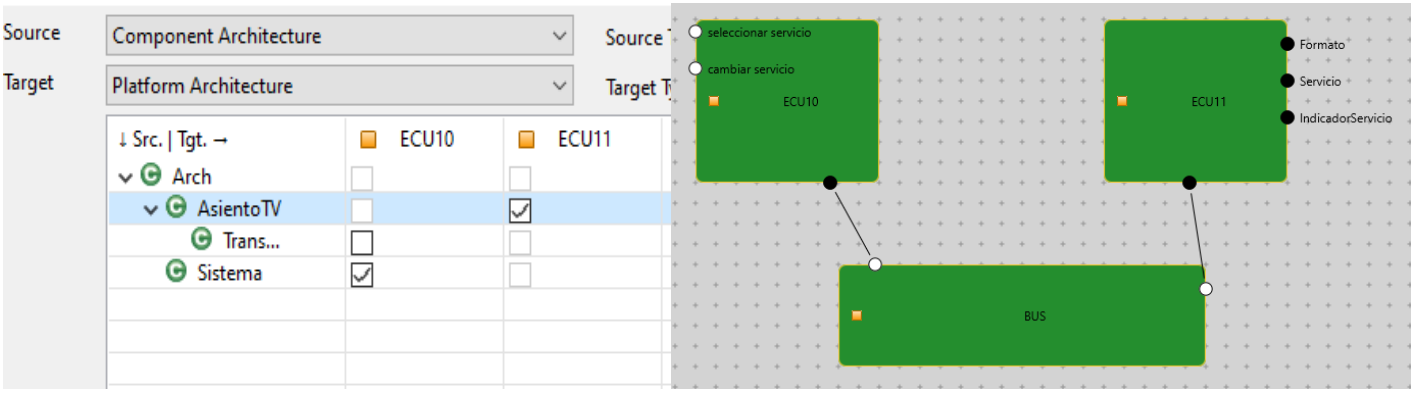


Imagen 4.34. Asignaciones de componentes al hardware con el diagrama de arquitectura genérica con asignaciones.

Generación de código

Es una de las fases mediante el cual un compilador convierte un programa sintácticamente correcto en una serie de instrucciones a ser interpretadas por una máquina. La generación de código es usada para construir programas de una manera automática evitando que los programadores tengan que escribir el código a mano y puede realizarse en tiempo de ejecución, tiempo de carga o tiempo de compilación. AutoFocus3 admite la generación de código C a partir de arquitecturas de componentes y una definición de asignación de plataforma asociada. Para generar el código de dichos componentes, se ejecuta una opción de almacenamiento y partir del diagrama de plataforma asociado se genera el código C. En la imagen 4.35, hasta la imagen 4.40 se observan el código generado con resultado favorable

creado con su lógica proporcionada, junto con sus archivos ejecutables del sistema sobre los componentes creados “Sistema”, “Asiento TV” y “Transmisión”. En el **Anexo B** se describe a detalle el código sobre cada componente.

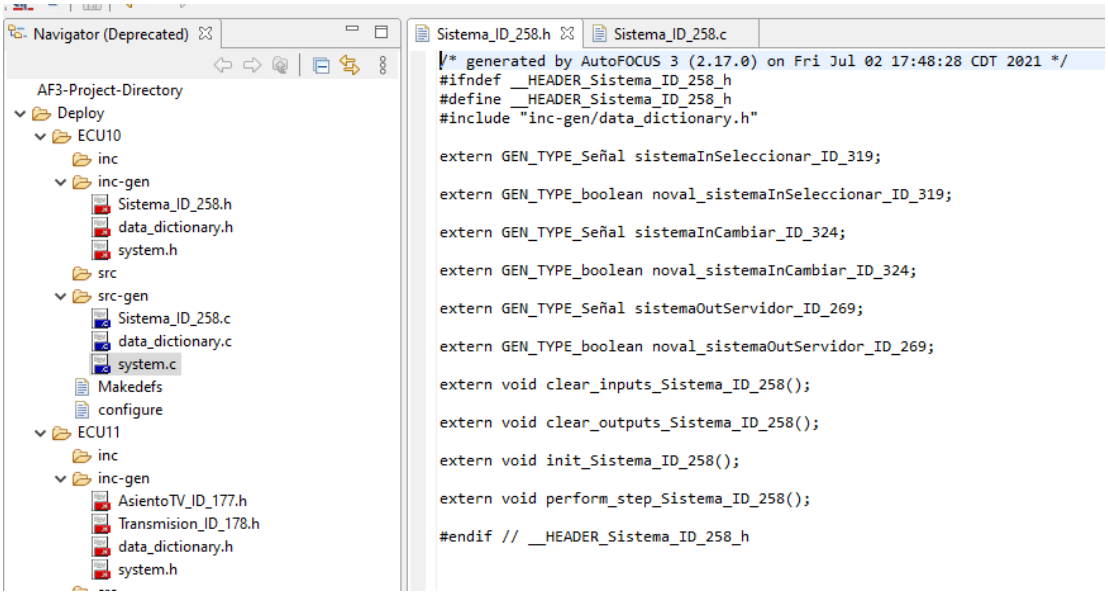


Imagen 4.35. Generación de código del componente “Sistema con extensión .h”.

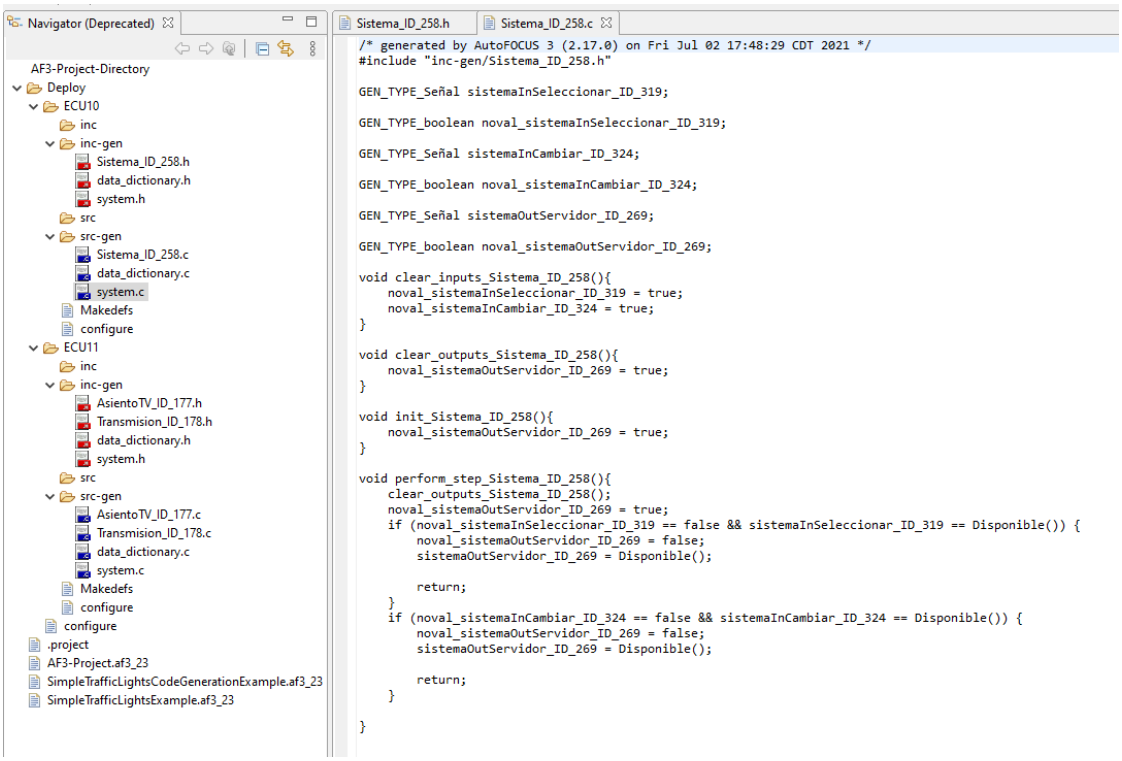


Imagen 4.36. Generación de código del componente “sistema con extensión .c”.

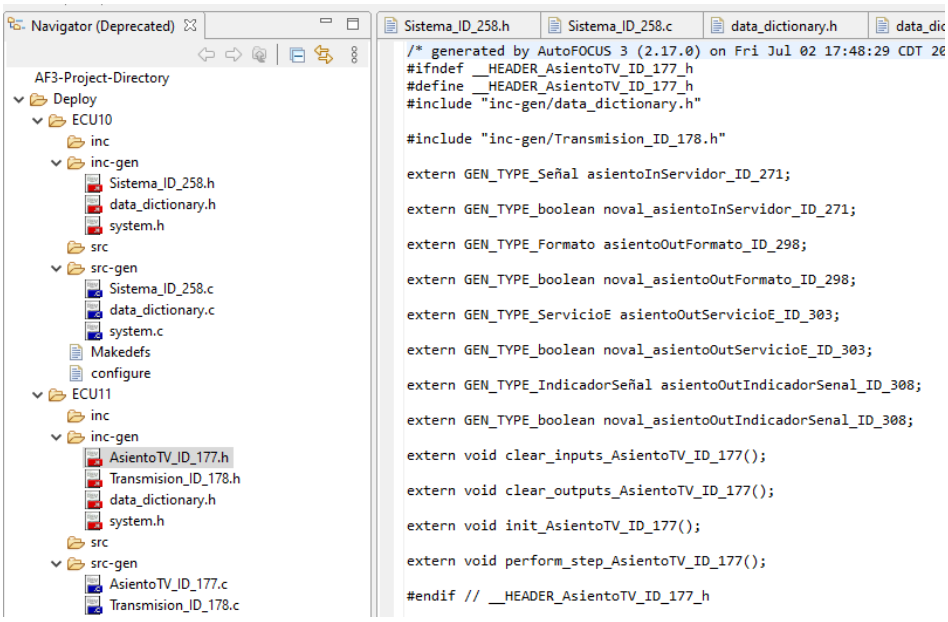


Imagen 4.37. Generación de código del componente "Asiento TV con extensión .h".

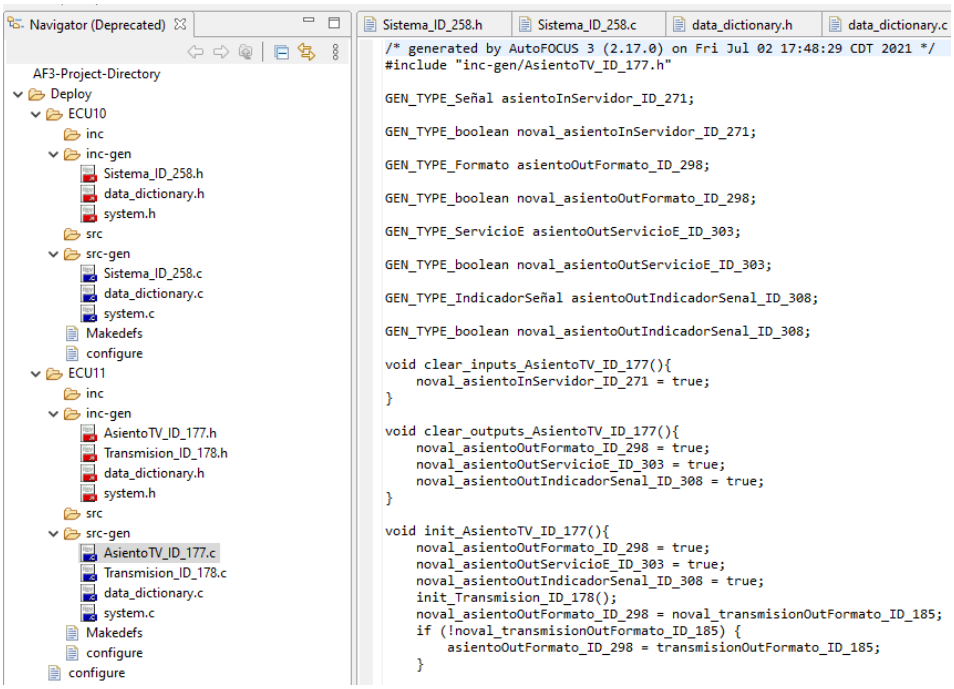


Imagen 4.38. Generación de código del componente "Asiento TV con extensión .c" parte 1.

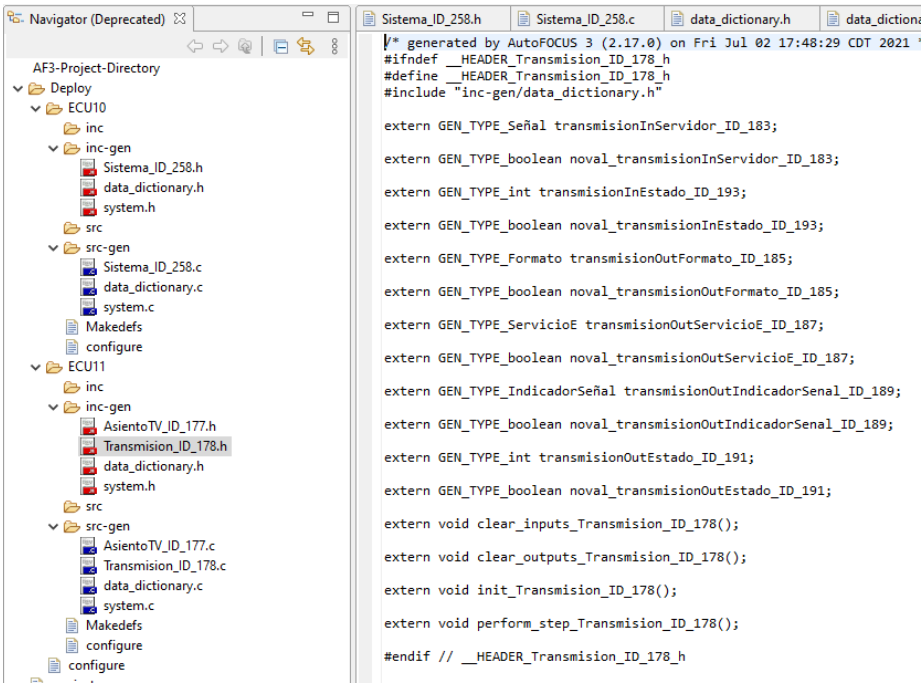


Imagen 4.39. Generación de código del componente "Transmisión con extensión .h".

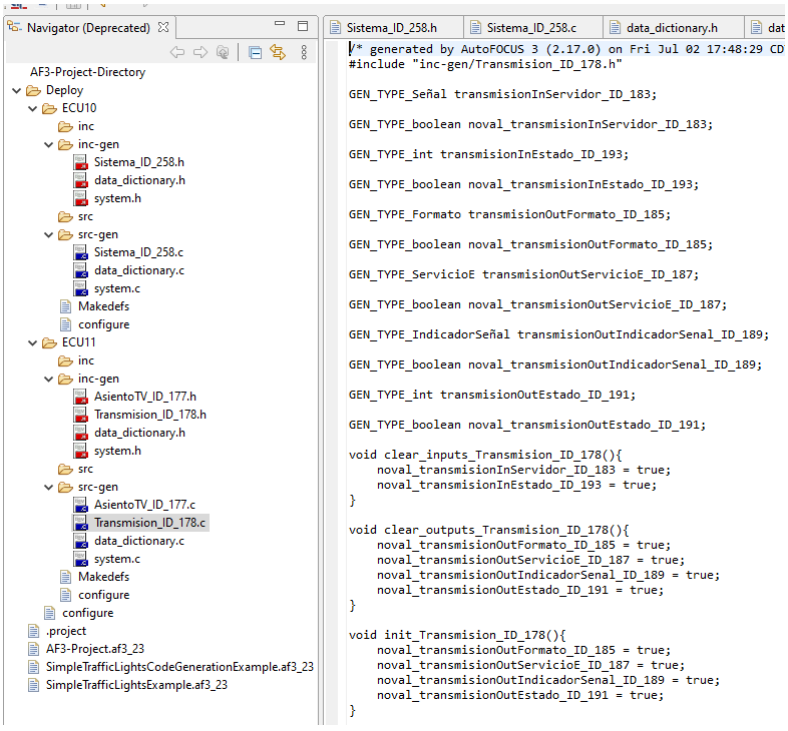


Imagen 4.40. Generación de código del componente "Transmisión con extensión .c" parte 1.

5. RESULTADOS Y DISCUSION

El estudio examinó cómo se puede ejecutar las actividades principales de MBSE, junto con el desarrollo de la arquitectura a través de un enfoque de desarrollo de arriba hacia abajo utilizando ARCADIA / Capella, junto con la colaboración de la herramienta AutoFocus3. Con base en los criterios de evaluación descritos en el Capítulo 3, la evaluación de las dos herramientas MDE, se realizó para comparar, de forma temprana, su atributo de "madurez" de sobrellevar un desarrollo en las dos herramientas MDE, con respaldo de la Metodología ARCADIA. Las siguientes secciones se centran en resumir la evaluación. La segunda sección destaca las equivalencias y diferencias clave entre las dos herramientas MDE. La tercera sección proporciona las observaciones clave y propone una visión para la digitalización del sistema al lograr el hilo digital a lo largo del ciclo de vida completo del sistema.

5.1. Evaluación

5.1.1. Selección de dos herramientas MDE por atributo "madurez"

La madurez se va a medir con la sumatoria de desarrollo, legado, actividad, industrialización y características, como se observa en la ecuación 1.

$$\textbf{(1)} \qquad Madurez = D + L + A + I + C$$

Desarrollo (D)

A continuación, en la tabla 5.1, se muestra el atributo a medir con sus respectivos sub-atributos, donde se contempla con sus respectivas métricas y principios de cada una con ponderaciones.

Tabla 5.1. Ponderaciones para medir el atributo "Desarrollo".

Atributo	Sub-atributo	Métrica	Ponderación	Principio
Desarrollo	Metodología	Mt	5	Mt existe y adaptable = 5 Mt existe = 2.5 Mt no existe = 0
	Modelado de arquitectura	Ma	5	Ma existe diagramas detallados = 5 Ma existe diagramas básicos = 2.5 Ma no existe = 0
	Transformación de modelos	Tm	5	Tm existe transformación completa = 5 Tm existe transición lineal = 2.5 Tm no existe = 0
	Generación de código	Gc	5	Gc genera código C = 5 Gc genera código java = 2.5 Gc no genera código = 0

Por consecuente, para determinar el desarrollo se realiza las sumatorias como en la siguiente ecuación 2.

$$\textbf{(2)} \qquad \boldsymbol{D} = Mt + Ma + Tm + Gc$$

Legado (L)

A continuación, en la tabla 5.2, se muestra el atributo a medir con sus respectivos sub-atributos, donde se contempla con sus respectivas métricas y principios de cada una con ponderaciones.

Tabla 5.2. Ponderaciones para medir el atributo "Legado".

Atributo	Sub-atributo	Métrica	Ponderación	Principio
Legado	Edad	Ed	5	Ed menos de 9 años = 5 Ed entre 10 y 19 años = 2.5 Ed más de 20 años = 0
	Historial	Hl	5	Hl mucha historia de uso = 5 Hl uso regular = 2.5 Hl no existe uso = 0
	Equipo de desarrollo	Eq	5	Eq muchas etapas de desarrollo = 5 Eq etapa de análisis = 2.5 Eq no admite desarrollo grupal = 0
	Popularidad	Pd	5	Pd popular entre empresas y desarrolladores = 5 Pd discreta entre desarrolladores = 2.5 Pd no se conoce = 0

Por consecuente, para determinar el legado se realiza las sumatorias como en la siguiente ecuación 3.

$$\textbf{(3)} \qquad \boldsymbol{L} = Ed + Hl + Eq + Pd$$

Actividad (A)

A continuación, en la tabla 5.3, se muestra el atributo a medir con sus respectivos sub-atributos, donde se contempla con sus respectivas métricas y principios de cada una con ponderaciones.

Tabla 5.3. Ponderaciones para medir el atributo "Actividad".

Atributo	Sub-atributo	Métrica	Ponderación	Principio
Actividad	Comunidad contribuyente	Cc	5	Cc intensa actividad en foros = 5 Cc regular actividad en foros = 2.5 Cc no existe actividad = 0
	Fallos	Fa	5	Fa fuerte asignación en tareas y roles = 5 Fa existente, pero sin proceso definido = 2.5 Fa no existe = 0
	Funcionalidades	Fu	5	Fu funcionalidades industrializadas = 5 Fu seguida por un grupo dedicado = 2.5 Fu sin nuevas características = 0
	Versiones	Ve	5	Ve versiones correctivas frecuentes = 5 Ve menos versiones correctivas = 2.5 Ve no existe = 0

Por consecuente, para determinar la actividad se realiza las sumatorias como en la siguiente ecuación 4.

$$(4) \qquad \boldsymbol{A} = Cc + Fa + Fu + Ve$$

Industrialización (I)

A continuación, en la tabla 5.4, se muestra el atributo a medir con sus respectivos sub-atributos, donde se contempla con sus respectivas métricas y principios de cada una con ponderaciones.

Tabla 5.4. Ponderaciones para medir el atributo "Industrialización".

Atributo	Sub-atributo	Métrica	Ponderación	Principio
Industrialización	Servicios	Se	5	Se muchos servicios ofrecidos = 5 Se servicios limitados = 2.5 Se no existen servicios = 0
	Documentación	Do	5	Do documentación actualizada = 5 Do documentación con poco detalle = 2.5 Do no existe = 0
	Aseguramiento de calidad	Ac	5	Ac basado en metodologías estándar = 5 Ac metodologías no normalizadas = 2.5 Ac no existe = 0
	Modificación código fuente	Mf	5	Mf bien definido y respetado = 5 Mf regular para modificar = 2.5 Mf no existe modificación = 0

Por consecuente, para determinar la industrialización se realiza las sumatorias como en la siguiente ecuación 5.

$$(5) \qquad \boldsymbol{I} = Se + Do + Ac + Mf$$

Características (C)

A continuación, en la tabla 5.5, se muestra el atributo a medir con sus respectivos sub-atributos, donde se contempla con sus respectivas métricas y principios de cada una con ponderaciones.

Tabla 5.5. Ponderaciones para medir el atributo "Características".

Atributo	Sub-atributo	Métrica	Ponderación	Principio
Características	Extensible	Ex	5	Ex abierto a extensiones = 5 Ex agregar algunas extensiones = 2.5 Ex no existe = 0
	Simplicidad	Si	5	Si amigable de utilizar = 5 Si cierta dificultad de uso = 2.5 Si mayor dificultad, difícil = 0
	Multiformato de datos	Md	5	Md admite extensiones de archivos = 5 Md admite aplicaciones = 2.5 Md admite solo propias = 0
	Multiplataforma	Mu	5	Mu cualquier sistema operativo = 5 Mu un sistema operativo = 2.5 Mu ninguno = 0

Por consecuente, para determinar las características se realiza las sumatorias como en la siguiente ecuación 6.

$$(6) \qquad C = Ex + Si + Md + Mu$$

Una vez terminadas las evaluaciones se debe hacer la comparación manual de las herramientas MDE, obteniendo un puntaje promedio con las ponderaciones elaboradas anteriormente. Se tomó como prioridad, de la taxonomía de herramientas de la tabla 3.3, las herramientas MDE que fueran de código libre y su licencia no fuera de paga, por lo cual, resultaron 15 herramientas MDE para la evaluación y selección respecto al atributo de calidad "madurez". Se muestran en el **Anexo C**, las tablas de comparaciones de las herramientas por criterio. En la imagen 5.6, se muestra el promedio final de las herramientas MDE.

Tabla 5.6. Comparación de herramientas por promedio final.

No.	Herramienta	Desarrollo	Legado	Actividad	Industrialización	Características	Promedio
1	*AndroMDA*	5	10	0	7.5	7.5	**30**
2	*Enterprise Architect*	7.5	12.5	10	12.5	15	**57.5**
3	*Visual Paradigm*	5	10	12.5	10	7.5	**45**
4	*StarUML*	7.5	7.5	12.5	12.5	12.5	**52.5**
5	*Sirius*	5	5	7.5	12.5	7.5	**37.5**
6	*ATL*	5	5	10	12.5	10	**42.5**
7	*VIATRA*	5	7.5	10	12.5	7.5	**42.5**
8	*Workflow*	2.5	2.5	0	5	2.5	**12.5**
9	*Epsilon*	7.5	10	5	10	7.5	**40**
10	*Acceleo*	2.5	7.5	7.5	12.5	7.5	**37.5**
11	*Simulink*	2.5	10	10	10	10	**42.5**
12	*CDT*	2.5	2.5	5	5	7.5	**22.5**
13	*Papyrus*	12.5	12.5	7.5	15	12.5	**60**
14	*AutoFocus3*	12.5	15	10	12.5	17.5	**67.5**
15	*Capella*	15	17.5	10	15	12.5	**70**

Según los resultados anteriores las herramientas a integrar son: AutoFocus3 y Capella, por lo tanto, se realizó un pequeño desarrollo para verificar su funcionalidad, empezando con el desarrollo de la arquitectura en la herramienta Capella y para después finalizar con la herramienta AutoFocus3.

5.2. Entregables claves del proceso

Es esencial que el modelo proporcione los resultados clave esperados del modelo durante las fases de desarrollo para facilitar un mayor desarrollo de la arquitectura. Las arquitecturas desarrolladas se identificaron artefactos de ambas herramientas que de alguna manera permitirán la provisión de entregables. Ambas soluciones proporcionan conceptos de metamodelo más o menos similares para admitir artefactos entregables de desarrollo de arquitectura. Las tablas 5.7 a 5.10, muestran la lista de entregables y artefactos modelo que respaldan la provisión de esos entregables. Esta no es una lista exhaustiva y no pretende cubrir todos los entregables requeridos en el proceso de MBSE. Sin embargo, la lista se puede considerar suficiente para evaluar la utilidad de los datos de la arquitectura del sistema en provisión de los entregables.

Tabla 5.7. Artefactos entregables de arquitectura después del análisis operacional.

Fase	**Entregable**	**ARCADIA / Capella**	
		Alcance del elemento	**Alcance del diagrama**
Análisis operacional ***Objetivo:*** - *Para identificar a todas las partes interesadas pertinentes* - *Para presentar con precisión sus necesidades* - *Integrar las necesidades en un conjunto coherente de requisitos de las partes interesadas* - *Validar el conjunto entre las partes interesadas* - *Para definir conceptos operativos* ***Propósito:*** - *Mejorar la comprensión de la misión y los requisitos de las partes interesadas*	*Especificación de los requisitos de las partes interesadas*	• *Requerimientos*	*N/A*
	Documento de justificación de requisitos (trazabilidad de requerimientos)	• *Enlace de requerimientos*	• *Diagrama de arquitectura operacional* • *Diagrama de interacción de actividades operacionales* • *Diagrama de capacidades operacionales* • *Diagrama de desglose de la actividad operativa* • *Navegador semántico*
	Entrada para borradores de planes de verificación y validación	• *Actividad operacional* • *Entidades operacionales* • *Actores operacionales* • *Actividades de interacción* • *Medios de comunicación*	• *Diagrama de escenarios de actividades operacionales* • *Diagrama de cadenas funcionales* • *Diagrama de escenarios de entidades operacionales* • *Diagrama de procesos operacionales*
	Conceptos operacionales	• *Capacidades operacionales* • *Enlace de participación*	• *Diagrama de capacidades operacionales*

Tabla 5.8. Artefactos entregables de arquitectura después del análisis de requerimientos del sistema.

Fase	Entregable	ARCADIA / Capella	
		Alcance del elemento	**Alcance del diagrama**
Análisis de requerimientos del sistema ***Objetivo:*** - *Traducir los requisitos de la misión y de las partes interesadas en un conjunto de requisitos del sistema de alta calidad* ***Propósito:*** - *Transformar la vista de las partes interesadas y orientada al usuario de las capacidades deseadas en una vista técnica de una solución que satisfaga las necesidades operativas del usuario.*	*Especificación de requerimientos del sistema*	• *Requerimientos*	*N/A*
	Definición de funciones del sistema (Arquitectura funcional)	• *Funciones del sistema* • *Intercambio funcional*	• *Diagrama de flujo de datos del sistema* • *Desglose funcional* • *Proceso funcional* • *Diagrama de arquitectura del sistema*
	Documento de justificación de requerimientos del sistema (Trazabilidad de requerimientos)	• *Requerimientos* • *Enlace de requerimientos*	• *Desglose funcional del sistema* • *Diagrama de capacidades del sistema* • *Navegador semántico*
	Diagrama contextual	• *Sistema* • *Actores del sistema* • *Puerto de componentes* • *Intercambio de componentes*	• *Diagrama contextual de interfaces externas* • *Diagrama contextual de actores del sistema* • *Diagrama contextual de interfaces detalladas*
	Documentación de interfaces externas	• *Clases* • *Propiedades*	• *Diagrama de clases*

Tabla 5.9. Artefactos entregables de arquitectura después de la definición de la arquitectura lógica.

Fase	Entregable	ARCADIA / Capella	
		Alcance del elemento	**Alcance del diagrama**
Definición de la arquitectura lógica ***Objetivo:*** - *Traducir los requisitos de la misión y de las partes interesadas en un conjunto de requisitos del sistema de alta calidad* ***Propósito:*** - *Para obtener una arquitectura de solución neutral del sistema que satisfaga los requisitos*	*Especificación de requerimientos del subsistema*	• *Requerimientos*	*N/A*
	Descripción del comportamiento interno	• *Funciones lógicas* • *Modo y estado* • *Intercambios lógicos* • *Escenarios de cadenas funcionales*	• *Diagrama de flujo de datos lógicos* • *Diagrama de arquitectura lógica* • *Diagrama de desglose de funciones lógicas* • *Diagrama de intercambio de escenarios*
	Descripción de la arquitectura del sistema (Arquitectura lógica)	• *Componentes lógicos* • *Actores lógicos* • *Intercambios funcionales*	• *Diagrama contextual de interfaces internas*
	Documentación de justificación de los requisitos del subsistema	• *Enlace de requerimientos*	• *Diagrama de navegador semántico*
	Descripción de la interfaz interna	• *Puertos de componentes* • *Intercambio de componentes* • *Clases* • *Propiedades*	• *Diagrama de clases*

Tabla 5.10. Artefactos entregables de arquitectura después de la definición de la arquitectura física.

Fase	Entregable	ARCADIA / Capella	
		Alcance del elemento	**Alcance del diagrama**
Definición de la arquitectura física ***Objetivo:*** - *Desarrollar una arquitectura que consta de componentes físicos dependientes de la tecnología o partes necesarias para cumplir las funciones de los componentes lógicos* ***Propósito:*** - *Definir los principios estructurales de la arquitectura y su comportamiento*	*Especificación de requerimientos de componentes*	• *Requerimientos*	*N/A*
	Descripción del comportamiento físico	• *Funciones físicas* • *Modo y estado* • *Intercambios funcionales* • *Cadenas de escenarios funcionales*	• *Diagrama de flujo de datos físicos* • *Diagrama de descomposición de la función física* • *Diagrama de arquitectura física* • *Diagrama de máquina de modo y estado* • *Diagrama de intercambio de escenarios*
	Descripción de la arquitectura física	• *Componentes de nodos físicos* • *Componentes de comportamiento* • *Intercambio de componentes*	• *Diagrama contextual de interfaces internas*
	Justificación de los requisitos de los componentes	• *Enlace de requerimientos*	• *Navegador semántico*
	Interfaces físicas	• *Puertos de componentes* • *Puertos de funciones* • *Puertos físicos* • *Enlaces físicos* • *Propiedades*	• *Diagrama de clases*

5.3. Aspectos destacados y diferencias clave

En esta sección se describen algunos de los aspectos más destacados entre las herramientas, Capella y AutoFocus3, y diferencias entre las mismas.

Transiciones automatizadas entre perspectivas en Capella y AutoFocus3

Capella proporciona sus capacidades utilizando cuatro perspectivas de modelado con diferentes objetivos para guiar a los usuarios finales. Mientras que en AutoFocus3 ofrece sus capacidades en tres perspectivas de modelado mediante composición, especificación y generación de código. Una de las principales características de Capella y AutoFocus3, que los hace más fácil de usar, son transiciones automatizadas entre elementos de diferentes perspectivas. Por ejemplo, en Capella, una actividad operativa puede convertirse en una función del sistema en “Arquitectura del sistema”, que luego puede realizarse como una función lógica en “Arquitectura Lógico” y posteriormente como una función física en “Arquitectura Física”. Esto crea enlaces de realización automática entre estos componentes, creando así una trazabilidad automática entre varios niveles de abstracción. Elementos como capacidades, componentes, actores y entidades, o estados, elementos de intercambio y tipos de datos, también se pueden cambiar. En AutoFocus3, sucede similar, solo que, a partir de la generación de componentes, se le asignan la lógica por medio de especificaciones de código

y automáticamente se crea un enlace sobre la "Arquitectura de plataforma" que se desee implementar. En la imagen 5.1 y 5.2, muestra las transiciones aplicadas en las herramientas Capella y AutoFocus3.

5.4. Resultados generales vs hipótesis y objetivos específicos

Como resultados, se puede decir que se cumple con lo especificado en las diferentes fases de desarrollo de un sistema embebido o ciber-físico y en la tabla 5.11, se puede observar los objetivos específicos con lo realizado para cumplir con la meta de valor.

Tabla 5.11. Resultados con los objetivos específicos planteados.

No.	***Objetivos específicos***	***Actividades realizadas***	***Herramienta o metodología utilizada***
1	*Construir una taxonomía de las herramientas disponibles de código libre/free para desarrollo dirigido por modelos (MDD)*	Taxonomía de herramientas	Búsqueda en repositorios
2	*Identificación de métricas*	Atributo "Madurez"	Metodología QSOS
3	*Identificación del problema modelo*	Sistema de entretenimiento en un vuelo	Ejemplo de la herramienta Capella
4	*Utilizar técnicas para evaluación de herramientas*	Evaluación y comparativa de las herramientas	Metodología QSOS
5	*Experimentación del desarrollo en cada entorno*	Análisis y diseño (Capella), Implementación (AutoFocus3)	Capella y AutoFocus3

En la tabla 5.12, se describen las hipótesis planteadas contra los resultados arrojados en el desarrollo del sistema de entretenimiento.

Tabla 5.12. Resultados con las hipótesis planteadas.

No.	***Hipótesis***	***Actividad relacionada***
1	*Se puede realizar una clasificación de los entornos de desarrollo y de las herramientas dirigidas por modelos, considerando todas sus características*	Taxonomía de herramientas
2	*Se puede seleccionar dos con mejores características*	Evaluación y comparativa de herramientas
3	*Se puede desarrollar una experimentación a un sistema embebido o ciber-físico con las herramientas seleccionadas*	Sistema de entretenimiento en un vuelo
4	*Se puede documentar el desarrollo por medio de una simulación con este tipo de herramientas*	Documentación de pruebas de desarrollo en herramientas

A continuación, se describen los resultados obtenidos contra las hipótesis planteadas de la tabla 5.12, sobre la propuesta de tesis y cumplir con la meta de valor de cada objetivo específico.

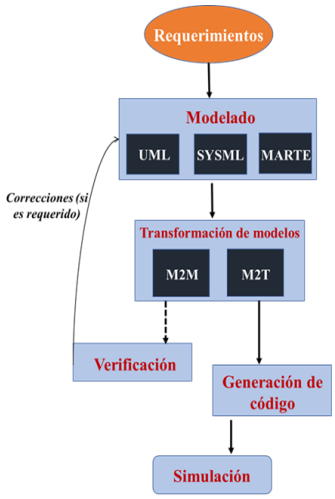
1. *Si se puede realizar una clasificación de las herramientas, pero se necesita retroalimentar a corto plazo para tener actualizada la lista de herramientas vigentes en el desarrollo sobre este tipo de sistemas embebidos o ciber-físicos.*

2. *Si se puede seleccionar 2 herramientas con sus mejores características, por lo que cada una ofrece, en similitudes de desarrollo, pero hay herramientas que son dedicadas a una etapa del desarrollo específicamente y es muy difícil poner una herramienta contra la otra porque la que ofrece una herramienta la otra la complementa.*

3. *Si se puede realizar un desarrollo de un sistema embebido o ciber-físico, por lo cual tiene que ser necesariamente con características de interacción en tiempo real, relacionado con partes físicas como lo son sensores, actuadores, entre otros.*

4. *Si se puede documentar el desarrollo del sistema, existe documentación referente a las funciones de la herramienta, pero no abordan un desarrollo completo y lineal, por lo que fue un poco difícil alinear lo que ofrecía la herramienta contra la función de los componentes y los límites que abordan.*

Referente a las actividades del desarrollo de la ingeniería basada en modelos (MBSE), de la imagen 5.3, se describen el uso de las herramientas elegidas en cada actividad.



No.	Actividad o Etapa	Herramienta utilizada
1	*Requerimientos*	AutoFocus3
2	*Modelado*	Capella /ARCADIA
3	*Transformación de modelos*	Capella / ARCADIA (Modelo a modelo) AutoFocus3 (Modelo a modelo y a texto)
4	*Verificación*	AutoFocus3 Capella / ARCADIA
5	*Generación de código*	AutoFocus3
6	*Simulación*	Lo ofrece AutoFocus3, pero no se abordó esta actividad.

Imagen 5.3. Uso de herramientas MDE en las diferentes etapas de MBSE.

6. CONCLUSION Y TRABAJOS FUTUROS

6.1. Conclusiones

La tesis investigó los enfoques de desarrollo de arquitectura para el método ARCADIA usando la herramienta Capella, con una colaboración de la herramienta AutoFocus3. Se implementó una evaluación sobre la selección de herramientas MDE, por medio del atributo de calidad "madurez", para poder realizar el desarrollo del sistema IFE en dos herramientas MDE candidatas de manera colectiva. El enfoque de modelado se presentó para respaldar las cuatro fases del desarrollo de la arquitectura; análisis operativo, análisis de requisitos del sistema, definición de la arquitectura lógica y definición de la arquitectura física que cumpliría con las actividades principales planteadas de la MBSE.

Capella proporciona características prometedoras a través del flujo de trabajo de modelado integrado en el método ARCADIA. Mientras que AutoFocus3 no impone ningún método específico para el modelado y aporta complementos específicos para poder finalizar exitosamente las actividades principales de MBSE, referente a generar el código C.

Sin embargo. ARCADIA es un método basado en el análisis funcional que se centra en definir y justificar la arquitectura. El análisis funcional ha sido una de las técnicas más importantes utilizadas en SE y es imperativo que cualquier solución MBSE lo admita. Capella se destaca por varias características intuitivas para el usuario, como las transiciones automatizadas, las cadenas funcionales y las capacidades de verificación de modelos. AutoFocus3 se basa en el modelado basado en componentes, mientras que Capella proporciona un modelado basado en instancias que simplifica la tarea de crear partes redundantes dentro de los modelos.

Además, ambas herramientas de software brindan asistencia para modelar arquitecturas complejas a través de varios complementos externos e integración de terceros. Capella brinda la capacidad de modelar restricciones y crear puntos de vista especializados personalizados a través de complementos de código abierto.

En resumen, se debe tener en cuenta que el alcance de ARCADIA / Capella se centra en las actividades de definición de arquitectura e ingeniería, mientras que el alcance de AutoFocus3 se extiende más allá de la definición de la arquitectura, como la creación de código, a partir de la arquitectura de plataforma genérica o especifica.

6.2. Trabajos futuros

Además de la evaluación, también identificamos algunas limitaciones, como algunos desafíos a corto y largo plazo.

6.2.1. Limitaciones

Una de las principales limitaciones es la naturaleza de la comparación, y es uno de los motivos por el cual se determinó llevar el desarrollo del sistema IFE, por medio de una colaboración entre las dos herramientas MDE. ARCADIA / Capella es una combinación de un método y una herramienta, mientras que AutoFocus3 es una herramienta de verificación y validación de componentes, que hace imposible realizar una comparación uno a uno. En

segundo lugar, el análisis funcional es una faceta clave de ARCADIA / Capella, por lo que en AutoFocus3, la especificación de código es propiamente el alma de la herramienta para hacer funcional los componentes creados. Las siguientes subsecciones describen las perspectivas futuras de aumentar este estudio de investigación a corto y largo plazo.

6.2.2. A corto plazo

1. Evaluación del análisis funcional: el estudio no evaluó explícitamente las capacidades del análisis funcional en ambas herramientas. Una evaluación más enfocada debe complementar sobre métodos de análisis funcional especializados.

2. Evaluación del código generado: a partir de una arquitectura de plataforma, se genera el código c, pero se necesita realizar una evaluación de funcionalidad sobre el código generado sobre el sistema que se esté desarrollando.

3. Estudio de caso de estudio alternativo: los resultados del estudio se pueden aumentar utilizando un ejemplo de un dominio de aplicación más especializado debido a la posibilidad de producir diferentes resultados al cambiar el dominio de la aplicación.

4. Comparativa: el ecosistema de Capella aporta los aspectos necesarios a la arquitectura de sistemas, es decir, un método, un lenguaje de modelado y una herramienta de apoyo. En cuanto a lo que ofrece AutoFocus3, aporta verificación y validación sobre generación de código y se necesita que coincida con las características que Capella ofrece para realizar un estudio más centrado, que podría demostrar varios beneficios adicionales de usar una herramienta especializada.

6.2.3. A largo plazo

1. Integración de arquitectura multidominio: MBSE es un marco, no una disciplina. El MBSE exitoso requiere que los modelos de varios dominios se integren para formar ingeniería integrada basada en modelos. Podrían integrarse herramientas de simulación y un trabajo similar entorno a las arquitecturas de Capella podría proporcionar resultados favorables.
2. Realizar una co-simulación con desarrollo de arquitecturas de plataforma especificas: una vez realizado el proceso de las actividades de MBSE, se necesitaría probar el código generado, incrustarlo en software de arquitectura de plataforma, verificando funcionalidad, en cuanto a evaluar el código con atributos de calidad orientados hacia funcionalidad y eficacia.

REFERENCIAS

Akash S. M., Mustafizur Rahman; QFD-Based Smart Cane Design: A Technology to Assist Visually Impaired; *International Journal of Business Analytics and Intelligence*; 6 (1) 2018, 25-30.

Arcadia method at a glance! https://www.polarsys.org/capella/arcadia.html. Acceso: 2020-04-09.

Bone M. and R. Cloutier R., "The current state of model based systems engineering: Results from the omgTM sysml request for information 2009," in *Proceedings of the 8th conference on systems engineering research*, 2010.

Burak Sarp, Tolga Karalar; Real Time Smart Door System for Home Security; *International Journal of Scientific Research in Information Systems and Engineering (IJSRISE)*; Volume 1, Issue 2, December-2015.

Carlos A. García, Esteban X. Castellanos, Marcel V. García; Developing cyber-physical production systems for batch control under IEC-61499 and ISA-88; *Ingeniare. Revista chilena de ingeniería*, vol. 27 Nº 3, 2019, pp. 443-453.

César Cuevas, Laura Barros, Patricia López Martínez y José M. Drake; Beneficios que aporta la metodología MDE a los entornos de desarrollo de sistemas de tiempo real; *Revista Iberoamericana de Automática e Informática industrial* 10 (2013) 216–227.

Charles Rivet (.ppt), Papyrus for RealTime- Executable modeling on Eclipse, *eclipsecon, Reston Virginia*, 2016.

Cook D. and Schindel W. D., "Utilizing mbse patterns to accelerate system verification," *Insight*, vol. 20, no. 1, pp. 32–41, 2017.

Crawley E., Cameron B., and Selva D., System architecture: strategy and producto development for complex systems. *Prentice Hall Press*, 2015.

Diana Julissa Ramirez Álvarez, Andres Felipe Tapia Vertel, Valery Jose Lancheros Suares, Luis Armando Espitia Sanjuán; Analysis of a Solar Intermittent Absorption Refrigeration System; *Revista de Ingenieria, Universidad de Los Andes Colombia*, 2015.

Dick J., Hull E., and Jackson K., *Requirements engineering*. Springer, 2017.

Enrique Calot, Mariano Maluf, Marcelo Neffa; Estacionamiento Inteligente con IoT; *Universidad Nacional de La Plata, Facultad de Informática*; Julio, 2017.

Ernesto Posse; PapyrusRT: Modelling and Code Generation; 2015.

Estefan J. A. et al., "Survey of model-based systems engineering (mbse) methodologies," *Incose MBSE Focus Group, vol. 25, no. 8, pp. 1–12*, 2007.

Fáber D. Giraldo, Sergio España, William J. Giraldo, Óscar Pastor; Evaluación de la calidad de lenguajes de modelado a través de análisis taxonómico: una propuesta preliminar; *Revistas Ingenierias Universidad de Medellin*; 2016.

Gabriela Sobrevilla, José Hernandez, Perla Velasco-Elizondo, Silvia Soriano; Applying Scrum and Software Engineering Practices to Continuously Improve the Development of a CyberPhysical System; *Computación e Informatica*; Mayo 2017.

Hoffmann H. P., "Systems engineering best practices with the rational solution for systems and software engineering," *IBM Software Group*, vol. 4, no. 2, 2011.

INCOSE T., "Systems engineering vision 2020," *INCOSE, San Diego, CA, acceso Enero, vol. 26, p. 2020,* 2007.

Ingham M. D., Rasmussen R. D, Bennett M. B., and Moncada A. C., "Engineering complex embedded systems with state analysis and the mission data system," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 12, pp. 507–536, 2005.

Jesus Sánchez Cuadrado, Esther Guerra, Juan de Lara; Static Analysis of Model Transformations; *IEEE Transactions on software engineering, vol. 43, N° 9,* September 2017.

Joel Greenyer, Daniel Gritzner, Jianwei Shi, Eric Wete; A Scenario-based MDE Process for Developing Reactive Systems: A Cleaning Robot Example; *Leibniz Universität Hannover, Germany*, 2017.

José Guillermo Fierro Mendoza, Julio Armando Asato España, José Benigno Molina Castro, Jonathan Giovanni Delgado Núñez, Emmanuel Noriega Vaca; propuesta metodológica para validar la funcionalidad de software en sistemas embebidos, *Pistas Educativas*, No. 122, diciembre 2016. México.

Juan A. Guerrero-Ibáñez, Jazmín Acosta-Mendoza, Juan J. Contreras-Castillo; Dándole Inteligencia a la Gestión de Estacionamientos en Zonas Públicas a través del Sistema i-PARKING; *Sistemas Cibernética e Informática*; Vol12 num2 2015.

Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, Alfonso Pierantonio; Collaborative Repositories in Model-Driven Engineering; *IEEE Software*; 2015.

Levi Lúcio, Sudeep Kanav, Andreas Bayha, and Johannes Eder; Controlling a virtual rover using AutoFOCUS3; 80805 München; 2018.

Long D. and Scott Z., *A primer for model-based systems engineering. Lulu. com,*2011.

Manuel A. Calle, Jose D. Tovar, Yor J. Castaño-Pino, Juan C. Cuellar; Comparación de Parámetros para una Selección Apropiada de Herramientas de Simulación de Redes; *Información Tecnológica*, vol. 29(6), 253-266 (2018).

Márcio Assis Miranda, Marcos Guiherme Ribeiro, Humberto Torres Marques-Neto, Mark Alan, Junho Song; Domain-specific language for automatic generation of UML models; *IET Software*, October 2017.

Marco Autili, Antonia Bertolino, Guglielmo De Angelis, Davide Di Ruscio, Alessio Di Sandro; A Tool-Supported Methodology for Validation and Refinement of Early-Stage Domain Models; *IEEE Transactions on software engineering, vol. 42, N° 1* January 2016.

Martin J. N., *Systems engineering guidebook: A process for developing systems and products.* CRC press, 1996, vol. 10.

Matworks; Model-Based Design for Embedded Control Systems; White paper; 2019.

Mercedes Picón M., Carmen Zulay Maldonado; Generación Automática de Código basada en Modelos UML; Colegio Universitario de Caracas, Caracas, Venezuela - 26 al 28 de octubre de 2016.

Muhammad Rashid, Muhammad Waseem Anwar, Aamir M. Khan; Toward the tools selection in model-based system engineering for embedded systems- Asystematic 75iterature review; *The Journal of Systems and Software* 106 (2015) 150-163.

Nafiseh Kahani, James R. Cordy; Comparison and Evaluation of Model Transformation Tools; *School of Computing, Queen´s University Kingston, Ontario*; December 2015.

Nafiseh Kahani, Nicolas Hili, James R. Cordy, Juergen Dingel; Evaluation of UML-RT and Papyrus-RT for Modelling Self-Adaptive Systems; *School of Computing, Queen's University Kingston,* Ontario, Canada, 2017.

Nicolas Hili and Juergen Dingel, Alain Beaulieu; Modelling and Code Generation for Real-Time Embedded Systems with UML-RT and Papyrus-RT; May 2017.

Open source solution for model-based systems engineering," https://www.polarsys.org/capella/, acceso: 2020-04-09.

Pablo Valencia Bibian, Amparo Dora Palomino Merino, Sergio Vergara Limon; Quadcopter trajectory tracking using visual servoing; *2do Congreso Internacional de Sistemas Embebidos y Mecatrónica*; 978-607-9394-05-9/©2016 AMESE.

Pascal Roques; MBSE with the ARCADIA Method and the Capella; *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016),* Jan 2016, Toulouse, France.

Peterson T. and Schindel W. D., "Model-based system patterns for automated ground vehicle platforms," in *INCOSE international symposium*, vol. 25, no. 1. Wiley Online Library, 2015, pp. 388–403.

Pyster A., Olwell D., Hutchison N., Enck S., Anthony J., Henry D., and Squires A., "Guide to the systems engineering body of knowledge (sebok) version 1.0," *Hoboken, NJ: The Trustees of the Stevens Institute of Technology, vol. 852, 2012*.

Rashid M., Anwar M. W., and Khan A. M., "Toward the tools selection in model based system engineering for embedded systems—a systematic literature review," *Journal of Systems and Software*, vol. 106, pp. 150–163, 2015.

Reichwein A. and Paredis C., "Overview of architecture frameworks and Modeling languages for model-based systems engineering," in *Proc. ASME*, 2011, pp. 1–9.

Ronan Barrett, Francis Bordeleau; 5 Years of "Papyrusing"- Migrating Industrial Development from Proprietary Commercial Tool to Papyrus; 2015.

Roques P., Systems Architecture Modeling with the Arcadia Method: A Practical Guide to Capella. Elsevier, 2017.

Sampson M., "Guiding principals for systems engineering tool deployment," *INCOSE International Symposium*, vol. 11, no. 1. Wiley Online Library, 2001, pp. 308–315.

Santiago Jácome, Juan de Lara; Controlling Meta-Model Extensibility in Model-Driven Enginnering; *IEEE* Access; April 25, 2018.

Schmidt D. C., "Model-driven engineering," *IEEE Computer*, vol. 39, no. 2, February 2006. [Online]. Disponible: http://www.truststc.org/pubs/30.html

Shortell T. M., *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities. John Wiley & Sons, 2015*.

Siemens PLM Software "Systems-driven product development-managing the development of complex automotive products through a systems-driven process,", Tech. Rep., 2011.

Silega N., Noguera M., Macías D.; Ontology-based Transformation from CIM to PIM; *IEEE Latin America Transactions, vol. 14, N° 9*, September 2016.

Sven Jäger, Ralph Maschotta, Tino Jungebloud, Alexander Wichmann, and Armin Zimmermann; Creation of Domain-Specific Languages for Executable System Models with the Eclipse Modeling Project; 10th Int. *IEEE Systems Conference (SysCon 2016)*, April 2016, Orlando, Florida.

Tatiana Chuprina, Florian Hölzl, Vincent Aravantinos; Experimental Comparison between AutoFOCUS3 and Papyrus-RT; FORTISS, 2015.

Tatiana Leal del Rio, Gustavo Juárez Gracia, L. Noé Oliva Moreno; Electronic system to monitor the truck route in the Mexico City; *2do Congreso Internacional de Sistemas Embebidos y Mecatrónica*; 978-607-9394-05-9/©2016 AMESE.

Vincent Aravantinos, Sudeep Kanav; Tool Support for Live Formal Verification; 2017.

Viviana Esterkin, Claudia Pons; Evaluación de calidad en el desarrollo de software dirigido por modelos; *Revista chilena de ingenieria, vol. 25 N° 3*, 2017, pp. 449-463.

Voirin J. L., Model-based System and Architecture Engineering with the Arcadia Method. *Elsevier*, 2017.

Voirin J. L., "Modelling languages for functional analysis put to the test of real life," in *Complex Systems Design & Management. Springer*, 2013, pp. 139–150.

Weilkiens T., *SYSMOD-The Systems Modeling Toolbox-Pragmatic MBSE with SysML*. Lulu. com, 2016.

Wortmann A., Combemale B., and Barais O., "A systematic mapping study on modeling for industry 4.0," in *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2017, pp. 281–291.

ANEXOS

Anexo A – Diagramas de la herramienta Capella por fases de diseño

A continuación, se muestran los diagramas de desglose resultantes del desarrollo del sistema IFE por fases de desarrollo.

Fase de Análisis Operacional

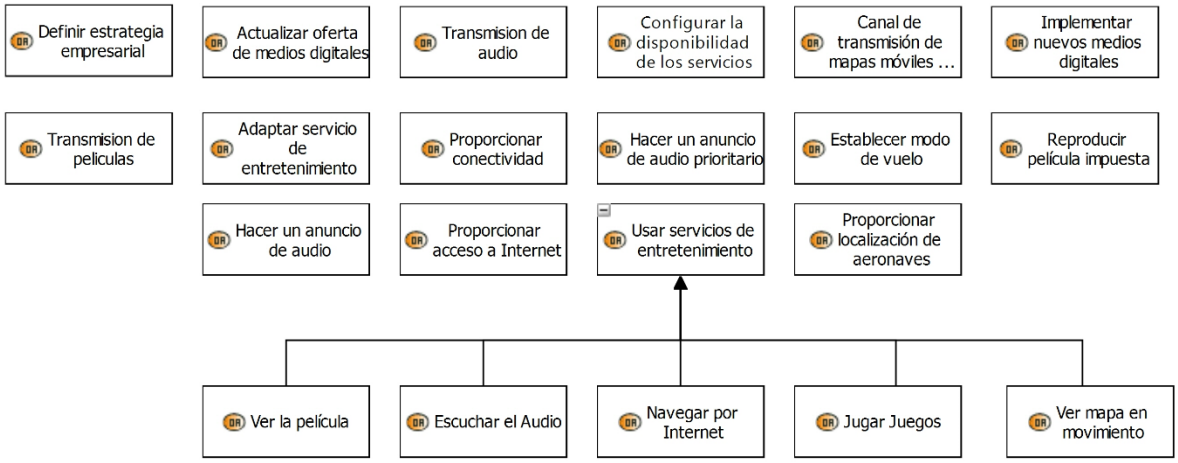


Imagen A.1. Desglose de todas las actividades operacionales.

Fase de Análisis Funcional

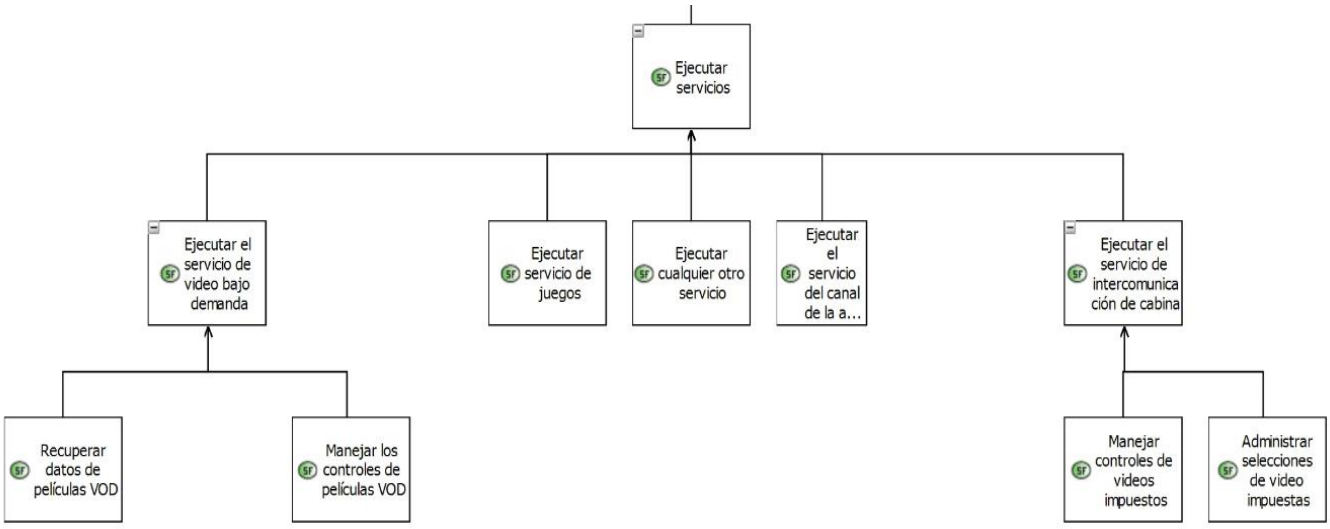


Imagen A.2. Desglose de funciones del sistema.

Fase de Arquitectura Lógica

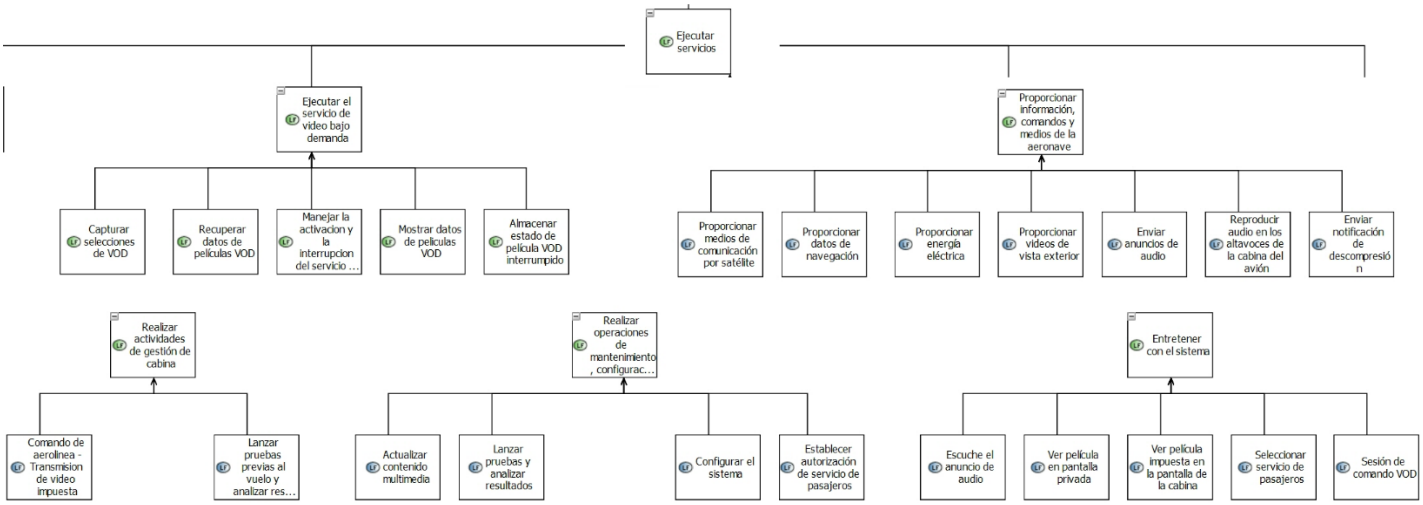
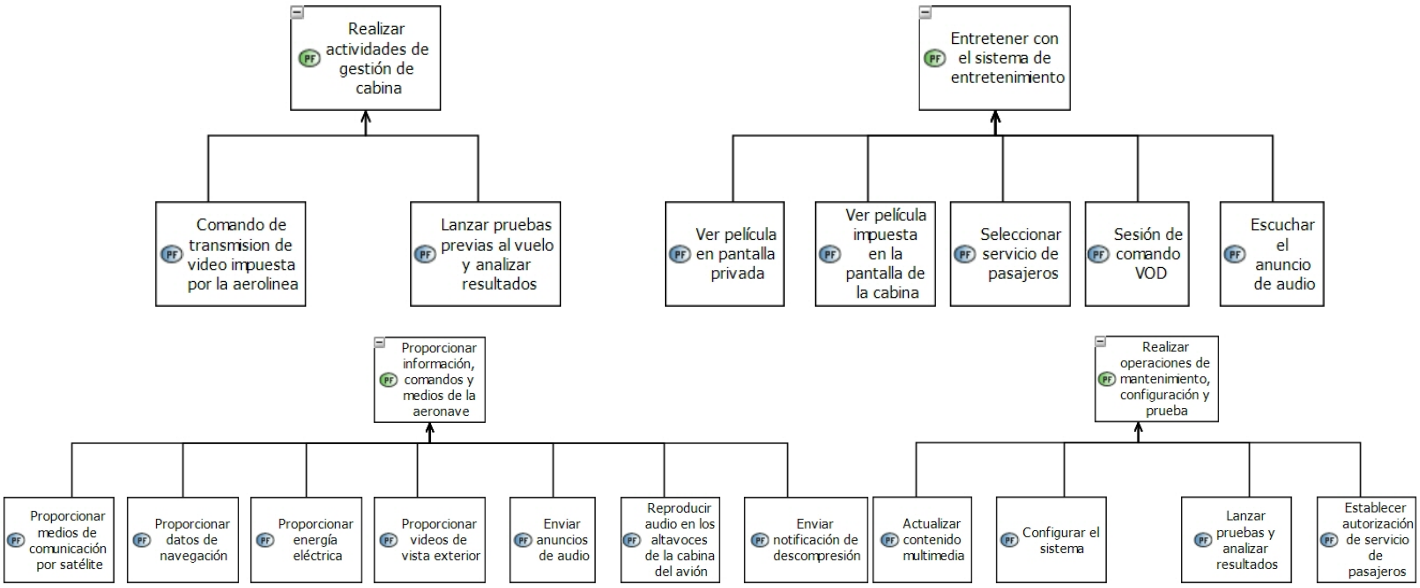


Imagen A.3. Desglose de todas las funciones lógicas.

Fase de Arquitectura Física



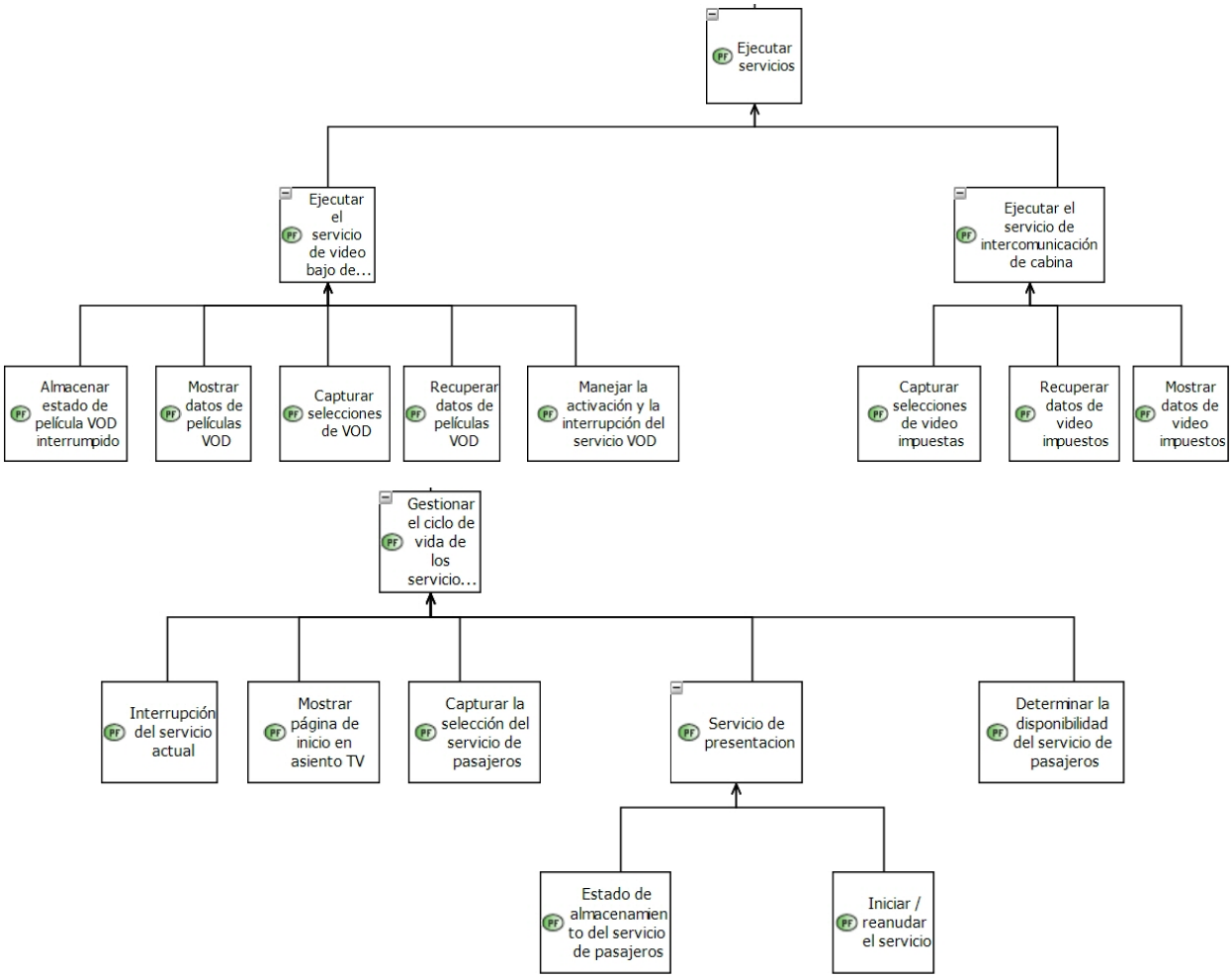


Imagen A.4. Desglose de todas las funciones físicas.

Anexo B – Código generado en la herramienta AutoFocus3 del sistema IFE

A continuación, se muestran los códigos generados, referente a los componentes creados, “sistema”, “Asiento TV” y “Transmisión”.

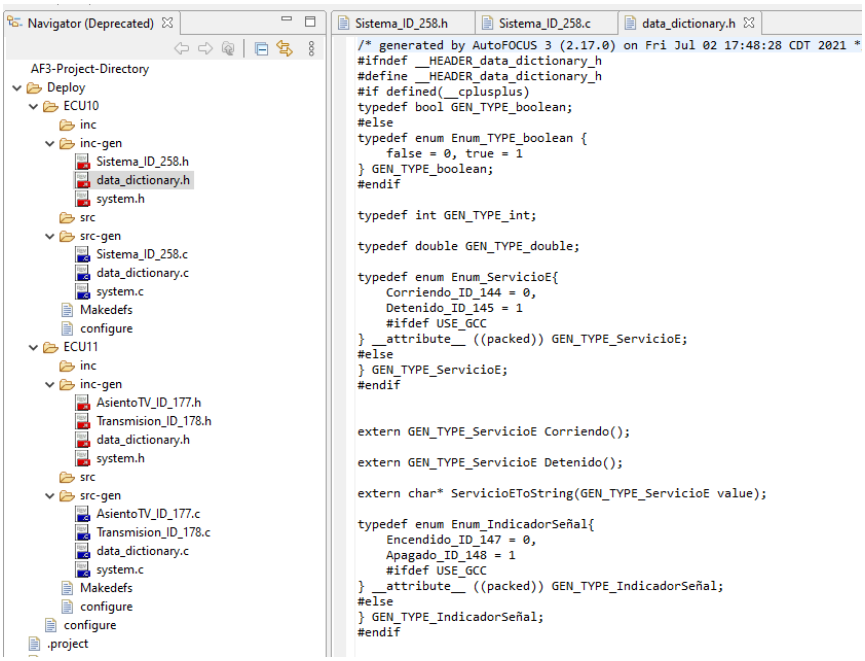


Imagen B.1. Generación de código del diccionario de datos relacionado al componente “sistema con extensión .h” parte 1.

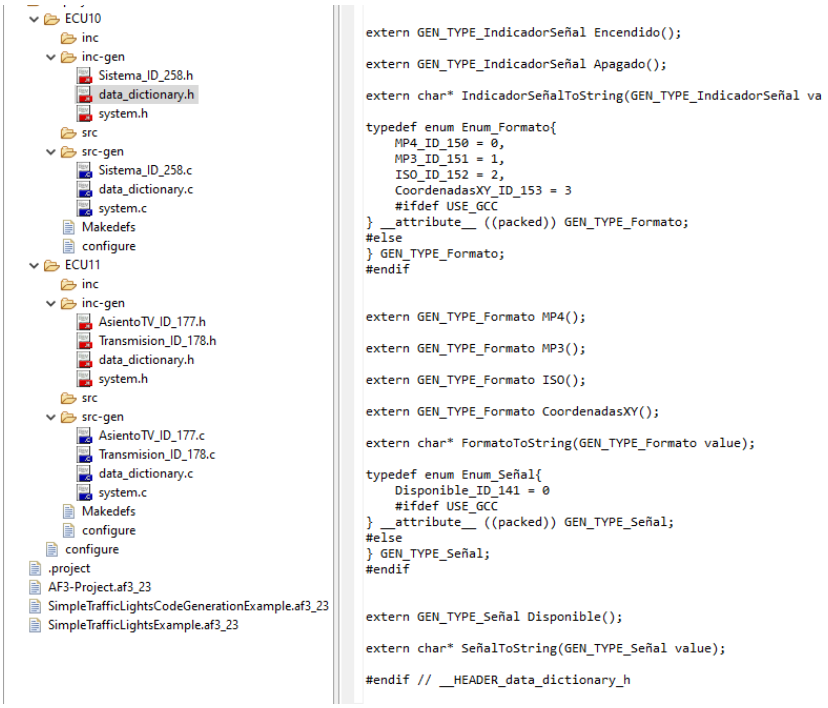


Imagen B.2. Generación de código del diccionario de datos relacionado al componente “sistema con extensión .h” parte 2.

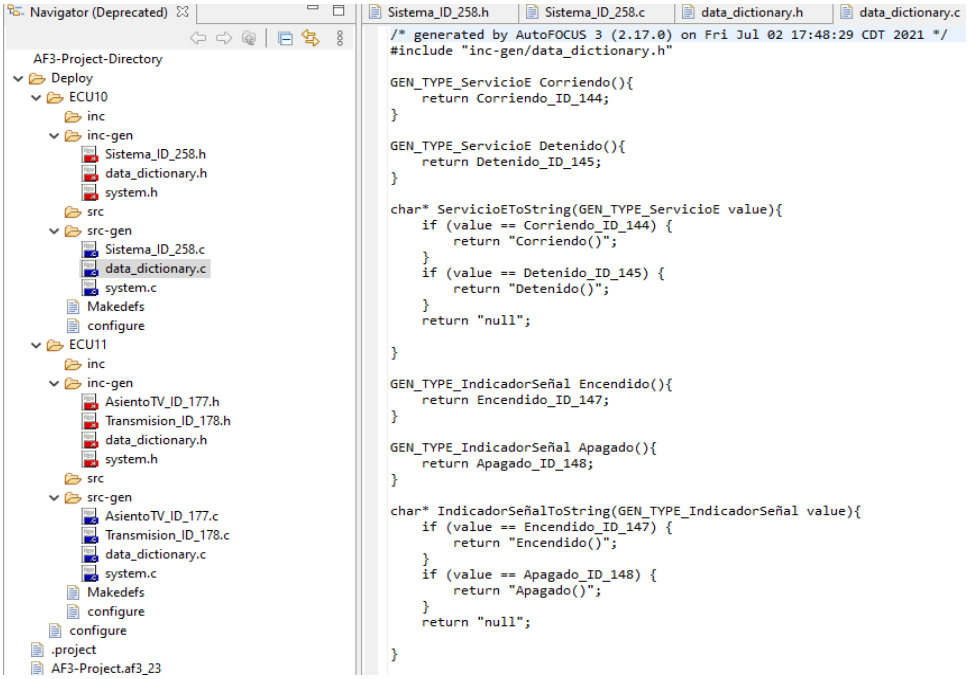


Imagen B.3. Generación de código del diccionario de datos relacionado al componente "sistema con extensión .c" parte 1.

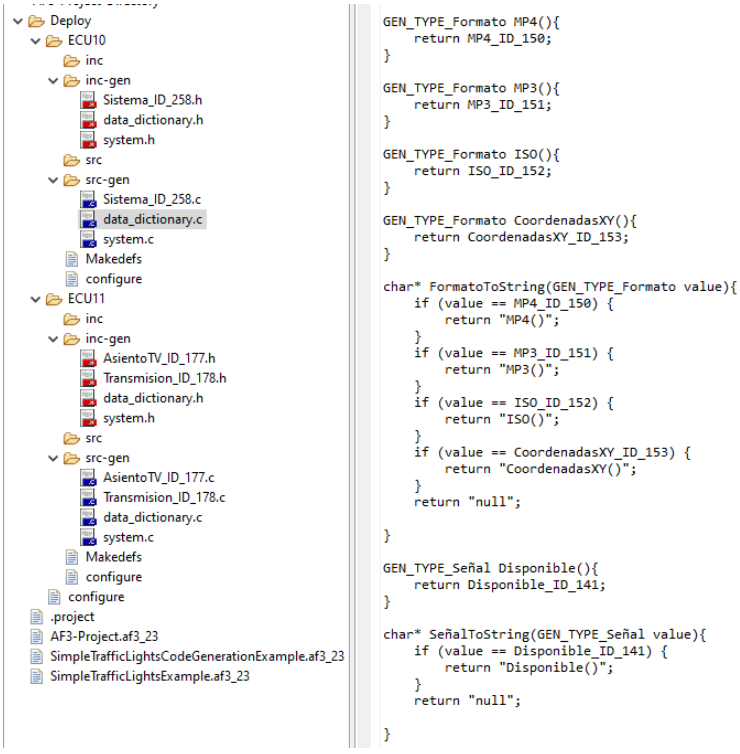


Imagen B.4. Generación de código del diccionario de datos relacionado al componente "sistema con extensión .c" parte 2.

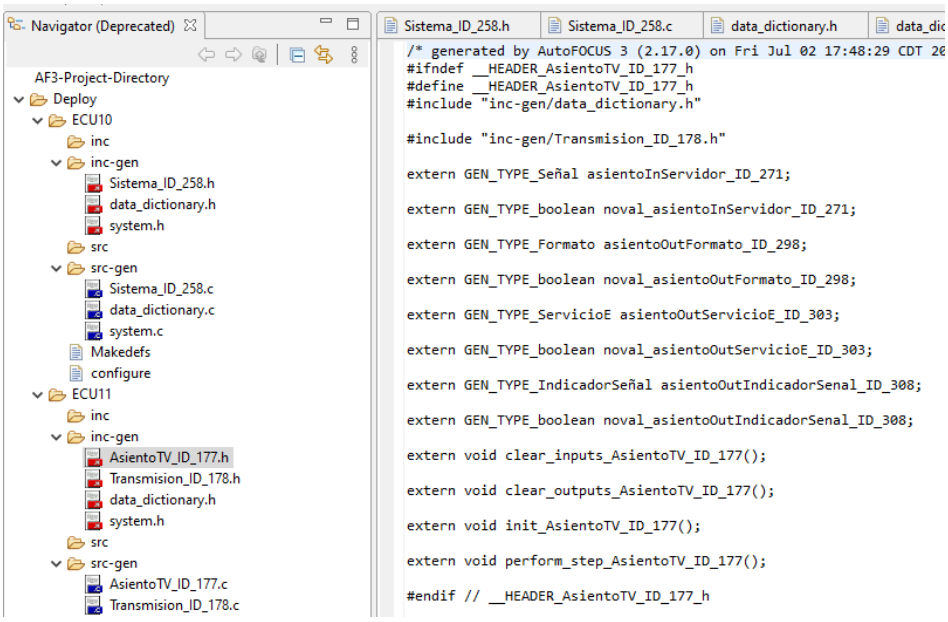


Imagen B.5. Generación de código del componente "Asiento TV con extensión .h".

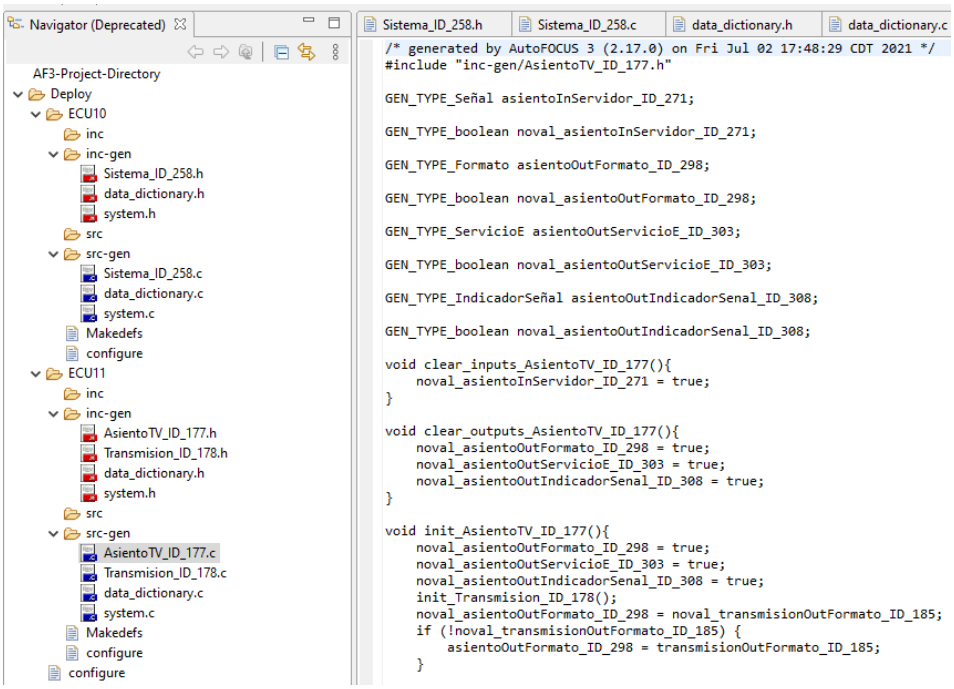


Imagen B.6. Generación de código del componente "Asiento TV con extensión .c" parte 1.

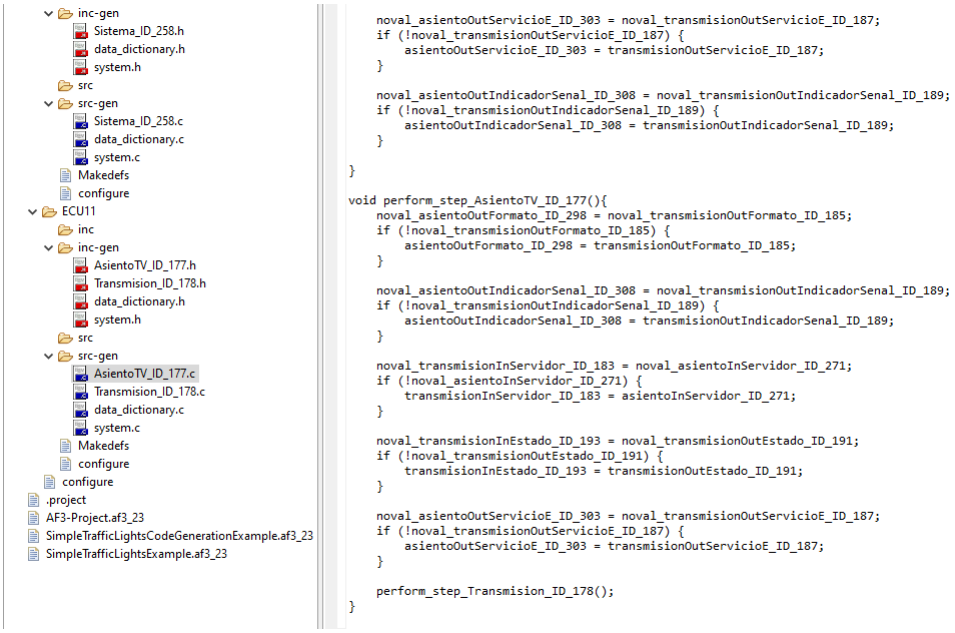


Imagen B.7. Generación de código del componente "Asiento TV con extensión .c" parte 2.

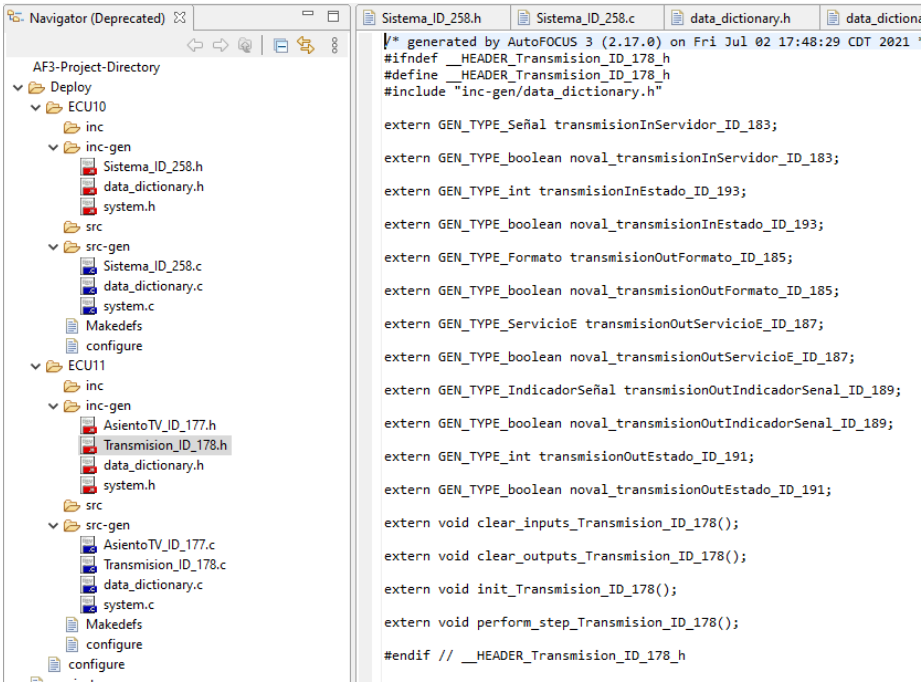


Imagen B.8. Generación de código del componente "Transmisión con extensión .h".

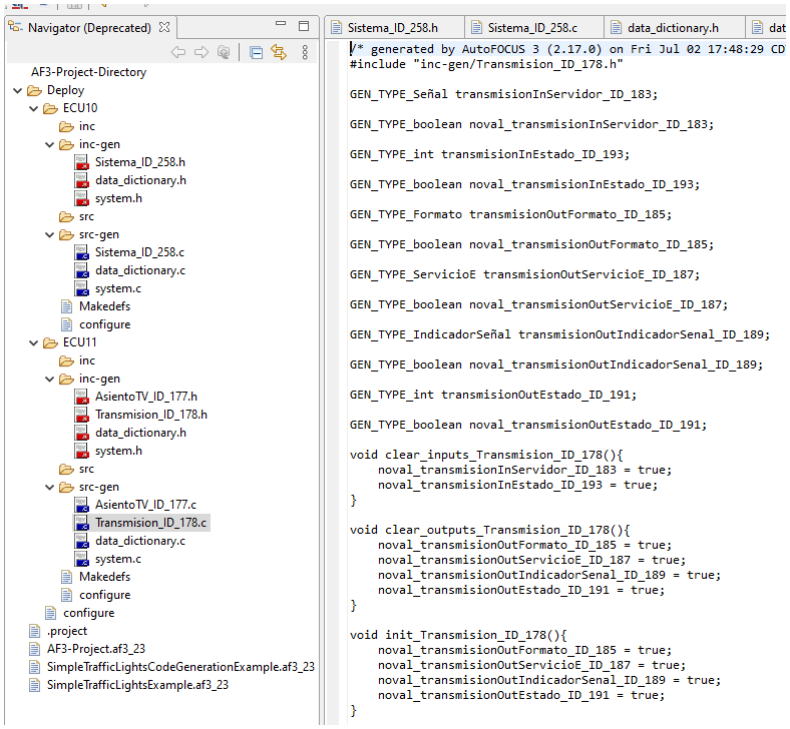


Imagen B.9. Generación de código del componente "Transmisión con extensión .c" parte 1.

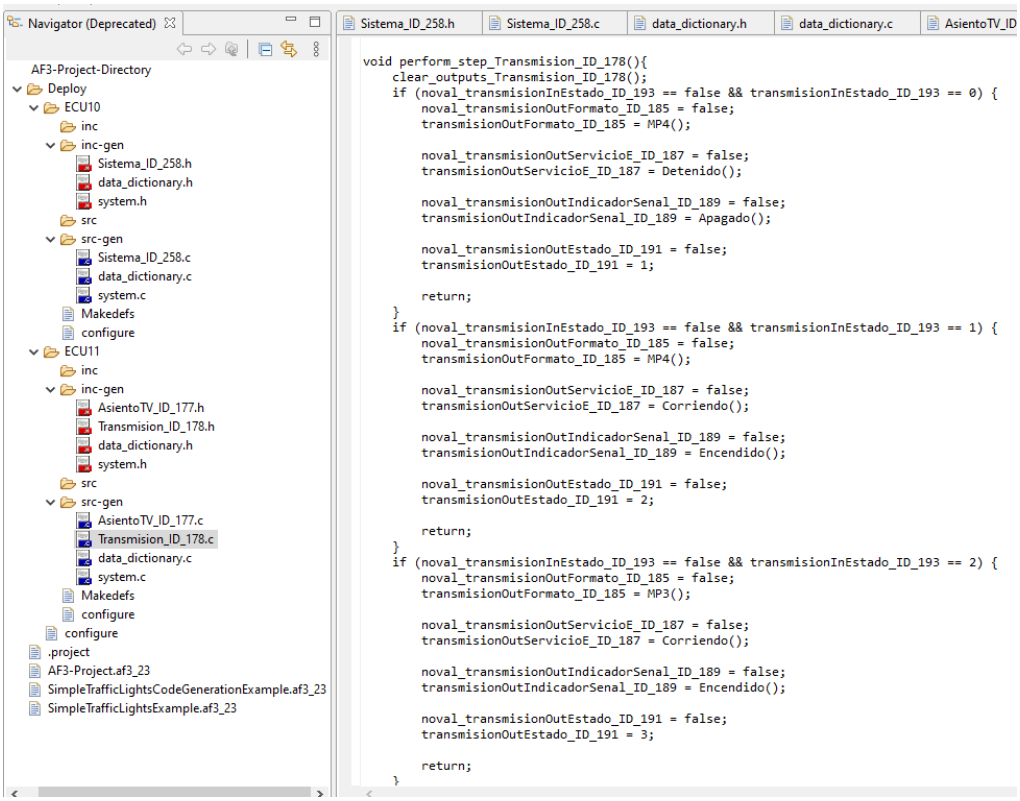


Imagen B.10. Generación de código del componente "Transmisión con extensión .c" parte 2.

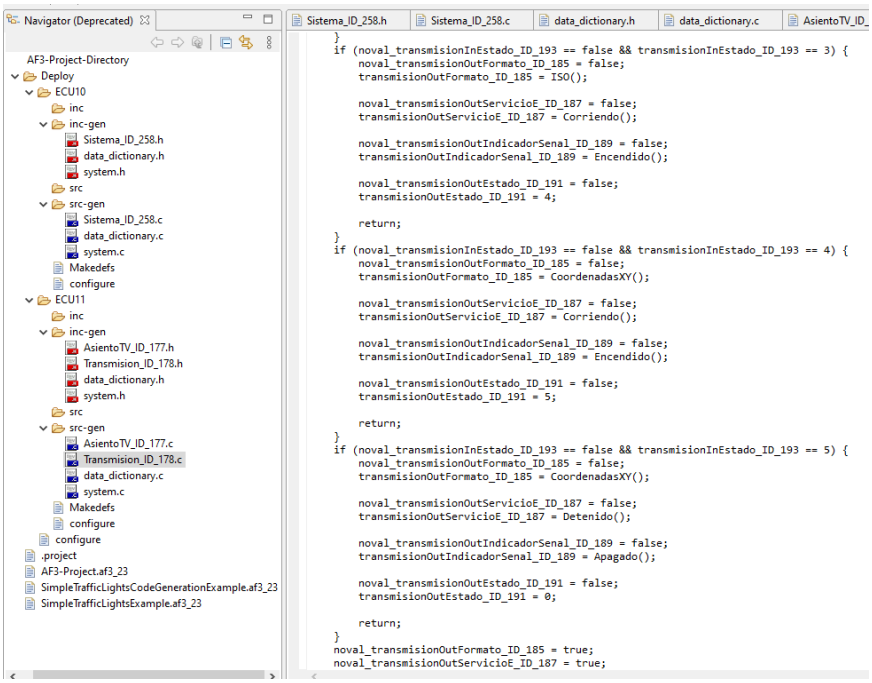


Imagen B.11. Generación de código del componente "Transmisión con extensión .c" parte 3.

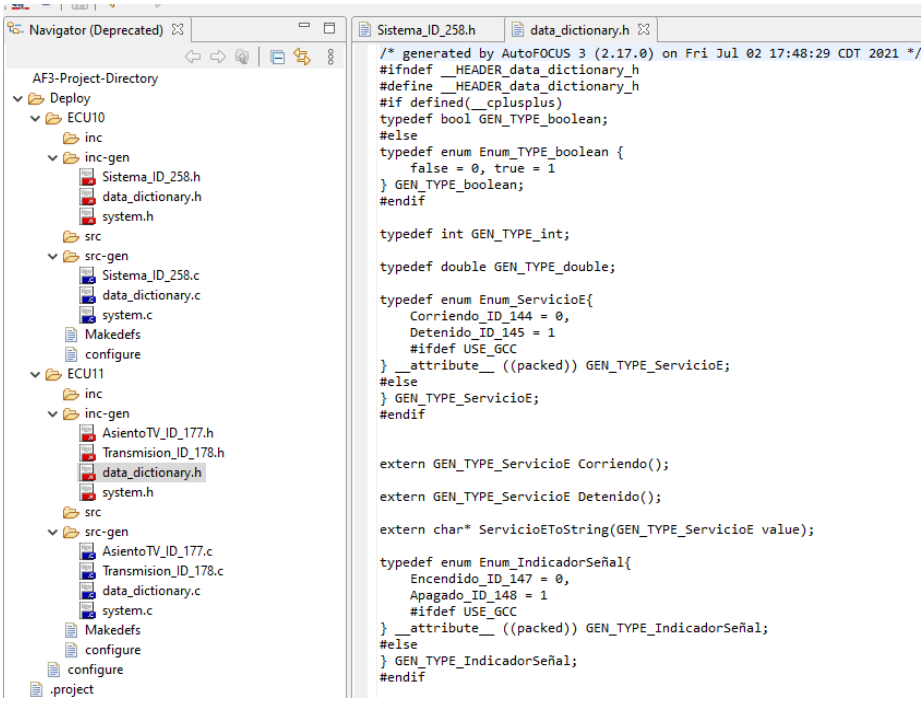


Imagen B.12. Generación de código del diccionario de datos relacionando al componente "Asiento TV y Transmisión con extensión .h" parte 1.

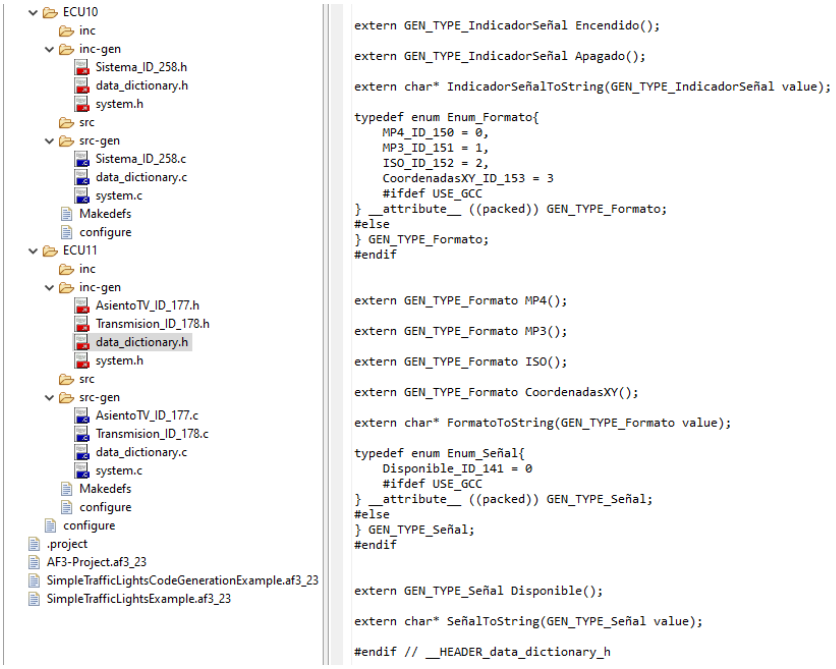


Imagen B.13.Generación de código del diccionario de datos relacionado al componente "Asiento TV y Transmisión con extensión .h" parte 2.

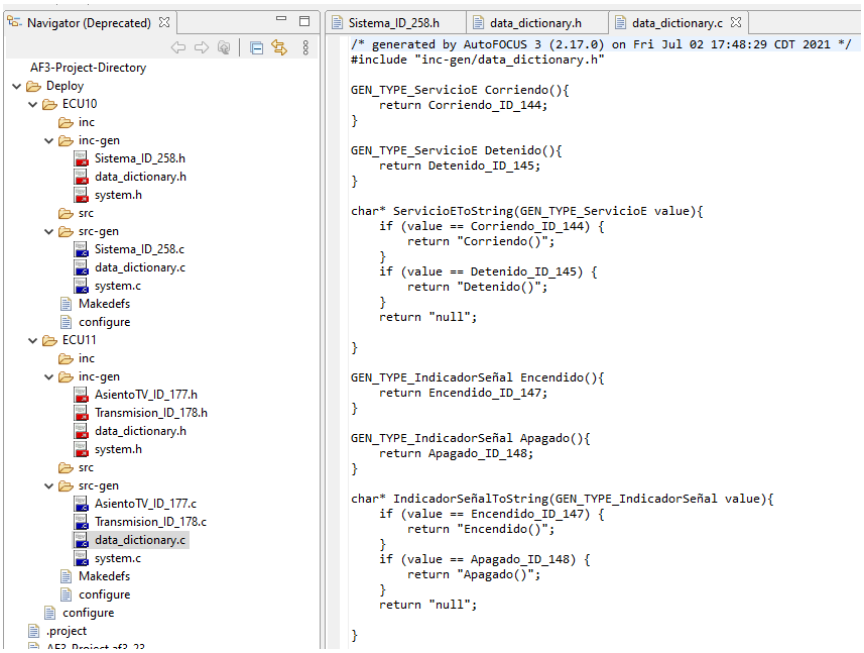


Imagen B.14. Generación de código del diccionario de datos relacionado al componente "Asiento TV y Transmisión con extensión .c" parte 1.

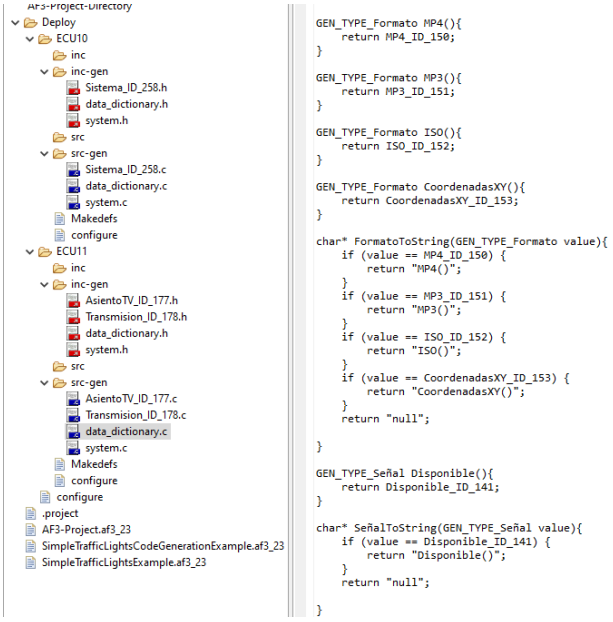


Imagen B.15. Generación de código del diccionario de datos relacionado al componente "Asiento TV y Transmisión con extensión .c" parte 2.

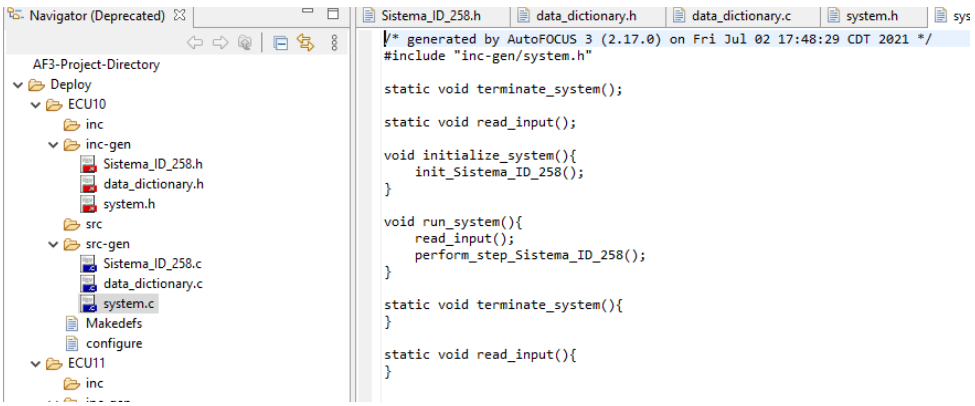


Imagen B.16. Generación de código ejecutable del componente "sistema".

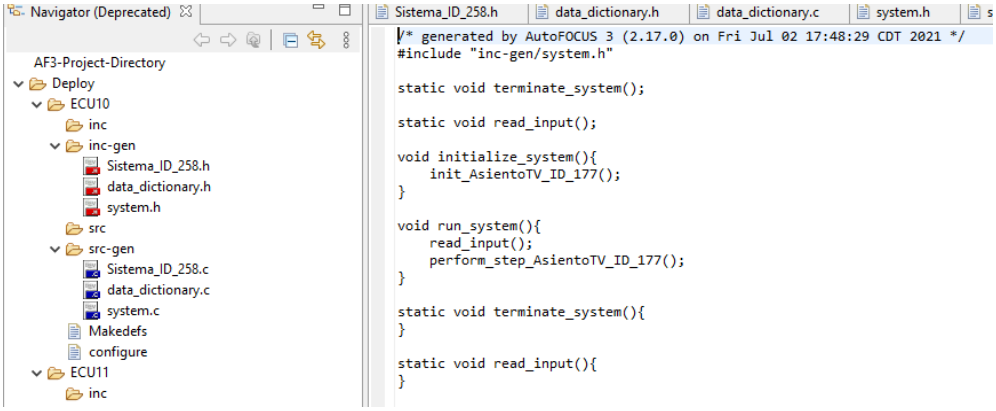


Imagen B.17. Generación de código ejecutable del componente "Asiento TV".

Anexo C – Evaluación de las herramientas candidatas para el desarrollo del sistema IFE.

A continuación, se describen las tablas y resultados de las evaluaciones por criterio, con su respectiva puntuación.

Tabla C.1. Comparación de herramientas por el criterio de desarrollo.

No.	Herramienta	Metodología	Modelado de arquitectura	Transformación de modelos	Generación de código
1	*AndroMDA*	2.5	2.5	0	0
2	*Enterprise Architect*	2.5	2.5	2.5	0
3	*Visual Paradigm*	2.5	2.5	0	0
4	*StarUML*	2.5	2.5	2.5	0
5	*Sirius*	2.5	2.5	0	0
6	*ATL*	2.5	0	2.5	0
7	*VIATRA*	2.5	0	2.5	0
8	*Workflow*	0	0	2.5	0
9	*Epsilon*	2.5	0	2.5	2.5
10	*Acceleo*	0	0	0	2.5
11	*Simulink*	0	2.5	0	0
12	*CDT*	0	0	2.5	0
13	*Papyrus*	2.5	5	2.5	2.5
14	*AutoFocus3*	2.5	2.5	2.5	5
15	*Capella*	5	5	5	0

Tabla C.2. Comparación de herramientas por el criterio de legado.

No.	Herramienta	Edad	Historial	Equipo de desarrollo	Popularidad
1	*AndroMDA*	2.5	2.5	2.5	2.5
2	*Enterprise Architect*	2.5	2.5	2.5	5
3	*Visual Paradigm*	2.5	2.5	2.5	2.5
4	*StarUML*	2.5	2.5	2.5	0
5	*Sirius*	2.5	2.5	0	0
6	*ATL*	2.5	2.5	0	0
7	*VIATRA*	2.5	2.5	2.5	0
8	*Workflow*	0	2.5	0	0
9	*Epsilon*	2.5	2.5	2.5	2.5
10	*Acceleo*	2.5	2.5	0	2.5
11	*Simulink*	2.5	2.5	2.5	2.5
12	*CDT*	2.5	0	0	0
13	*Papyrus*	2.5	5	2.5	2.5
14	*AutoFocus3*	2.5	5	5	2.5
15	*Capella*	5	2.5	5	5

Tabla C.3. Comparación de herramientas por el criterio de actividad.

No.	Herramienta	Comunidad contribuyente	Actividad en los fallos	Actividad en las funcionalidades	Actividad en las versiones
1	*AndroMDA*	0	0	0	0
2	*Enterprise Architect*	2.5	2.5	2.5	2.5

3	*Visual Paradigm*	2.5	2.5	2.5	5
4	*StarUML*	5	2.5	2.5	2.5
5	*Sirius*	2.5	2.5	2.5	0
6	*ATL*	2.5	2.5	2.5	2.5
7	*VIATRA*	2.5	2.5	2.5	2.5
8	*Workflow*	0	0	0	0
9	*Epsilon*	2.5	2.5	0	0
10	*Acceleo*	2.5	2.5	0	2.5
11	*Simulink*	2.5	2.5	2.5	2.5
12	*CDT*	0	0	2.5	2.5
13	*Papyrus*	2.5	0	2.5	2.5
14	*AutoFocus3*	2.5	2.5	2.5	2.5
15	*Capella*	2.5	2.5	2.5	2.5

Tabla C.4. Comparación de herramientas por el criterio de industrialización.

No.	Herramienta	Servicios	Documentación	Aseguramiento de calidad	Modificación código fuente
1	*AndroMDA*	2.5	2.5	2.5	0
2	*Enterprise Architect*	2.5	5	2.5	2.5
3	*Visual Paradigm*	2.5	2.5	2.5	2.5
4	*StarUML*	2.5	5	2.5	2.5
5	*Sirius*	2.5	5	2.5	2.5
6	*ATL*	2.5	5	2.5	2.5
7	*VIATRA*	2.5	5	2.5	2.5
8	*Workflow*	2.5	0	2.5	0
9	*Epsilon*	0	5	2.5	2.5
10	*Acceleo*	2.5	5	2.5	2.5
11	*Simulink*	2.5	5	2.5	0
12	*CDT*	0	2.5	2.5	0
13	*Papyrus*	5	5	2.5	2.5
14	*AutoFocus3*	5	2.5	2.5	2.5
15	*Capella*	5	2.5	5	2.5

Tabla C.5. Comparación de herramientas por el criterio de características.

No.	Herramienta	Extensible	Simplicidad	Multiformato de datos	Multiplataforma
1	*AndroMDA*	0	2.5	0	5
2	*Enterprise Architect*	2.5	5	5	2.5
3	*Visual Paradigm*	2.5	2.5	2.5	0
4	*StarUML*	2.5	5	2.5	2.5
5	*Sirius*	2.5	2.5	2.5	0
6	*ATL*	2.5	2.5	2.5	2.5
7	*VIATRA*	0	2.5	2.5	2.5
8	*Workflow*	0	0	0	2.5
9	*Epsilon*	2.5	2.5	0	2.5
10	*Acceleo*	2.5	2.5	2.5	2.5
11	*Simulink*	0	2.5	2.5	2.5
12	*CDT*	0	0	2.5	0
13	*Papyrus*	2.5	2.5	2.5	5
14	*AutoFocus3*	5	2.5	5	5
15	*Capella*	5	5	0	2.5