Casa abierta al tiempo

PC y TI

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Iztapalapa

División de Ciencias Básicas e Ingeniería

Posgrado en Ciencias y Tecnologías de la Información

“Arquitectura autonómica para sistemas de software de apoyo al diagnóstico médico”

Idónea comunicación de resultados que para obtener el grado de:

Maestro en Ciencias y Tecnologías de la Información

Presenta:

Lic. Narmo Dioney Reyes Cardoso
Matrícula: 2173802305

Asesores:

Dra. Angelina Espinoza Limón
Lic. Luis Fernando Castro Careaga

Sinodales:

Lic. Luis Fernando Castro Careaga
M.en.C. Marco Antonio Núñez Gaona
Dra. Reyna Carolina Medina Ramírez

Iztapalapa, Ciudad de México, a 13 de julio del 2020.

Agradecimientos

A mis padres, Elia y Aristeo

El texto en una tesis no bastaría para expresarles mi admiración y agradecimientos por su apoyo incondicional, consejos oportunos y, sobre todo, su cariño constante. Los quiero con toda mi alma.

Dra. Angelina Espinoza Limón

Por su inmensa paciencia y su imbatible espíritu docente; por tantas horas de dedicación y tantos consejos que no solo han hecho salir adelante a este proyecto, sino también a mí como persona.

A mis sinodales

Por sus atinadas observaciones, recomendaciones y por su muy amable atención y retroalimentación brindadas para la redacción y corrección de esta tesis.

A mi familia y amigos.

Por hacer mi vida mas alegre e interesante y llenarme de cariño, especialmente le agradesco a mi novia Maricela por el apoyo incondicional que me ha brindado desde el inicio de esta maestría.

"La inteligencia es la capacidad de adaptarse al cambio"

Índice general

Índice de figuras

Lista de tablas

Resumen:

Los sistemas actuales cada vez se vuelven más difíciles de integrar, instalar, configurar y mantener, además del gran costo que esto conlleva. La computación autonómica busca solucionar estos problemas mediante el desarrollo de sistemas que permita la auto-gestión de ellos mismos. Los sistemas de información para el cuidado de la salud (HCIS, por sus siglas en inglés "Health-Care Information System"), son sistemas que se desarrollan para apoyar el diagnóstico médico y presentan estas características que pretende afrontar la computación autonómica. La arquitectura de un sistema es un conjunto de estructuras relacionadas entre sí, que comprende elementos de software y que permite diseñar la construcción del sistema.

En el Capítulo 1 se presenta la motivación y objetivos. En el Capítulo 2, se presenta el análisis del estado del arte, realizando un Estudio de Mapeo Sistemático (SMS, por sus siglas en inglés "Sistematic Mapping Study"), relacionado con los modelos de arquitectura de computación autonómica para desarrollar sistemas de información de apoyo al diagnóstico médico. En el Capítulo 3, se presentan los antecedentes, metodología y conceptos para diseñar el meta-modelo de arquitectura. En el Capítulo 4, se presenta el diseño del meta-modelo de arquitectura autonómica para sistemas de software de apoyo al diagnóstico médico. En el Capítulo 5, se presenta un proceso de instanciación para el meta-modelo de arquitectura. En el Capítulo 6, se presenta la evaluación del meta-modelo de arquitectura. En el Capítulo 7, se presenta el trabajo a futuro y por último en el Capítulo 8 se presentan las conclusiones.

Capítulo 1

Introducción

El uso de la tecnología en los sistemas de información para el cuidado de la salud (HCIS, "Health-Care Information System"), ha experimentado un gran aumento en los últimos años. Debido al crecimiento tecnológico se han desarrollado sistemas que se pueden implementar en dispositivos móviles, computadoras portátiles y otros dispositivos que generan una gran cantidad de información que alimentan a los sistemas médicos. Uno de los principales problemas que tienen estos sistemas es el manejo de grandes volúmenes de datos y la heterogeneidad de estos mismos, ya que al recibir datos de diferentes fuentes de información realizar la integración de estos es sumamente complicado. A diferencia de otros sistemas, estos también están rezagados en términos de gestión de información, seguridad, almacenamiento e infraestructura que ayude a modificar, instalar, configurar, mantener y diseñar nuevos componentes dentro del sistema. Por lo que existe una gran urgencia de desarrollar soluciones que permitan generar sistemas de información para el cuidado de la salud de alta calidad.

Además, estos sistemas HCIS permiten llevar la administración de los hospitales, pacientes, médicos y farmacias de una manera controlada. En este campo de investigación se requiere tener una gran colaboración de médicos, pacientes y expertos para lograr resultados relevantes.

La Computación Autonómica [0] (CA, por sus siglas en inglés "Autonomic Computing"), busca solucionar este tipo de problemas, mediante la creación de sistemas auto-gestionados. Estos sistemas abordan problemas actuales como complejidad y costo [1], [20].

La CA es una propuesta lanzada por IBM en el año 2001, esta está inspirada en el sistema nervioso humano el cual regula y mantiene la homeostasis de los esfuerzos inconscientes del cerebro. De manera análoga permite que los sistemas y aplicaciones de computación se administren así mismo con la mínima intervención humana. La CA diseña sistemas autónomos e inteligentes mediante elementos autónomos. Estos son componentes básicos de la autonomía de los sistemas, que a través de sus interacciones mutuas producen el comportamiento general de autogestión [17].

La iniciativa de la computación autonómica [2], tiene un fuerte enfoque en la administración de sistemas complejos a través de la automatización de tareas basadas en el patrón MAPE-K (Mapea, Analiza, Planea, Ejecuta, Conocimiento). Este patrón ha sido ampliamente utilizado para diseñar sistemas auto-configurados que automáticamente adoptan su estructura y comportamiento en función de los cambios de contexto.

En trabajos existentes de computación autonómica aplicada a sistemas de información para el cuidado de la salud como en [2], [3], [4], se abordan los problemas de los sistemas mediante el diseño de la arquitectura utilizando estilos y patrones de arquitectura, con la finalidad de desarrollar sistemas flexibles, inteligentes y escalables. Pero estos diseños son deficientes en términos de arquitectura, por la falta de diseños eficientes que utilicen una notación estándar para representar la arquitectura, además de que son diseños a muy alto nivel por lo que realizar un cambio a un sistema repercute en tiempo y costo.

Un modelo genérico de arquitectura con características autonómicas de apoyo al diagnóstico médico es una base para diseñar sistemas de software, con la finalidad de construir sistemas específicos a partir de un diseño genérico. El desarrollo de un sistema de arquitectura autonómico de apoyo al diagnóstico médico genérico beneficiará a los arquitectos disminuyendo los tiempos y costos del diseño, aumentando la calidad de los sistemas de software de apoyo al diagnóstico médico.

En esta tesis, se presenta una revisión detallada de estudios en el área de arquitectura de software y computación autonómica aplicada a sistemas de información de apoyo al diagnóstico médico. Enfocado en el diseño de la arquitectura que implementa características autonómicas y que sirve en el área médica. Para realizar este análisis se propone la metodología de Estudio de Mapeo Sistemático. Se han documentado todos los pasos del EMS para obtener el resultado deseado de acuerdo con los objetivos de investigación. Se presenta una clasificación de acuerdo con criterios definidos y estadísticas que describen la evidencia disponible sobre el tema de modelos de arquitectura con características autonómicas de apoyo al diagnóstico médico.

Existen varias definiciones de arquitectura de software, pero aquí vamos a considerar la definida por Bass, Clements y Kazman [22], que estipula lo siguiente.

"La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema, considerando elementos de software, las relaciones entre ellos y las propiedades de ambos".

1.1 Motivación del proyecto

Los sistemas actuales cada vez son más difíciles de integrar, instalar, configurar y mantener, además del gran costo que esto conlleva. La computación autonómica busca la solución de estos problemas mediante el desarrollo de sistemas que permita la autogestión de ellos mismos. Específicamente IBM propone cuatro principios 1) configuración de sistemas. 2) seguridad de sistemas. 3) optimización de sistemas. 4) curación de sistemas.

Los sistemas de información para el cuidado de la salud (HCIS), son sistemas que se desarrollan para apoyar el diagnóstico médico y que presentan las características de los sistemas actuales más las características particulares de los sistemas médicos. Estos sistemas que son requeridos por hospitales necesitan que los sistemas sean implementados con la mínima intervención humana debido a los problemas asociados al mantenimiento de los sistemas, por el alto valor de la información. También se requiere tener sistemas que siempre estén activos y si en un momento de operación el sistema llegase a fallar, se requiere que se configure el sistema de repuesto o se repara el sistema en el menor tiempo posible.

Por lo tanto, EMS va a dirigir el estado del arte de trabajos en sistemas HCIS con computación autonómica, en especial el diseño de arquitectura que estos presentan.

Objetivos

1.1.1 Objetivo general

Definir un modelo de arquitectura autonómico para sistemas de información de apoyo al diagnóstico médico.

1.1.2 Objetivos particulares

I. Investigar las propuestas relacionadas a los modelos autonómicos para sistemas de información en el cuidado a la salud, en particular en el apoyo al diagnóstico médico.
II. Definir un modelo de arquitectura autonómico para sistemas de información de apoyo al diagnóstico médico, para una especialidad médica.
III. Realizar la verificación del modelo de arquitectura, considerando una metodología para evaluación de diseño de arquitectura (por Ejemplo, ATAM (Kazman et. al, 2000)).

Capítulo 2

Estado del Arte

2.1 Estudio de Mapeo Sistemático (EMS)

El EMS es una metodología diseñada para realizar una amplia investigación general en un área particular, ofreciendo evidencias sobre los temas de interés y permitiendo cuantificar la evidencia existente [14].

El EMS define un estudio primario y secundario para generar estudios de literatura de alta calidad. De acuerdo con Kitchenham & Charters un estudio primario es: *Crear cadenas de búsqueda estructuradas con los términos de población y términos relevantes impulsados por la pregunta de investigación*. Un estudio secundario es: *Realizar el análisis de todos los estudios primarios relacionados con la pregunta de investigación*.

2.2 Estructura de la metodología EMS

Aplicando la metodología EMS propuesta por Kitchenham et al. [13], para realizar el estado del arte siguiendo la siguiente estructura:

1. Pregunta de investigación: definición y validación.
2. Estrategia de búsqueda de estudios primarios:
   a. Búsqueda automatizada.
   b. Selección de motores de búsqueda.
   c. Conformación de cadenas de búsqueda.
   d. Refinamiento de cadenas de búsqueda.
3. Selección de estudios primarios relevantes.
4. Definición de criterios de clasificación y análisis.

5. Análisis de los estudios primarios.
6. Documentación de resultados.

2.3 Ejecución de la metodología EMS

2.3.1 Definición de la pregunta de investigación

La etapa 1 del EMS, considera identificar las siguientes áreas de investigación: Arquitectura de software, Computación autonómica y Sistemas de información de apoyo al diagnóstico médico. involucradas en la definición de la pregunta de investigación de esta tesis. La pregunta de investigación se creó bajo el enfoque de los temas de interés siguiendo el área de investigación de computación autonómica, arquitectura y sistemas médicos.

La etapa 2 considera definir sub-preguntas a partir de la pregunta de investigación para tener un mejor enfoque del tema y delimitar la investigación. Las preguntas definidas se encuentran en la *Tabla 1 y 2*.

Tabla 1. Pregunta de investigación

ID	Pregunta de investigación	Objetivo de la pregunta
PI-1	¿Existen modelos de arquitectura con características autonómicas para sistemas de información en el cuidado de la salud que apoyen al diagnóstico médico?	Buscar modelos de arquitectura con características autonómicas

Tabla 2. Sub-Pregunta de investigación

ID	Sub-Pregunta de investigación	Objetivo de la pregunta
SPI-1	¿Qué drivers arquitectónicos se tomaron en cuenta en las arquitecturas relacionadas con los HCIS?	Encontrar drivers arquitectónicos dentro de los HCIS
SPI-2	¿Qué atributos de calidad se tomaron en cuenta para soportar las características autonómicas?	Encontrar atributos de calidad
SPI-3	¿Se tomaron en cuenta conceptos de diseño como patrones o tácticas para soportar las características autonómicas?	Encontrar patrones que soporten características autonómicas
SPI-4	¿Qué características autonómicas implementan los sistemas revisados?	Encontrar características autonómicas

SPI-5	¿Qué estilo de arquitectura implementan para diseñar el comportamiento autonómico?	Encontrar estilos de arquitectura

2.3.2 Definición de los términos relevantes

Los términos relevantes se tomaron en cuenta bajo el enfoque de los temas de interés definiendo así la **POBLACIÓN**, **INTERVENCIÓN**, **RESULTADOS RELEVANTES** y **DISEÑO EXPERIMENTAL**. Estos términos se pueden apreciar en la *Tabla 3*.

2.3.3 Búsqueda de sinónimos de las palabras claves

Se tomaron en cuenta los términos relevantes para realizar la cadena de búsqueda, pero además también se consideraron los sinónimos de las palabras claves para cubrir todas las posibilidades al realizar la búsqueda de los artículos. Estos términos se pueden encontrar en la *Tabla 4*.

2.3.4 Búsqueda de estudios primarios

Se seleccionaron las bibliotecas digitales para realizar la búsqueda de los estudios primarios. Estas bibliotecas fueron seleccionadas por la facilidad de búsqueda y el acceso que se tiene.

2.3.5 Bibliotecas digitales

- IEEE XPlore
- ACM DIGITAL LIBRARY
- Springer
- Scopus

Tabla 3. Términos relevantes de la pregunta de investigación

Pregunta de investigación	Población	Intervención	Resultados relevantes	Diseño experimental
PI-I	HCIS, Ingeniería de software, medical, diagnosis.	Arquitecturas autonómicas.	Modelos de Arquitectura. Modelos de arquitectura con características autonómicas.	Cualquier método

Tabla 4. Sinónimos de las palabras claves

Palabras Claves	Sinónimos
HCIS	Healthcare information system, medical, health system, medical system, patient
Computación Autonómica	Autonomic system, autonomic elements Sistemas Autonómicos, Autonomic technologies
Modelos de arquitectura	Architecture design, Architecture diagrams, Architecture views

2.3.6 Construcción de la expresión de búsqueda

Para realizar la expresión de búsqueda o cadena de búsqueda se tomaron en cuenta las palabras claves dentro del campo de POBLACIÓN, los sinónimos y RESULTADOS RELEVANTES, junto con las expresiones booleanas AND, OR, NOT, y las características de cada motor de búsqueda se conformaron las expresiones de búsqueda. La *Tabla 5* muestra las palabras claves y sinónimos utilizados.

La *Tabla 6* muestra la conformación de las expresiones adaptadas a cada motor de búsqueda.

Tabla 5. Palabras claves y sinónimos en términos de población y resultados relevantes

	Términos más importantes	Sinónimos y términos alternativos
Población	HCIS, Software engineering, medical diagnosis, autonomic computing	Healthcare, health-care, Medical, medical diagnosis, patients
Intervención	Arquitecturas autonómicas	
Resultados de relevancia	Architecture system model, architecture style, quality attributes, application domain, self-features, design patterns	Architecture views, Architecture diagrams, System, software, Self-*
Diseño experimental	Cualquier método	

Tabla 6. Expresiones de búsqueda

Library	PI-I	Resultados
IEEE	((("Document Title":.QT.healthcare.QT. OR "Document Title":.QT.health-care OR "Document Title":.QT.medical diagnosis.QT. OR "Document Title":.QT.medical.QT.) AND ("Document Title":.QT.autonomic.QT. OR "Document Title":self*)) AND ("Abstract":.QT.architecture.QT. OR "Abstract":.QT.model.QT. OR "Abstract":.QT.autonomic.QT.)) NOT ("Abstract":.QT.biology.QT. OR "Abstract":.QT.networks.QT. OR "Abstract":.QT.cloud.QT. OR "Abstract":.QT.wireless.QT.)	*Journals 3* *Conferences 17* *Books 0*
ACM	"query": { (+Healthcare +Medical +Software +engineering +System +Autonomic +computing +Architecture+Patient+Model) }	Journals 1 Conferences 4 Books 0
Springer	'Software AND o AND engineering AND o AND System AND o AND Architecture AND model AND o AND Self-adaptive AND (Healthcare OR o OR Medical OR o OR OR Architecture)'	Journals 0 Conferences 9 Books 16
Scopus	**(TITLE (**autonomic**) AND TITLE-ABS-KEY (** model **) AND ALL (** healthcare **) AND ALL (** patients **) AND ALL (**	Journals1 Conferences 2 Books 0

	medical) **AND ALL (** architectural) **AND NOT TITLE-ABS-KEY (** biology) **AND NOT TITLE-ABS-KEY (** cloud) **AND NOT TITLE-ABS-KEY (** wireless))	
Total		Journals 5 Conferences 32 Books 16

2.3.7 Selección de los estudios primarios relevantes

Para realizar la selección de los estudios primarios relevantes se establecieron criterios de inclusión y exclusión, para enfocar la búsqueda en términos de POBLACIÓN y los RESULTADOS RELEVANTES y eliminar aquellos resultados que no aportan información a la investigación. En la *Tabla 7 y 8* se definen los criterios de exclusión e inclusión.

Tabla 7. Criterios de exclusión

ID	Criterios de Exclusión	Número de artículos eliminados	Total, de artículos 53
CE-1	El estudio primario no esté escrito en español o inglés	0	53
CE-2	El estudio primario sea inferior a la fecha 2006	1	52
CE-3	El artículo no esté escrito en términos de POBLACIÓN y TÉRMINOS RELEVANTES	22	30
CE-4	El trabajo es inferior a 5 páginas	2	28
CE-5	El artículo se encuentre repetido	7	21

Tabla 8. Criterios de Inclusión.

ID	Criterios de inclusión	Número de artículos aceptados	Total, de artículos 21
CI-1	El abstract hace referencia a los términos de la columna RESULTADOS RELEVANTES en la población indicada en el protocolo	21	21
CI-2	Los artículos son de revista.	7	14
CI-3	Artículos de conferencia centrados en el campo de población.	8	6

CI-4	El título hace referencia a los términos de la columna RESULTADOS RELEVANTES en la población indicada en el protocolo.	6	0

2.3.8 Criterios de clasificación de los estudios primarios

Los resultados arrojados de las cadenas de búsqueda y la aplicación de los criterios de inclusión y exclusión fueron clasificados de la siguiente manera.

1. Clasificación por contribución en porcentajes de acuerdo con el contenido de los términos de población y los términos relevantes en el título y abstract se clasifican en Alto, Medio-Alto, Medio-Bajo y Bajo. En la *Tabla 9* se muestran los criterios de clasificación.
2. Clasificación por el tipo de documento en: revistas, conferencias y capítulos de libros con la finalidad de dar prioridad a los artículos de revista.
3. Clasificación por año de publicación con la finalidad de ver el avance o el trabajo realizado.

2.4 Resultados

El resultado de aplicar el EMS al estudio de interés que es generar un modelo de arquitectura con características autonómicas aplicado a los sistemas de información de apoyo al diagnóstico médico ha arrojado un total de 53 estudios primarios, a los cuales se les han aplicado los criterios de inclusión y exclusión dando como resultado un total de 21 estudios primarios prioritarios.

Tabla 9. Criterios de clasificación

Términos de población	Términos relevantes	Contribución	Criterios	Porcentaje
Healthcare, Medical, Medical diagnosis, autonomic, patient.	Architecture system model, Architecture style, Quality attributes, Application domain, Self-features, Design patterns	Alta	Los artículos contienen todos los términos del campo de población en el título o abstract y contienen los términos relevantes en el abstract o conclusiones y están centrados en el campo de intervención.	<80 %
		Media Alta	Los artículos contienen ¾ partes de los términos del campo de población en el título o abstract y contienen los términos relevantes en el abstract o conclusiones y están centrados en el campo de intervención.	60 % >80%
		Media Baja	Los artículos contienen la mitad de los términos del campo de población en el título o abstract y contienen los términos relevantes en el abstract o conclusiones y están centrados en el campo de intervención.	40% > 60%
		Baja	Los artículos contienen menos de la mitad de los términos del campo de población en el título y el abstract y	20% > 40%

contienen los términos relevantes en el abstract o conclusiones y están centrados en el campo de intervención.

2.5 Estadísticas

En esta parte se presentan las estadísticas de los criterios de clasificación. La *Tabla 10* presenta la cantidad de estudios primarios obtenidos para la pregunta de investigación (PI-I), arrojados por cada motor de búsqueda. La Figura 1 muestra el refinamiento de la selección de estudios primarios.

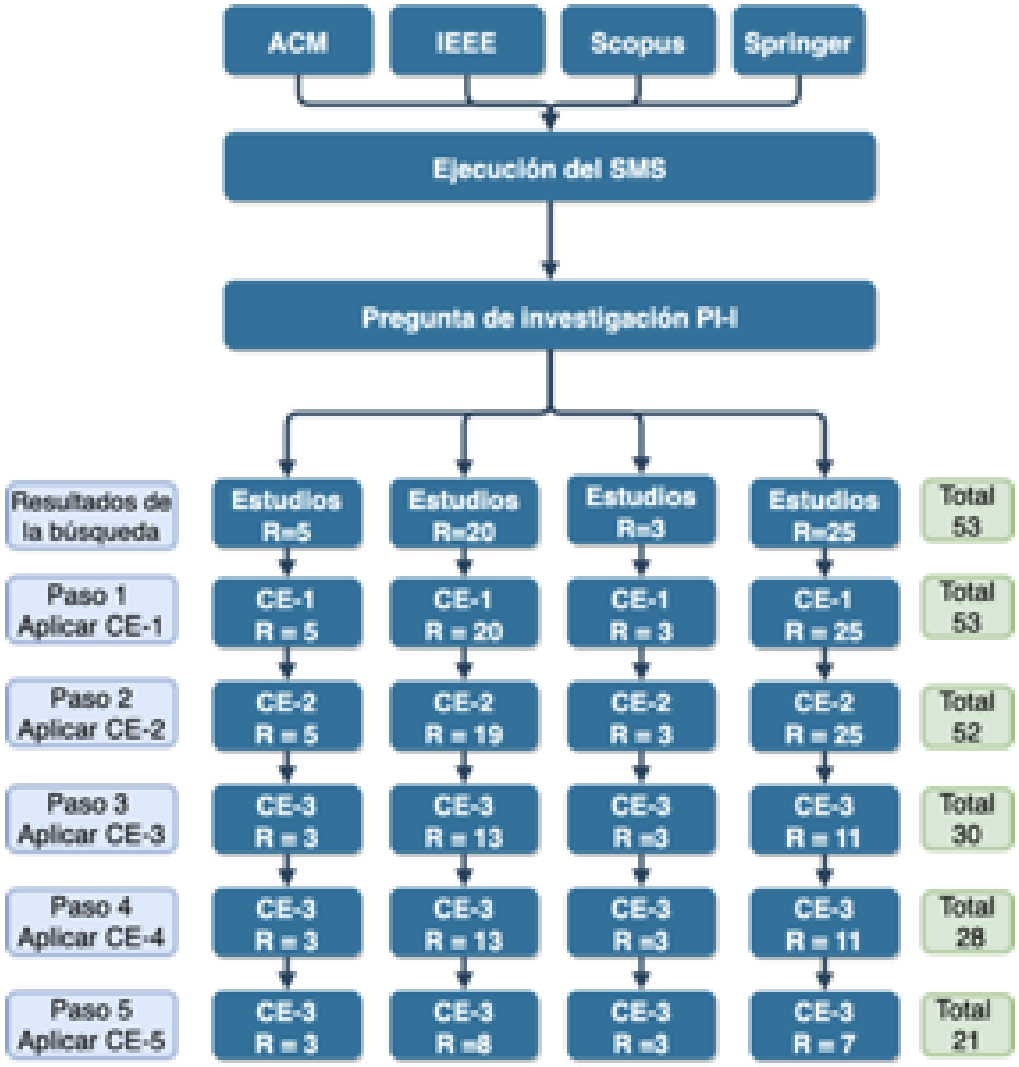

Figura 1. Criterios de clasificación de artículos.

Tabla 10. Estudios Primarios

Motor de búsqueda	Pregunta de investigación
ACM	5
IEEE	20

Scopus	3
Springer	25
Total	53

2.6 Clasificación por contribución de los términos

2.6.1 Clasificación por aportación

En la Figura 2 se muestra la contribución de los artículos en función de los términos de POBLACIÓN y TÉRMINOS RELEVANTES (*Tabla* 11 Clasificación por prioridad), como se puede observar solo el 19% de los artículos tiene prioridad alta, por otro lado, el 43% tiene prioridad baja.

Figura 2. Clasificación por contribución de artículos.

Tabla 11. Clasificación por prioridad.

Prioridad	Total de Artículos	Porcentaje de contribución.
Alta	4	19%
Media-Alta	3	14.3%
Media-Baja	5	23.7%
Baja	9	43%
Total	21	100%

2.6.2 Clasificación por tipo de documento

La Figura 3 ilustra la clasificación por tipo de estudio (Revista, Conferencia, Capítulo de libro) asignando las siguientes prioridades de acuerdo con la *Tabla 12*.

Figura 3. Clasificación por tipo de estudio.

Tabla 12. Clasificación por tipo de estudio

Tipo de estudio	Total de artículos	Porcentaje de contribución
Revistas	6	28.5%
Conferencias	13	62%
Capítulos de libro	2	9.5%
Total	21	100%

2.6.3 Clasificación por año de publicación

La Figura 4, indica el año de publicación de los 21 artículos analizados, la gráfica muestra que el año donde se realizaron más estudios de computación autonómica de apoyo al diagnóstico médico fue el 2012, después bajó considerablemente hasta el año 2017 donde fue retomado.

Tabla 13. Clasificación por año de publicación

Año de publicación	Número de artículos	Porcentaje
2005-2010	7	33.2%
2011-2015	9	42.8%
2016-2018	5	24%
Total	21	100%

Figura 4. Clasificación por año de publicación.

2.7 Análisis de los estudios primarios: Estado del arte

El estudio primario realizado bajo la metodología EMS brinda respuesta a la pregunta de investigación PI-I y a las sub-preguntas generadas a partir de la PI-I, por lo que en esta sección se analizarán y se discutirán los estudios primarios que aportan información para el desarrollo de un modelo de arquitectura con características autonómicas de apoyo al diagnóstico médico. Cada artículo analizado se presenta en la *Tabla 11*.

De los estudios primarios se realizó el análisis siguiendo el siguiente enfoque de búsqueda: drives arquitectónicos, atributos de calidad, patrones o tácticas, estilos de arquitectura y características autonómicas para dar respuesta a la pregunta de investigación PI-I.

2.7.1 Análisis de Modelos y Patrones de arquitectura

Drira et al. [1], presenta una metodología impulsada por modelos que hacen uso de patrones de arquitectura con la finalidad de desarrollar un sistema de monitorización flexible y cognitivo para el manejo de la salud de los pacientes. El objetivo de estos patrones de arquitectura es lograr un sistema ampliamente flexible, escalable e inteligente. La metodología impulsada por modelos hace uso de patrones que combinan principios de modelado de software e ingeniería de conocimientos para facilitar el desarrollo de los sistemas inteligentes. En la *Tabla 14* se muestra el análisis de estos patrones. Dentro de esta metodología se identifican dos fases principales. 1) Identificación de requerimientos. 2) Formalización de requerimientos.

Requerimientos de identificación: Es la discusión con los expertos de los requerimientos del sistema, aquí se tienen requerimientos funcionales y no funcionales.

Formalización de requerimientos: se centra en formalizar y estructurar los requisitos identificados en modelos concretos que describen las interacciones de los procesos del sistema. Este está basado en el patrón MAPE-K y se constituye de 5 componentes: Monitoreo, Análisis, Plan, Ejecución y Conocimiento. **El monitoreo:** Se encarga de recopilar, agregar, correlacionar y filtrar información para determinar un síntoma que posteriormente será analizado. **Análisis:** Realiza razonamiento de los datos sobre los síntomas proporcionado por la función del Monitor. **Plan:** Estructura las acciones necesarias para lograr las metas y los objetivos. **Ejecución:** Realiza los pasos a seguir de acuerdo con la etapa de ejecución. **Conocimiento:** Recolecta la información en una base de datos, aprende y genera un nuevo conocimiento (ver Figura 5).

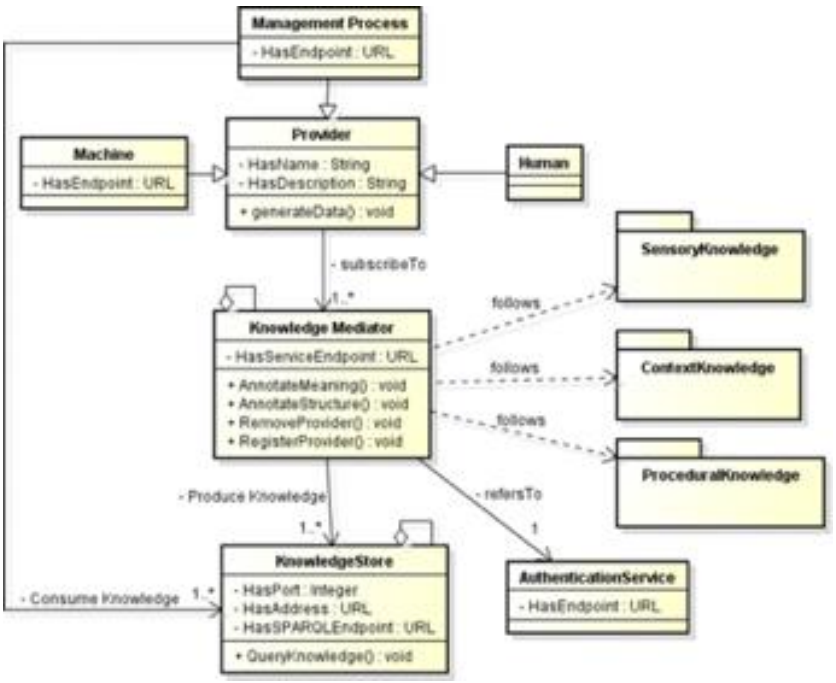

Figura 5. Patrón: Mediador del conocimiento semántico [5].

Tabla 14. Análisis de patrones de arquitectura propuesto por Drira et.al. [1]

Patrones.	Atributos de calidad	Auto Características	Descripción
Cognitive Monitoring Management pattern	Interoperabilidad, Escalabilidad	Auto-Configuración	• La integración de nuevos dispositivos a la red genera nuevos datos no necesariamente heterogéneos. • El patrón permite la interacción con humanos expertos, extraer nuevos conocimientos evitando

			tener dispositivos específicos que solo analizan un tipo de dato en específico.
Predictive Cognitive Management Pattern		Auto-Optimización	Modela la coordinación entre el monitoreo, los procesos de análisis y el experto. También con los procesos de gestión de conocimiento sensorial, para generar nuevos conocimientos sobre el elemento gestionado.
Prescriptive Cognitive Management Pattern	Usabilidad		Coordina el monitoreo, análisis y los procesos mientras interactúan con los expertos. También genera recomendaciones para que los expertos tomen mejores decisiones.
Autonomic Cognitive Management Pattern			Propone descubrimientos dinámicos de procesos de gestión basados en el contexto del sistema para gestionar el tiempo de ejecución de la evaluación de los requerimientos del sistema
Knowledge Pattern	Eficiente, Disponibilidad	Auto-Optimización	• Propone descomponer el conocimiento centralizado para evitar cuellos de botella y escalabilidad. Se divide en tres sub-componentes que son: conocimiento sensorial, contexto y procedimental. • Sensorial: describe una forma semántica de datos y las características para la captura de datos. • Contexto: Describe semánticamente el elemento gestionado. • Procedimental: Describe semánticamente el conocimiento que incluye los procedimientos para resolver problemas.
Patrón mediador del conocimiento semántico Figura 6:	Eficiente, Escalable, Confiabilidad	Auto-configuración, Auto optimización	El patrón permite la colaboración de diferentes tipos de proveedores (humanos, máquinas) al poblar la base de conocimiento, basado en el patrón de conocimiento

Bendiab et al. [5], describe cada uno de estos componentes aplicados a la medicina, redefiniendo el modelo MAPE-K propuesto por Kephart en [9]. A continuación, se describen

cada uno de los elementos modificados por Bendiab. **Monitoreo de salud:** selecciona los dispositivos médicos que recaban la información de los pacientes. **Analizador médico:** analiza los datos recolectados para predecir el diagnóstico del paciente. **Plan médico:** decide el plan médico para el manejo del caso del paciente. **Ejecución:** implementa los planes sugeridos, en esta parte se realiza la interacción entre el sistema, paciente y médico. **Conocimiento médico:** representa la ontología del sistema de base de información en el formato semántico. El sistema de conocimiento se encarga de vincular todos los componentes del sistema. La Figura 6 ejemplifica el modelo propuesto.

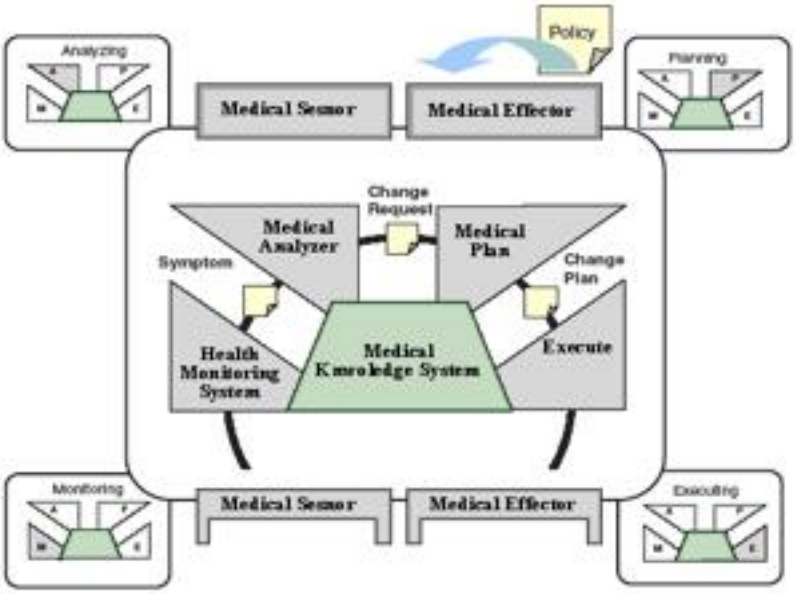

Figura 6. Modelo MAPE-K aplicado a HCIS [9].

Lasierra et al. [2], propone un modelo ontológico llamado HOTMES (Home Ontology for inTegrated Management in home-based Scenarios), basado en computación autonómica para afrontar el desafío de integrar, analizar y evaluar grandes volúmenes de datos. Una ontología constituye un enfoque teórico perfecto para realizar la integración de los datos, proporcionando un marco orientado al conocimiento dónde se pueden mapear fácilmente diferentes fuentes de datos. Las ontologías se usan para expresar el conocimiento de una manera estática y declarativa como un conjunto de cosas que son verdaderas. En general, las soluciones desarrolladas combinan el conocimiento estático con el conocimiento dinámico de las ontologías presentando el uso de reglas de razonamiento.

La ontología propuesta pretende afrontar los problemas de integración de datos, cuando estos provienen de diferentes dispositivos: unificar, acceder, controlar, evaluar y transferir información mediante el diseño de reglas junto con una instancia del perfil de gestión. La computación autonómica implementa un bucle de control inteligente que consiste en recopilar datos, analizar los datos, determinar un plan o secuencia de acciones en respuesta al análisis y finalmente ejecutar el plan. Mediante el uso de computación autonómica (ver Figura 7), se propone una ontología que ayuda a la integración de datos y a los procedimientos de gestión de información. La *Tabla 15* describe el modelo propuesto.

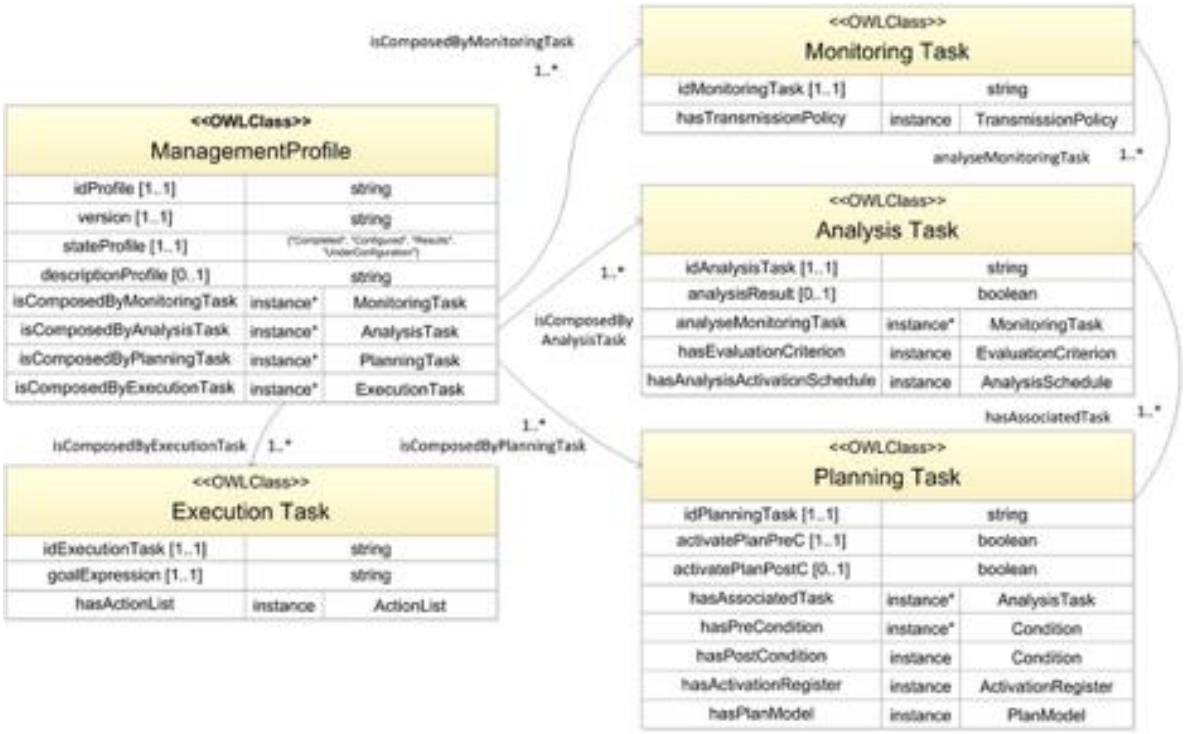

Figura 7. Modelo de ontología [2].

Urovi et al. [18], propone un modelo ontológico para los registros electrónicos de salud (EHR). Este modelo pretende afrontar los problemas de la recopilación de datos de pacientes, la transferencia y la toma de decisiones mediante el uso de reglas de inferencia. El modelo también describe semánticamente las bases de conocimiento de las comunidades de salud y coordina sus interacciones en el intercambio EHR entre distintas ontologías.

De la misma manera Taweel et al. [19], propone una arquitectura en capas enfocándose en un modelo ontológico para los EHR para afrontar diversos problemas como son: la heterogeneidad de los datos, la entrega de información en cualquier punto clínico, la falta de protocolos para el intercambio de información y la seguridad de los datos. La heterogeneidad de los datos ha resultado en diferencias significativas entre los EHR en términos de la representación de datos, la comunicación y la interoperabilidad entre ellos. Las complejidades de interoperabilidad ocurren en varias formas como son: sintácticas, semánticas, estructurales y de seguridad. Con el uso de ontologías se proporcionará una visión segura y coherente de los datos de diferentes sistemas permitiendo políticas de acceso e intercambio de datos independientes y diferentes. Permitir cambios independientes en la fuente de datos o sistemas logrará una integración transparente o perfecta de las fuentes de datos. La Figura 8 muestra la arquitectura propuesta.

Figura 8. Arquitectura RHM [18].

La utilización de la tecnología de CA puede ayudar a los Sistemas de Información de Salud (HIS por sus siglas en inglés “Health Information system”), a anticipar, detectar y responder a ataques conocidos y desconocidos con la mínima intervención humana. Un HIS auto-protector tendrá un rendimiento confiable y resistente incluso si está comprometido por ataques cibernéticos.

Chen et al. [3], propone el modelo de auto-protección HIS (ver Figura 9), el modelo como primera línea de defensa, anticipa las anomalías venideras, envía alertas tempranas al controlador y protege el sistema al indicarle al controlador que inicie una respuesta activa a las amenazas seleccionadas. La funcionalidad de cada módulo se analiza de la siguiente manera: Risk Assessment: Evalúa los riesgos y ayuda a organizar y determinar el impacto y la probabilidad de que una amenaza determinada pueda explotar con éxito una vulnerabilidad en particular. Intrusión estimation: Este anticipa los próximos ataques al comparar la "región normal" con los resultados estimados del modelo del sistema. Intrusion Detection: Este módulo analiza el comportamiento de los datos detectando anomalías dentro del sistema,

mediante una herramienta de minería de datos en tiempo real. Intrusión response: Este modelo permite que HIS se auto-proteja para mitigar los impactos de ataque y regular el comportamiento del sistema a la normalidad.

Tabla 15. Modelo autonómico propuesto por Lasierra et.al. [2].

Ontología	Patrón	Componentes	Utilización
HOTMES	MAPE	Monitorización.	Realizar una tarea de monitoreo generalmente significa recopilar información de un sistema, probarla y conocer su estado.
		Análisis	Una tarea de análisis proporciona un conjunto de procesos y reglas que permiten observar una situación para determinar si algo ha cambiado o necesita cambiarse. La tarea de análisis descrita en la ontología propuesta explica cómo razonar con los datos de supervisión adquiridos definiendo diferentes métodos de análisis.
		Planeación	Una tarea de planificación puede definirse como una formulación detallada de una secuencia de tareas y actividades para resolver un problema de decisión secuencial.
		Ejecutor	Una tarea de ejecución se puede definir como un conjunto de acciones realizadas por un agente específico. En este modelo, la tarea de ejecución se refiere a las acciones que debe realizar el administrador o el EM para reaccionar a una determinada situación o resolver un problema detectado.

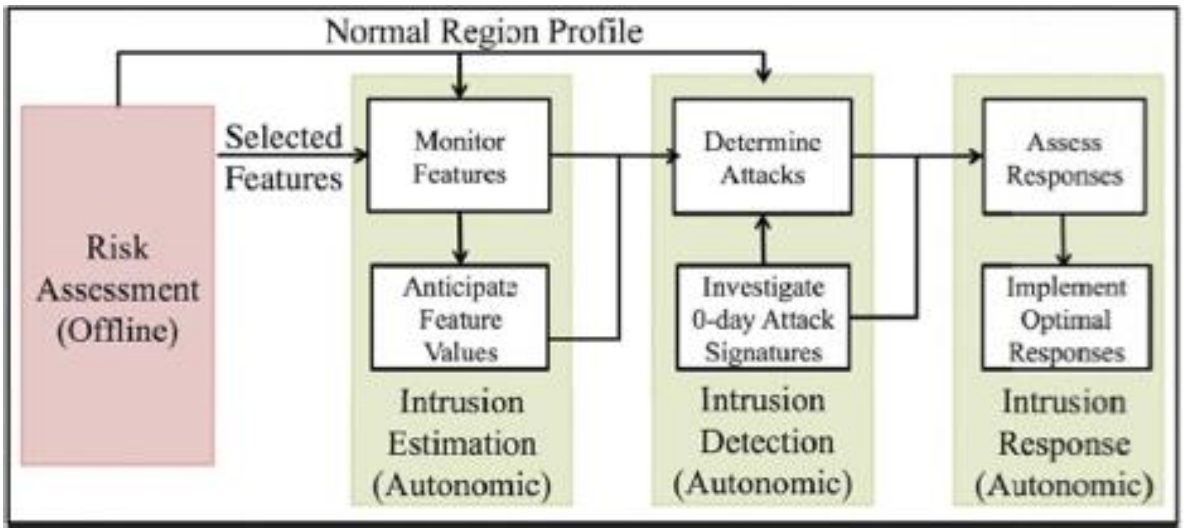

Figura 9. Modelo de auto-protección [3].

Sloman et al. [7], presenta un sistema llamado *Vesta* que afronta los desafíos como: la administración y seguridad de sistemas mediante la aplicación de un patrón de arquitectura llamado: célula auto-gestionada (SMC, por sus siglas en inglés “Self Managed Cell"). La Figura 10 presenta la arquitectura del sistema V*esta*, basada en el patrón SMC.

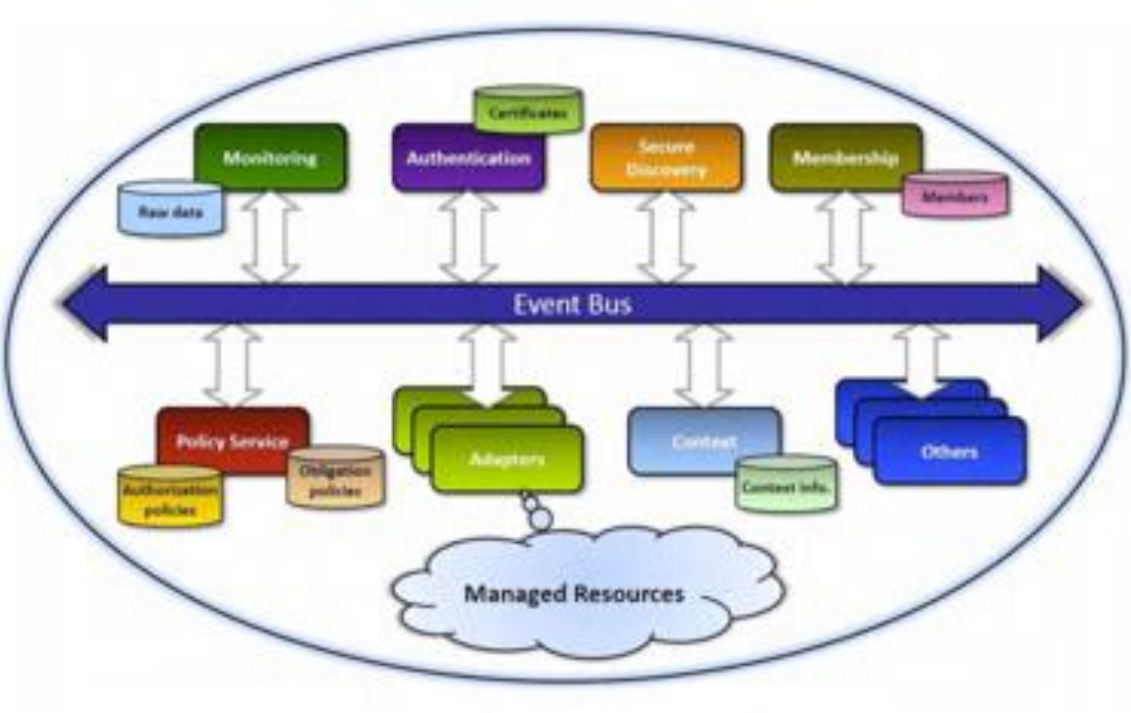

Figura 10. Arquitectura de una célula auto-gestionada [7].

Zang et al. [4], propone un diseño de nube autonómico para la gestión de los servicios de atención médica. Estos servicios podrían beneficiarse del entorno de la nube al aumentar la calidad de los servicios, brindar atención rápida al paciente y distribuir el servicio para todos los pacientes en cualquier lugar. Para diseñar el sistema se aplica una arquitectura orientada a servicios (SOA por sus siglas en inglés "Service Oriented Architecture") que permite minimizar los costos de servicios para muchos usuarios. El diagrama de arquitectura tiene un diseño de muy alto nivel como se muestra en la Figura 11.

Chandrasekharan et al. [6], propone una solución para el problema de acceder a los registros médicos del paciente y recuperar datos de manera eficiente utilizando un sistema automatizado para acceder al Registro Médico Electrónico (EMR por sus siglas en inglés "Electronic Medical Record"). Los registros clínicos implican la recuperación de la información, integración, procesamiento y la transmisión mediante el rol de la computación autonómica. El sistema propuesto es un conjunto de elementos autónomos integrados identificados en múltiples niveles que ejecutan simultáneamente varias funciones al tiempo que derivan servicios de varios otros**.**

Quadri et. al. [21], presenta un patrón de arquitectura resultado de la combinación de los siguientes patrones: patrón reactor y patrón de razonamiento basado en casos. El patrón de diseño reactor maneja las solicitudes de servicio que se reciben simultáneamente de más de un cliente, también se encarga de manejar los servicios, cada servicio es una aplicación representada por un controlador de eventos. El patrón de razonamiento es el encargado de la toma de decisiones basado en reglas para determinar un plan de re-configuración correcto. Este patrón de diseño separará la lógica de toma de decisiones de la lógica funcional de la

aplicación. El patrón propuesto es utilizado para realizar reglas de inferencia que ayuden a la toma de decisiones de la re-configuración.

Figura 11. Layered Architecture of personal Health monitoring and analysis system [6].

2.7.2 Análisis de atributos de calidad

Gran parte de los modelos o patrones de arquitectura presentados anteriormente, presentan atributos de calidad, por ejemplo, Drira et al. [1], presenta una serie de patrones de arquitectura con el objetivo de lograr sistemas ampliamente flexibles, escalables e inteligentes. Por ejemplo, el patrón Cognitive Monitoring Management está dedicado a la integración de nuevos dispositivos a la red, generando nuevos datos no necesariamente heterogéneos. Como podemos ver, este patrón reporta el atributo de calidad de escalabilidad al realizar la integración de nuevos dispositivos, analizando más a detalle los patrones propuestos. La *Tabla 14* muestra la descripción de estos atributos de calidad: Escalabilidad, disponibilidad y usabilidad. La *Tabla 16* presenta los atributos de calidad propuestos.

Lasierra et al. [2] propone un modelo ontológico el cual se enfoca a integrar, analizar y evaluar grandes cantidades de datos, siguiendo este enfoque se realizó la búsqueda de atributos de calidad dentro del modelo ontológico con la finalidad de satisfacer los objetivos del modelo, dando como resultado los siguientes atributos de calidad como son: usabilidad, interoperabilidad, escalabilidad y eficiencia. La *Tabla 16* presenta los atributos de calidad propuestos por Lasierra.

Con el desarrollo de los sistemas de información para el cuidado de la salud, donde los datos son accedidos, modificados y compartidos electrónicamente, se desea tener sistemas de bajo costo en instalación y mantenimiento, pero el alto valor de los datos de los pacientes conlleva

a crear sistemas con un alto nivel de seguridad. Chen et al. [3] reporta el uso de los atributos de calidad como son: Seguridad y Modificabilidad mediante el uso de estándares de comunicación de información. Otro trabajo que reporta el uso de seguridad es Sloman et. al [7], [8], el cual atiende los problemas relacionados con la confidencialidad de los datos y el acceso a estos, mediante el uso del patrón de célula auto-gestionada. La *Tabla 16* presenta los atributos de calidad propuestos.

En menor medida, los artículos [4], [5], [6] reportan solo un atributo de calidad de usabilidad. Este atributo de calidad es un indicador sobre la facilidad con la cual el sistema puede ser utilizado por los usuarios. Recordando que el sistema de apoyo al diagnóstico médico no será utilizado por especialistas en el área de computación sino por médicos. Los sistemas implementados deberán tener este atributo de calidad. Por ejemplo, Salih et al. [4], propone un sistema para el cuidado de la salud en la nube, este sistema al estar en la nube podrá ser accedido por cualquier usuario que tenga algún dispositivo con acceso a internet, el sistema ayuda a diagnosticar paciente vía remota.

Tabla 16. Atributos de calidad

Atributo de calidad	Como se implementa
Escalabilidad	• En [1], mediante el patrón Cognitive Monitoring Management, el cual permite la integración de nuevos dispositivos. • En [2], se propone una ontología la cual pretende realizar la integración de datos de diferentes fuentes de información.
Usabilidad	La usabilidad en cualquier sistema de software que se maneje es importante ya que este es un indicador sobre la facilidad con la cual el sistema puede ser utilizado por los usuarios.
Disponibilidad	En [1], mediante el patrón Knowledge, el cual permite evitar cuellos de botella mediante la descentralización del conocimiento.
Seguridad	• En [2], se propone una ontología, la cual ejecuta un análisis del sistema permitiendo saber si ha ocurrido un cambio o no dentro del sistema. • En [3], mediante el modelo de auto-protección el cual anticipa anomalías venideras, envía alertas tempranas al controlador y protege el sistema indicándole al controlador que inicie una respuesta activa a las amenazas seleccionadas. • En [7], [8] mediante el patrón de célula auto-gestionada el cual garantiza el intercambio de información mediante un bus de eventos.
Modificabilidad	En [3], se realiza dentro del modelo de auto protección el cual permite reparar las amenazas detectadas en el caso de que el sistema no sea capaz de resolver el problema por sí mismo.

2.7.3 Análisis de características autonómicas

La CA afronta distintos problemas de los sistemas de apoyo al diagnóstico médico mediante la implementación de características autonómicas, estas pretenden que los sistemas diseñados tengan la mínima intervención humana cuando estén en funcionamiento [15].

En el año 2001 cuando se acuña el término de CA propuesto por IBM se presentan cuatro características autonómicas que son: auto-configuración, auto-optimización, auto-curación y auto-protección [15].

Mediante el análisis de los estudios primarios realizados se recabaron algunas otras características autonómicas. En la *Tabla 17* se enlistan las características autonómicas encontradas.

Tabla 17. Características autonómicas

Auto-Características	Como se implementa
Auto-configuración	• En [1], mediante el patrón Cognitive Monitoring Management, el cual permite la integración de dispositivos, estos se integran y configuran automáticamente. • En [16], se presenta un algoritmo genético llamado platón, para realizar la toma de decisiones, en lugar de ejecutar reglas de inferencia que solo sirven para escenarios concretos.
Auto-Optimización	• En [1], mediante el patrón Predictive Cognitive Management, el cual monitorea, analiza y gestiona los procesos de conocimiento, generando nuevos conocimientos a partir de los ya existentes. • En [1], mediante el patrón Prescriptive Cognitive Management, el cual monitorea y analiza los procesos mientras interactúa con los expertos generando nuevos conocimientos. • En [1], mediante el patrón Knowledge, el cual proponen descomponer el conocimiento centralizado para evitar cuellos de botella • En [2], se presenta un modelo ontológico con la capacidad de gestionar e integrar la información y generar conocimiento mediante la aplicación de reglas de inferencia.
Auto-Búsqueda	En [6], se presenta un elemento autónomo, con la capacidad de realizar búsquedas automáticas en los datos de paciente en función de la especialización del paciente.
Auto-Detección	En [6], se presenta un elemento autónomo, con la capacidad de detectar automáticamente el formato de los registros médicos y combinarlo con los ya existentes.
Auto-Administración	En [4], se presenta el patrón Prescriptive Cognitive Management, el cual es el encargado de administrar los servicios de salud en la nube.
Auto- protección (Seguridad)	En [3], se presenta el modelo Autonomic Security Management, el cual implementa un mecanismo de seguridad para anticipar anomalías, enviar alertas y proteger el sistema. En [7], mediante la aplicación del patrón de célula auto- gestionada el cual garantiza el intercambio de información.

2.7.4 Análisis del uso de estilos de arquitectura

Dentro de los estudios primarios analizados solo se encontró un estudio que presenta estilos de arquitectura. En [8] Lupo et. al. presenta una serie de estilos de arquitectura con el objetivo de construir y diseñar sistemas más grandes a partir de sistemas simples mediante la combinación de estilos como elementos de diseño.

Los estilos de arquitectura mostrados carecen de modelos de diseño ya que solo se presenta una descripción muy breve de como interactúa el estilo de arquitectura con el patrón de célula auto-gestionada SMC. La *Tabla 18* muestra los estilos de arquitectura encontrados.

2.8 Análisis de los modelos y patrones propuestos

La *Tabla 18* describe los modelos encontrados contra los atributos de calidad.

Tabla 18. Análisis de los Modelos y los atributos de calidad

Modelo/Atributo de calidad: Layered Architecture of personal Health monitoring and analysis system. Eficiencia: No. Usabilidad: Sí, sistema encontrado en la nube, de fácil acceso para el usuario final. El sistema puede ser accedido desde cualquier dispositivo con acceso a internet. Seguridad: No. Escalabilidad: No. Interoperabilidad: No. Modificabilidad: No. Disponibilidad: No.	Modelo/Atributo de calidad: HOTMES. Eficiencia: Sí, se basa en el patrón MAPE y en una ontología para la integración y la manipulación correcta de la información. Usabilidad: No. Seguridad: No. Escalabilidad: Sí, dentro de la ontología se maneja la integración de los datos mediante la aplicación del patrón MAPE. Interoperabilidad: Sí, los datos recibidos dentro de la ontología provienen de diferentes dispositivos. Modificabilidad: No. Disponibilidad: No.
Modelo/Atributo de calidad: Célula auto-gestionada (SMC). Eficiencia: Sí, el bus de eventos implementa una etapa de monitorización la cual permite observar si ha ocurrido un cambio. Usabilidad: No. Seguridad: Sí, implementa un bucle de control de realimentación basado en políticas que determinan qué acciones de gestión y re-configuración deben realizarse en respuesta a eventos de interés, como fallas de dispositivos o cambios de contexto. Escalabilidad: No, pero podría implementarse siguiendo el patrón observador. Interoperabilidad: No. Modificabilidad: Sí, gracias al bus de eventos que está dividido en diferentes fases lo cual hace posible modificar una fase sin afectar a las demás. Disponibilidad: No.	Modelo/Atributo de calidad: Healthcare Information system and communications protocols. Eficiencia: No. Usabilidad: No. Seguridad: No. Escalabilidad: No. Interoperabilidad: No. Modificabilidad: Sí, el sistema maneja una arquitectura en capas, la cual permite realizar cambios de manera eficiente. Disponibilidad: No.
Modelo/Atributo de calidad: Auto-Search. Eficiencia: Sí, búsqueda automática de los documentos pertinentes del paciente en función de la especialización del médico. Usabilidad: No. Seguridad: No. Escalabilidad: No. Interoperabilidad: No. Modificabilidad: No. Disponibilidad: No.	Modelo/Atributo de calidad: The publish/ subscribe pattern. Eficiencia: No. Usabilidad: Sí, presenta una estructura semántica de datos bien definida para que, al agregar nuevos dispositivos, el proceso de datos no se afecte. Seguridad: No. Escalabilidad: Sí, ofrece una administración de dispositivos, así como una fácil ampliación del sistema para admitir nuevos dispositivos. Interoperabilidad: Si, administra diferentes dispositivos. Modificabilidad: No.

	Disponibilidad: No.
Modelo/Atributo de calidad: Functional details of the Medical Autonomic Model Eficiencia: No. Usabilidad: Sí, mediante la utilización del patrón MAPE-K. Seguridad: Sí, ejecuta la fase de análisis del modelo MAPE-K. Escalabilidad: No. Interoperabilidad: No. Modificabilidad: Sí, mediante la ejecución la etapa de planeación del modelo MAPE-K. Disponibilidad: No.	Modelo/Atributo de calidad: Observer pattern Eficiencia: Sí, responde al cambio de los caches distribuidos en el momento correcto. Usabilidad: Sí, actualiza automáticamente los cambios de la memoria cache. Seguridad: No. Escalabilidad: No. Interoperabilidad: No. Modificabilidad: No. Disponibilidad: No.
Modelo/Atributo de calidad: Cache pattern. Eficiencia: No. Usabilidad: No. Seguridad: No. Escalabilidad: Sí, es útil para sistemas de gran escala o sistemas distribuidos donde se generan cuellos de botella y limitaciones de rendimiento al acceder a la capa de datos. Interoperabilidad: No. Modificabilidad: No. Disponibilidad: No.	

La *Tabla* 19 describe la comparación de cada uno de los modelos contra las auto-características encontradas.

Tabla 19. Análisis del modelo de las características autonómicas

Model/Self*	Optimización	Configuración	Protección	Administración	Detección
Layered Architecture of personal Health monitoring and analysis system	Si Los sensores y actuadores de la interfaz administran los componentes autónomos que se pueden implementar como servicios web.	No	No	No	No
HOTMES	Si Mediante integración de datos, cuando	Si Utiliza una ontología que es basada en el	No	No	No

	estos provienen de diferentes dispositivos, unificar, acceder, controlar, evaluar y transferir información mediante el diseño de reglas.	patrón MAPE, la cual analiza los datos y aplica reglas de inferencia para la integración de estos.			
Célula auto-gestionada (SMC)	No	Si Si mediante el componente de monitorización el cual detecta, diagnostica y avisa de posibles cambios a los de más componentes.	Si Mediante el uso de un bus de eventos el cual detecta los cambios en los distintos componentes.	No	Si Mediante el componente de monitoreo.
Healthcare Information system and communications protocols	No	No	Si Proporciona un modelo para adoptar la teoría de control para evaluar las vulnerabilidades y para establecer la referencia de la región normal del comportamiento HIS para un entorno operacional específico	No	No
Auto-Search	Si Búsqueda automática de los documentos pertinentes del paciente en función de la especialización del médico.	No	No	No	Si Realiza búsquedas automáticas mediante la especialización del médico. Detecta los cambios si es

					médico o paciente.
The publish/ subscribe pattern	No	Si Se presenta una estructura semántica de datos bien definida para que al agregar nuevos dispositivos.	No	Si Ofrece una administración de dispositivos, así como una fácil ampliación del sistema para admitir nuevos dispositivos.	No
Functional details of the Medical Autonomic Model.	No	No	No	Si El modelo propuesto de computación autonómica basado en el patrón MAPE-K.	No
Observer pattern	Si Se presenta una estructura semántica de datos bien definida para que al agregar nuevos dispositivos el proceso de consumo de datos no se afecte.	No	No	No	No
Cache pattern	Si Es útil para sistemas de gran escala o sistemas distribuidos donde se generan cuellos de botella y limitaciones de rendimiento al acceder a la capa de datos.	No	No	No	No
Patrón mediador del	Si	No	No	No	No

conocimiento semántico	El patrón permite la colaboración de diferentes tipos de proveedores (humanos, máquinas) al poblar la base de conocimiento, basado en el Patrón de conocimiento.				
Knowledge Pattern	Si Propone descomponer el conocimiento centralizado para evitar cuellos de botella.	No	No	No	No
Estadísticas	7	4	2	2	2

La *Tabla* 20 describe la comparación de cada uno de los modelos contra los estilos de arquitectura encontrados.

Tabla 20 Estilos de arquitectura vs Modelos de arquitectura

Estilo de arquitectura/Modelo	Célula auto-gestionada (SMC)
Peer-to-Peer	Modo de interacción ordinario y simétrico entre SMC que intercambian servicios.
Composition	Un SMC encapsula la interfaz de otro y determina su visibilidad a través de la mediación.
Aggregation	El SMC interno se convierte en un recurso externo, pero sin imponer la encapsulación (permite compartir).
Fusion	Combina las interfaces, políticas y objetos gestionados de dos SMC constituyentes en un nuevo SMC
Hierarchical Control	Un SMC de nivel superior controla la ejecución de un conjunto de SMC de hoja.
Cooperative Control	El SMC de una hoja está controlado por un conjunto de SMC de nivel superior de administrador cooperante
Bidding	Ejecución de tareas cooperativas empleando un enfoque de negociación (emisores y oferentes).
Collaborative	Interacción totalmente descentralizada donde las SMC pueden cargar y recibir tareas de sus socios.

Shared-Bus	Proporciona una pizarra para la comunicación desconectada basada en eventos entre SMC.
Correlation	Se pueden combinar eventos individuales generando eventos de nivel superior.
Diffusion	Proporciona una forma de reenviar eventos directamente a SMC que interactúan.
Store-and-Forward	Útil en configuraciones ad hoc donde las SMC pueden no tener una conexión permanente con sus socios que interactúan

2.9 Conclusión del EMS

En los estudios primarios analizados se encontró que la mayoría de los modelos de arquitectura de los sistemas de información para el cuidado de la salud afronta los problemas de análisis, integración y gestión de la información por medio de patrones de arquitectura. Algunos como Lasierra y Urovi proponen un diseño de un modelo ontológico para afrontar este tipo de problemas. Mediante el análisis de los modelos revisados se concluye que la mayoría de las arquitecturas propuestas son deficientes, ya que los modelos propuestos carecen de diseño, además de no considerar un lenguaje estándar para realizar modelos, como lo es UML.

Al realizar la ejecución de la metodología EMS se encontraron trabajos que abordan los siguientes atributos de calidad: Escalabilidad, Seguridad, Usabilidad, Disponibilidad e Interoperabilidad, pero ninguno de los trabajos analizados reporta Mantenibilidad.

La mayoría de los trabajos reporta al menos una de las siguientes auto-características propuestas por IBM que son: auto-configuración, auto-optimización, auto-curación y auto-seguridad, siendo la característica de auto-optimización una de las que más implementan los sistemas.

Resultados de los atributos de calidad y las auto-características.

Tabla 21 Atributos de calidad y publicaciones asociadas

Atributo de calidad	Número de publicaciones	Referencia
Escalabilidad	2	[1],[2]
Seguridad	2	[2],[3]
Interoperabilidad	1	[2]
Usabilidad	5	[2],[4],[5],[1],[6]
Disponibilidad	1	[1]
Modificabilidad	1	[4]

Tabla 22 Auto-características y publicaciones asociadas

Auto-Características	Número de publicaciones	Referencia
Auto-Configuración	3	[1],[18],[19]
Auto-Optimización	5	[1],[2],[3],[4],[5]

Auto-Detección	1	[6]
Auto-Protección	2	[3],[7]
Auto-Administración	1	[4]

Capítulo 3

Antecedentes y Metodología de la investigación

Una metodología de desarrollo de arquitectura de software se ha seguido para desarrollar el meta-modelo de arquitectura propuesto en los objetivos de esta tesis. En otras palabras, se adoptó un marco de trabajo que permita efectuar actividades estrechamente relacionadas al desarrollo arquitectónico. La metodología utilizada, las actividades desarrolladas para el meta-modelo de arquitectura y las herramientas se describen a lo largo de este capítulo.

3.1 Conceptos de arquitectura de software

En el Capítulo 1 se introdujo una definición de arquitectura de software, en ella se hace hincapié sobre la importancia que tienen las estructuras de software. Para estructurar un sistema es necesario implementar diferentes vistas como son: lógica, física, despliegue, de procesos, entre otras. Vista lógica: representa la funcionalidad que el sistema proporcionará a los usuarios finales, es decir, se ha de representar lo que el sistema debe hacer, las funciones y servicios que ofrece. Para completar la documentación de esta vista se pueden incluir los diagramas de clases, de comunicación o de secuencia. Vista física: se muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema, así como las conexiones físicas entre los componentes que conforman la solución (incluyendo los servicios). Para completar la documentación de esta vista se puede incluir el diagrama de despliegue. Vista de despliegue: en esta vista se muestra el sistema desde la perspectiva de un programador y se ocupa de la gestión del software; o, en otras palabras, se mostrará cómo está dividido el sistema de software en componentes y las dependencias que hay entre los componentes. Para completar la documentación de esta vista se pueden incluir los diagramas de componentes y de paquetes. Vista de procesos: se muestran los procesos que hay en el sistema y la forma en la que se comunican; es decir, se representa desde la perspectiva de un integrador del sistema. Para completar la documentación de esta vista se puede incluir el diagrama de actividad [23].

Retomando la definición de arquitectura de software en esta tesis se han construido diferentes vistas, con la finalidad de estructurar el sistema y ayudar al arquitecto a entender cómo se construye el meta-modelo de arquitectura. A partir del diseño el arquitecto podrá generar modelos de arquitectura particulares.

3.2 Ciclo de desarrollo de la arquitectura de software

3.2.1 Requerimientos

La etapa de requerimientos describe en términos generales la funcionalidad, los alcances, los objetivos y las restricciones cuando se construye un sistema de software. La identificación de requerimientos se realiza mediante el análisis del problema a resolver.

Existe desde hace varios años la ingeniería de requerimientos, su principal objetivo es definir lo que se desea producir y su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedad, en forma consistente y compacta el comportamiento del sistema, de tal manera que el comportamiento del sistema minimice los problemas relacionados al sistema. Recolectar los requerimientos consiste en entrevistar al propietario y a todos los involucrados dentro del sistema para generar una lista con base a las necesidades de los involucrados. Debido a que en un sistema se tienen diferentes personas involucradas, todas estas tienen diferentes objetivos primordiales. Por ejemplo, a la persona A, le interesa tener más seguridad dentro del sistema ya que desea realizar transacciones bancarias, pero a la persona B, le interesa que el sistema pueda soportar un gran número de personas. A manera de facilitar la comprensión de los requerimientos, es conveniente que sean clasificados en tipos y niveles de abstracción.

3.2.1.1 Tipos de requerimientos

Requerimientos de negocio: representan los objetivos de alto nivel de la organización o del cliente que requiera al sistema.

Requerimientos de usuario: especifican servicios que por lo habitual dan soporte a procesos de negocio que los usuarios podrán llevar a cabo mediante el sistema.

Requerimientos funcionales: describen detalles finos de diseño y/o implementación relacionados a los requerimientos de usuario.

3.2.1.1 Atributos de calidad

Los atributos de calidad son características no funcionales que se consideran deseables en un sistema de software. También especifican características útiles para establecer criterios sobre la calidad del software. Sin embargo, no todos los sistemas de software deben de tener en cuenta todos los atributos o cualidades, algunos serán más importantes que otros dependiendo del sistema. Actualmente no existe un listado único de atributos de calidad, para la presente tesis utilizamos el estándar para la evaluación de la calidad, **ISO/IEC SQuaRE 9126**.

(International Standard Organization, 2009) [24], el cual es una guía para la identificación y especificación de drivers arquitectónicos. La Figura 13 presenta los atributos de calidad que se consideran en el estándar.

Figura 12. Estándar de atributos de calidad [24].

3.2.1.2 Drivers arquitectónicos

Dentro de los requerimientos que se consideran para el desarrollo de un sistema y que se derivan de los objetivos de negocio, existe un subconjunto que tiene una gran importancia relativa a la arquitectura. Estos requerimientos se conocen en inglés como drivers de la arquitectura. El término "drivers" puede traducirse como "guías", ya que estos requerimientos "guían" el diseño de la arquitectura del sistema. Una estructuración correcta del sistema permitirá satisfacer la mayoría de estos drivers.

Los drivers de la arquitectura incluyen principalmente a los atributos de calidad. Además de esto, incluyen a un subconjunto de los casos de uso que se consideran como primarios. Los casos de uso primarios son aquellos de mayor importancia o de mayor complejidad para el negocio. Las restricciones también son consideradas como drivers arquitectónicos [25]. En el contexto del proceso de arquitectura los drivers se clasifican en tres clases:

Drivers funcionales: es un subconjunto de requerimientos funcionales, los cuales se han definido en la sección de tipos de requerimientos.

Drivers de atributos de calidad: por lo general, todos los atributos de calidad son considerados drivers arquitectónicos, pero un sistema de software no puede contener todos los drivers arquitectónicos, es decir, se tiene que realizar un acotamiento de los drivers ya que implementar todos, repercutirá en tiempo de desarrollo y costo del producto.

Drivers de restricciones: delimitan el proceso de desarrollo del sistema de software, y están divididas en dos clases:

Restricciones técnicas: describen aspectos que restringen el sistema de software.

Restricciones administrativas: describen aspectos que restringen el proceso de desarrollo del sistema.

3.2.1.3 Documento de Visión y Alcance

Un documento de visión define el alcance y el objetivo de alto nivel de un programa, producto o proyecto. Una declaración clara del problema, una propuesta de solución y las características de alto nivel de un producto ayudan a establecer expectativas y reducir riesgos. A continuación, se muestra un esquema del contenido potencial de un documento de visión.

Esquema de documento de visión (Plantilla proporcionada por IBM) [28].

1: Introducción

Esta introducción ofrece una visión general de todo el documento de visión. Incluye el objetivo, el alcance, las definiciones, los acrónimos, las abreviaturas, las referencias y una visión general de todo el documento.

1.1 Objetivo: Presentar el objetivo de este documento de visión.

1.2 Alcance: Describir brevemente el alcance de este documento de visión, incluidos los programas, proyectos, aplicaciones y procesos empresariales asociados con el documento. Incluya cualquier otro elemento que afecte o influya en el documento.

1.3 Visión general: Describa el contenido del documento de visión y explique cómo se organiza el documento.

2: Posicionamiento

2.1 Oportunidad de negocio: Describa brevemente la oportunidad de negocio que aborda este proyecto.

2.2 Declaración de problema: Resuma el problema que este proyecto soluciona. Utilice las declaraciones siguientes como modelo, proporcionando detalles para sustituir los elementos entre paréntesis:

El problema de (describir el problema) afecta a (las partes interesadas a las que afecta el problema). El impacto del problema es (impacto del problema). Una solución eficaz sería incluir (listar algunos beneficios clave de una solución eficaz).

3: Descripciones de la parte interesada y del usuario

Para proporcionar productos y servicios que satisfagan las necesidades de las partes interesadas y usuarios, debe identificar e implicar a todas las partes interesadas como parte del proceso de definición de requisitos. También debe identificar a los usuarios del sistema y asegurarse de que la comunidad de las partes interesadas los representa adecuadamente.

Esta sección ofrece un perfil de las partes interesadas y los usuarios que están implicados en el proyecto. Esta sección también identifica los problemas clave que las partes interesadas y los usuarios consideran que la solución propuesta debe abordar. Esta sección no describe solicitudes ni requisitos específicos; un artefacto de solicitudes de parte interesada independiente se ocupa de estas cuestiones. La descripción del problema clave ofrece los fundamentos y la justificación de los requisitos.

3.1 Necesidades clave de la parte interesada o del usuario: enumere los problemas clave con las soluciones existentes según la percepción de la parte interesada. Aclare estos temas para cada problema:

- ¿Qué causa este problema?
- ¿Cómo se soluciona el problema ahora?
- ¿Qué soluciones desea utilizar la parte interesada?

Debe comprender la importancia relativa que la parte interesada asigna a la solución de cada problema. Las técnicas de clasificación y votación acumulativa ayudan a indicar los problemas que deben resolverse en comparación con las cuestiones que les gustaría abordar a las partes interesadas. Utilice esta tabla para capturar las necesidades de la parte interesada.

Necesidades de la parte interesada

Necesidad
Prioridad
Problemas
Solución actual
Solución propuesta

4: Visión general del producto

Esta sección ofrece una vista de alto nivel de las capacidades del producto, interfaces de otras aplicaciones y configuraciones de sistemas. Esta sección suele constar de tres sub-secciones:

- Perspectiva del producto
- Funciones del producto
- Suposiciones y dependencias

4.1 Perspectiva del producto: coloque el producto en perspectiva en relación con otros productos relacionados con el entorno del usuario. Si el producto es independiente y totalmente auto-contenido, indíquelo aquí. Si el producto es un componente de un sistema más amplio, explique cómo estos sistemas interactúan e identifique las relaciones relevantes entre los sistemas. Un modo de visualizar los componentes principales de los sistemas, interconexiones y relaciones externas más amplios consiste en utilizar un proceso de negocio o un diagrama de caso de uso.

5: Características del producto

Enumere y describa brevemente las características del producto. Las características son las capacidades de nivel superior del sistema que son necesarias para ofrecer beneficios para los usuarios. Cada característica es un servicio solicitado que normalmente requiere una serie de entradas para lograr un resultado satisfactorio. Por ejemplo, una característica del sistema de rastreo de problemas puede ser la capacidad de ofrecer informes de tendencia. A medida que el modelo caso de uso toma forma, actualice la descripción que hará referencia a los casos de uso.

6: Restricciones

Tenga en cuenta cualesquier restricciones de diseño, restricción externa, como requisitos operativos o reglamentarios u otras dependencias.

7: Rangos de calidad

Define los rangos de calidad relativos al rendimiento, la solidez, la tolerancia a fallos, la usabilidad y características similares que la característica establecida no describe.

8: Precedencia y prioridad

Defina la prioridad de las diferentes características del sistema.

9: Otros requisitos del producto

En un nivel elevado, enumere los estándares aplicables, los requisitos de hardware o plataforma, los requisitos de rendimiento y los requisitos de entorno.
9.1 Estándares aplicables: enumere todos los estándares con los que debe cumplir el producto. La lista puede incluir los estándares siguientes:

- Estándares legales y reguladores (FDA, UCC)
- Estándares de comunicación (TCP/IP, ISDN)
- Estándares de cumplimiento de plataforma (Windows, UNIX, etc.)
- Estándares de calidad y seguridad (UL, ISO, CMM)

3.2.1.4 Taller de atributos de calidad (QAW)

El taller de atributos de calidad es un método desarrollado por el SEI [28], que define un proceso para buscar el descubrimiento de atributos de calidad en las fases iniciales de desarrollo, el cual provee una forma para identificar dichos atributos y clarificar los requerimientos del sistema antes de empezar a realizar el diseño de la arquitectura. Los pasos para realizar el QAW se presentan a continuación:

1. Introducción y presentación del QAW a los participantes: se presenta el taller de atributos de calidad de las fases y las actividades a realizar.
2. Presentación de la misión y visión de la organización: el cliente explica el problema, su contexto y los objetivos principales, mientras un miembro del equipo QAW identifica los drivers arquitectónicos.

3. Presentación del plan de arquitectura: el arquitecto presenta un boceto (borrador) de la arquitectura cubriendo los drivers recolectados en el Paso 2.
4. Identificación de los drivers arquitectónicos: con base a la información presentada, el facilitador presenta un conjunto de drivers haciendo énfasis a los que tiene que ver con atributos de calidad, posteriormente se realiza una lluvia de ideas controlada por el equipo del QAW para refinar los drivers.
5. Lluvia de ideas para la generación de escenarios: a partir de los atributos de calidad, cada participante propone un escenario, de modo que el facilitador se encarga de que cada atributo de calidad tenga asociado un escenario.
6. Consolidación de los escenarios: el facilitador pide a los participantes que identifiquen los escenarios que son similares o repetidos, para consolidar y tener un grupo reducido.
7. Priorización de los escenarios: por medio de la votación los participantes del QAW llevan la priorización de los escenarios.
8. Refinamiento de los escenarios: se especifican con mayor detalle los escenarios que son prioritarios.

3.2.1.5 Plantilla SAQAS

La plantilla Identify Self-Adaptive Quality Attribute Scenarios (SAQAS), sirve para documentar los atributos de calidad obtenidos en la etapa de requerimientos, pero particularmente esta plantilla se enfoca en documentar atributos de calidad autonómicos. La plantilla contiene los siguientes componentes [29].

Source: indica el operador que participa dentro del escenario autonómico.
Stimulus: describe el evento el cual causa el problema.
Artifact: indica el elemento que se ve afectado por el estímulo.
Environment: describe el medio ambiente en el cual se ejecuta el estímulo.
Response: indica las acciones que realiza el sistema para resolver el problema.

3.2.2 Diseño de arquitectura

A pesar de que la palabra diseño se menciona en repetidas ocasiones en la ingeniería de software, no se tiene una definición formal para esta. A continuación, mostramos la definición propuesta por Ralph y Wand.

"El diseño es la especificación de un objeto, creado por algún agente, que busca alcanzar ciertos objetivos en un entorno particular, usando un conjunto de componentes básicos, satisfaciendo un conjunto de requerimientos y sujetándose a ciertas restricciones" [26].

De la definición anterior inferimos que el diseño de software busca formar estructuras a partir de un conjunto de requerimientos, que incluyen las restricciones, los requerimientos no funcionales y funcionales.

Una vez que se han analizado y se han especificado los requerimientos, el diseño de software es la segunda de las cuatro etapas que se han propuesto para realizar el meta-modelo de arquitectura. Por lo general el diseño de software sigue un enfoque de divide y vencerás o de modularidad. El objetivo es dividir un problema grande en problemas de menor tamaño, que

puedan ser resueltos más fácilmente. Además, el enfoque de divide y vencerás permite realizar el diseño de forma iterativa e incremental.

El proceso de diseño de la arquitectura está formado principalmente por tres tareas como se muestra en la Figura 13.

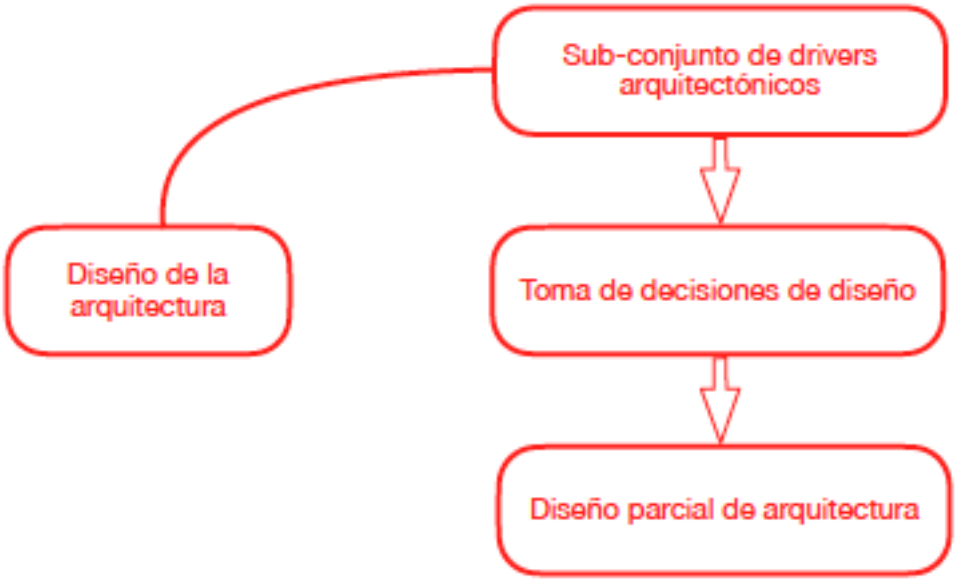

Figura 13. Diseño de la arquitectura.

Seleccionar un subconjunto de drivers arquitectónicos, como se mencionó anteriormente cuando se realiza un sistema de software, se busca un enfoque de divide y vencerás, para realizar el trabajo más fácilmente y realizar el trabajo de una forma iterativa, por lo que se busca principalmente seleccionar un conjunto de drivers que sean de prioridad alta para realizar en la primera iteración.

Toma de decisiones de diseño, dado que los sistemas de software que se construyen son principalmente de uso comercial, los principales problemas que tienen este tipo de software ya han sido atacados con anterioridad. Por ello, se busca identificar, seleccionar y adecuar soluciones existentes que ayuden a ahorrar tiempo y mejorar la calidad. Para esto tenemos los conceptos de diseño, los cuales buscan soluciones probadas a los problemas recurrentes de diseño como son los patrones y las tácticas.

Patrones: son soluciones conceptuales a problemas recurrentes de diseño y proporcionan catálogos de elementos re-usables en el diseño de sistemas de software que formalizan un vocabulario común entre diseñadores estandarizando el modo en que se desarrolla el diseño [27].

Tácticas: las tácticas son conceptos de diseño que influyen sobre el control de la respuesta a un atributo de calidad particular. A diferencia de los patrones, no presentan soluciones

conceptuales detalladas, sino que son técnicas probadas de las ciencias de la computación con las que se resuelven problemas en aspectos particulares relacionados con diversos atributos de calidad [27].

3.2.2.1 Diseño guiado por atributos (ADD)

El método de diseño guiado por atributos (Attribute Driven Design, o ADD), es desarrollado por el Instituto de Ingeniería de Software Carnegie Mellon [35].

El método ADD es un enfoque para definir una arquitectura de software en que el proceso de diseño se basa en los requisitos de atributos de calidad del software. ADD sigue un proceso recursivo que descompone un sistema o elemento del sistema mediante la aplicación de tácticas arquitectónicas y patrones que satisfacen sus requisitos de atributos de calidad de conducción. Los pasos del método se enlistan a continuación:

1. Tener toda la información de los drivers arquitectónicos.
2. Elegir un elemento básico del sistema a descomponer.
3. Identificar los drivers asociados a un elemento.
4. Seleccionar un concepto de diseño (patrones o tácticas) que satisfaga los drivers elegidos.
5. Crear instancias de los elementos seleccionados.
6. Definir interfaces para los elementos seleccionados.
7. Verificar la satisfacción de los drivers seleccionados
8. Repetir el procedimiento para nuevos elementos del sistema.

3.2.2.2 Proceso Unificado Abierto (OpenUP)

OpenUP [30] es un proceso, modelo y extensible, dirigido a la gestión y desarrollo de proyectos de software. OpenUP fue donado a Eclipse Process Framework [31] y desarrollado por un conjunto de empresas de tecnología. OpenUP es un proceso mínimo y suficiente por lo que provee lineamientos para todos los elementos que se manejen en un proyecto de software de tal manera que provee un ciclo de vida estructurado en cuatro fases: concepción, elaboración, construcción y transición.

Cada fase de OpenUP consta de una o más iteraciones en las que se pueden incluir varias actividades, según se efectúe la instancia del proceso. Cada actividad se divide a su vez en varias tareas que pueden requerir ciertos artefactos como entrada, y generan otros artefactos como salida. Algunas actividades pueden incluso tener anidadas a otras actividades. Las siguientes sub-secciones describen brevemente cada una de las fases de OpenUP.

A. Concepción

En esta fase inicia el proyecto de software, las necesidades de cada participante del proyecto son tomadas en cuenta y plasmadas en objetivos, recabando así los requerimientos funcionales, no funcionales y las restricciones del proyecto. Se definen para el proyecto: el ámbito, los límites, el criterio de aceptación, los casos de uso críticos, una estimación inicial del costo y un boceto de la planeación.

Para recabar los requerimientos del sistema se emplean entrevistas con el cliente, con el objetivo de identificar las necesidades del negocio, el arquitecto del equipo de desarrollo realiza anotaciones de los requerimientos del sistema y los transforma en casos de uso, ya que es una manera muy intuitiva de recolectar requerimientos, además de ser un elemento clave que guía todo el proceso de desarrollo, pues expresan la funcionalidad del sistema.

El hito de esta fase se alcanza cuando las primeras versiones de los documentos conteniendo todo lo anterior han sido generadas, y por lo tanto los objetivos para el ciclo de desarrollo en curso han sido estipulados y comprendidos.

B. Elaboración

En la fase de elaboración se identifican, detallan y comprenden todos los requerimientos del sistema; con el objetivo de priorizar los casos de uso para realizar la primera iteración del sistema. Se deben realizar tareas de análisis del dominio y definición de la arquitectura para elaborar un plan de proyecto, estableciendo los requisitos principales de la arquitectura. Al final de la fase se debe tener una definición clara y precisa de los casos de uso, actores, arquitectura del sistema de software y un prototipo ejecutable.

C. Construcción

En esta fase se implementan los componentes y funcionalidades del sistema de software. Los resultados obtenidos en forma de incrementos deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado. Hasta este punto se han realizado pruebas a todos los componentes desarrollados, reduciendo el índice de riesgos.

Figura 14. OPENUP.

D. Transición

El objetivo de la transición es poner en entorno de operación la versión final del producto de software. La transición consta de las siguientes sub-fases: pruebas beta, pilotaje y capacitación de los usuarios finales y de los encargados del mantenimiento del sistema. En función a la respuesta obtenida por los usuarios puede ser necesario realizar cambios en las entregas finales o implementar alguna funcionalidad más solicitada, en caso de que esto sucediera se tendría que implementar un nuevo ciclo de desarrollo. Para evitar pérdidas económicas se deberá de considerar desde un principio que el usuario final solicite correcciones debido a problemas de funcionalidad.

3.2.3 Documentación de la arquitectura

En el Capítulo 2 se habló del proceso que se realiza para la construcción de un sistema de software, como se muestra en la Figura 12 del Capítulo 2. Como primer paso se tiene el levantamiento de requerimientos, posteriormente se define el diseño de la arquitectura a partir de los requerimientos, para posteriormente pasar a una etapa de documentación.

La palabra "documentar" en ingeniería de software se asocia al proceso de elaboración de documentos, enfocados a transmitir información relevante del sistema. Además de comunicar la información, documentar un sistema, permite persistir la información para evitar posibles riesgos en el futuro, por ejemplo: si un miembro del equipo de desarrollo abandona el proyecto, la información no se perdería ya que sus productos de trabajo quedarían almacenados y disponibles para continuar con el proceso de construcción del sistema.

Algunas de las razones más relevantes para realizar la documentación de la arquitectura son:

1. Transmitir la información sin ambigüedades.
2. Preservar la información sobre la arquitectura, sin riesgo alguno.
3. Proveer un lenguaje estándar para la comunicación de la información.
4. Guiar a los desarrolladores en la construcción de sistemas de software.

La documentación de la arquitectura nos permite describir estructuras con el propósito de transmitir la información eficientemente a los diferentes interesados en el sistema, pero documentar todos los elementos que contiene un sistema es sumamente complicado si no se tiene una guía, debido a que en un sistema se tienen diferentes interesados y no todos los elementos, estructuras y perspectivas son relevantes para cada uno de ellos. Por lo que uno de los conceptos más utilizados para realizar la documentación es la "vista".

Una vista describe una o más estructuras de la arquitectura en términos de los elementos que la conforman.

En el contexto del ciclo de vida para el desarrollo de software se tienen diferentes vistas como son: vistas lógicas, vista de procesos, vista física y de escenarios**.**

La arquitectura de software se trata de abstracciones, de descomposición y composición, de estilos y estética. También tiene relación con el diseño y la implementación de la estructura de alto nivel del software. Los diseñadores construyen la arquitectura usando varios elementos arquitectónicos elegidos apropiadamente. Estos elementos satisfacen la mayor parte de los requisitos de funcionalidad y desempeño del sistema. Pero como vimos en la sección anterior todas estas estructuras y elementos de la arquitectura no sirven, si no se tiene una buena comunicación de la información para el desarrollo del sistema. Para esto existen métodos de documentación que ayudan a transmitir la información de una manera clara y formal.

La propuesta de 4+1 vistas, es un método para la documentación de arquitectura propuesto por Philippe Kruchten [32], este método ayuda a documentar la información a partir de la generación de diferentes vistas como son:

1. Vista lógica: esta vista representa la funcionalidad que el sistema proporcionará a los usuarios finales. Es decir, se ha de representar lo que el sistema debe hacer, y las funciones y servicios que ofrece.

2. Vista de procesos: esta vista muestra los procesos que hay en el sistema y la forma en la que se comunican estos procesos; es decir, se representa desde la perspectiva de un integrador de sistemas, el flujo de trabajo paso a paso de negocio y operacionales de los componentes que conforman el sistema

3. Vista de desarrollo: esta vista muestra el sistema desde la perspectiva de un desarrollador y se ocupa de la gestión del software; o, en otras palabras, se mostrar como está dividido el sistema en componentes y las dependencias que hay entre esos componentes.

4. Vista física: esta vista se muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema, así como las conexiones físicas entre esos componentes que conforman la solución (incluyendo los servicios).

5. Vista "+1": esta vista es representada por los casos de uso y tiene la funcionalidad de unir y relacionar las otras 4 vistas, esto quiere decir que desde un caso de uso podemos ver como se ligan las otras 4 vistas, con lo que tendremos una trazabilidad de componentes, clases, equipos, paquetes, etc., para realizar cada caso de uso.

3.2.4 Evaluación de la arquitectura

Cuando un producto que se encuentra en el mercado falla, el impacto dentro de la empresa suele ser grande a nivel monetario, detectar riesgos o errores antes de que el producto sea puesto en el mercado ayuda a evitar este tipo de problemas. Hablando de ingeniería de software, la evaluación es una técnica con la que cuentan los desarrolladores y arquitectos para evitar que las fallas y riesgos lleguen a los usuarios finales, evitando así corregir los errores en momentos que es complicado y costoso.

La verificación y validación son dos conceptos considerados en la ingeniería de software para la detección de riesgos, defectos y errores. En un nivel más detallado y para el contexto de las arquitecturas de software, las inspecciones se especializan en evaluaciones de arquitectura.

Validación: Revisión realizada sobre entregables intermedios o finales de un proyecto para asegurar que satisfacen las necesidades de los usuarios finales [27].

Verificación: Revisión realizada sobre entregables intermedios o finales de un proyecto para asegurar que cumplen con los requerimientos definidos del proyecto [27].

El diseño de la arquitectura es la base para la construcción de un sistema de software, por lo que es importante que la arquitectura se centre en los objetivos, requerimientos y requisitos planteados por el arquitecto y el cliente, desviarse de estos atributos de calidad repercutirá en tiempo de desarrollo, costo y no entregarlo en la fecha acordada.

Realizar evaluaciones de arquitectura consiste en revisar e inspeccionar las vistas arquitectónicas buscando errores o riesgos que no satisfagan los objetivos, requerimientos y restricciones acordados con el cliente.

3.2.4.1 Método de evaluación de equilibrios de arquitectura (ATAM)

Uno de los procesos de evaluación para arquitecturas de software más conocidos es el método de análisis de equilibrios de la arquitectura, o ATAM (por sus siglas en inglés). Desarrollado por los investigadores del SEI, tiene como objetivo "evaluar las consecuencias de las decisiones arquitectónicas respecto de los requerimientos de drivers del sistema" (Kazman, Klein y Clements, 2000).

Su nombre se debe a que no solo se evalúa que una arquitectura de software cumpla con la calidad exigida, sino que analiza las interacciones de los distintos atributos de esta y cómo ellos se equilibran o restringen unos a otros. Puede realizarse más de una vez durante el diseño de la arquitectura, o bien, una vez concluido, y aplicarse además a productos de software terminados o en proceso. Es definido por Kazman, Klein y Clements (2000), y se explica con mayor amplitud en el libro de Clements, Kazman y Klein (Evaluating Software Architectures, 2002).

El ATAM toma como entradas la documentación de la arquitectura y cualquier otro artefacto que haya sido producido por el diseño de esta. El resultado de hacer una evaluación usando el método es una lista de riesgos en ella, puntos de sensibilidad o que requieren atención, y puntos de equilibrios entre atributos de calidad. Los roles principales en una evaluación con el ATAM son el arquitecto de software, los interesados en el sistema con la arquitectura y el personal que apoya la realización de la evaluación con este método, conocido como equipo ATAM.

3.3 Metodología para el diseño del meta-modelo de arquitectura con características autonómica de apoyo al diagnóstico médico

La metodología es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de la arquitectura.

El diseño de meta-modelo de arquitectura fue construido basado en la metodología de ciclo de vida de diseño de la arquitectura, en la cual hemos contemplado cuatro grandes fases:

1) Levantamiento de requerimientos de la arquitectura: en esta fase se contemplaron los siguientes puntos de interés:

- Análisis del estado del arte: el principal objetivo de este análisis es encontrar los trabajos relacionados con arquitecturas autonómicas de apoyo al diagnóstico médico y en este caso en particular brindó las principales deficiencias de los sistemas HCIS.

- Documento de Visión y Alcance: el objetivo de este documento fue establecer los alcances del diseño de la arquitectura y determinar las principales limitaciones.

- Taller de atributos de calidad: en base a que ya se tenían las principales necesidades de los sistemas HCIS, se enfocó el taller en QAW en las principales necesidades y se seleccionaron algunos de los escenarios principales a ser considerados para el diseño del meta-modelo.

2) Diseño del meta-modelo de arquitectura: en esta fase se contemplaron los siguientes puntos:

- Diseño del primer meta-componente a realizar: en base al levantamiento de requerimientos se tomó la decisión para realizar el diseño del primer meta-componente.

- Vistas arquitectónicas: se realizó la construcción de meta-componentes en base a un conjunto de vistas.

- Patrones de arquitectura: se seleccionó y se implantaron patrones de arquitectura para solucionar problemas recurrentes de diseño.

3) Documentación de la arquitectura: en esta fase se contemplan los siguientes puntos de interés:

- Documentación del diseño del meta-modelo de arquitectura: permite realizar la persistencia del meta-modelo de arquitectura, además de un fácil acceso a la información.

- Documentación del proceso de instanciación: permite realizar la persistencia de la información, además de brindar un fácil acceso a la información y establecer un lenguaje estándar de comunicación.

4) Evaluación del meta-modelo de arquitectura: en esta fase se contemplan los siguientes puntos de interés:

- Implementar un meta-modelo de arquitectura con las mínimas fallas posibles dentro de la arquitectura.

- Implementar un meta-modelo de arquitectura de mejor calidad.

Las cuatro fases antes mencionadas son fases requeridas para la construcción de cualquier modelo de arquitectura, principalmente las hemos seleccionado pensando en desarrollar un meta-modelo de arquitectura. Siendo estas cuatro fases nuestra metodología, la Figura 15, muestra el orden de cada una de estas fases en el cual se tiene que ejecutar.

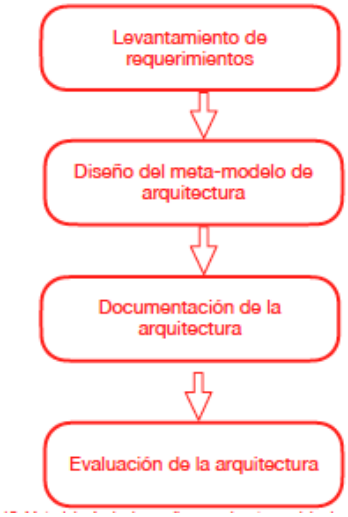

Figura 15. Metodología de desarrollo para el meta-modelo de arquitectura.

3.4 Metodología utilizada para realizar el meta-modelo de arquitectura ARTLESS

Para realizar el diseño (Fase 2) del meta-modelo de arquitectura se ha seguido una versión de ADD que se verá con mayor profundidad en el Capítulo 4. La Figura 16 representa una versión de la metodología seguida para crear el meta-modelo de arquitectura, como se observa primero se ha pasado una fase de levantamiento de requerimientos, que posteriormente se convierten en escenarios autonómicos, estos alimentan el diseño de la arquitectura y como resultado final se tiene el diseño de algunas vistas arquitectónicas más representativas para el modelo propuesto. También de forma paralela a los escenarios se tiene la documentación de la arquitectura.

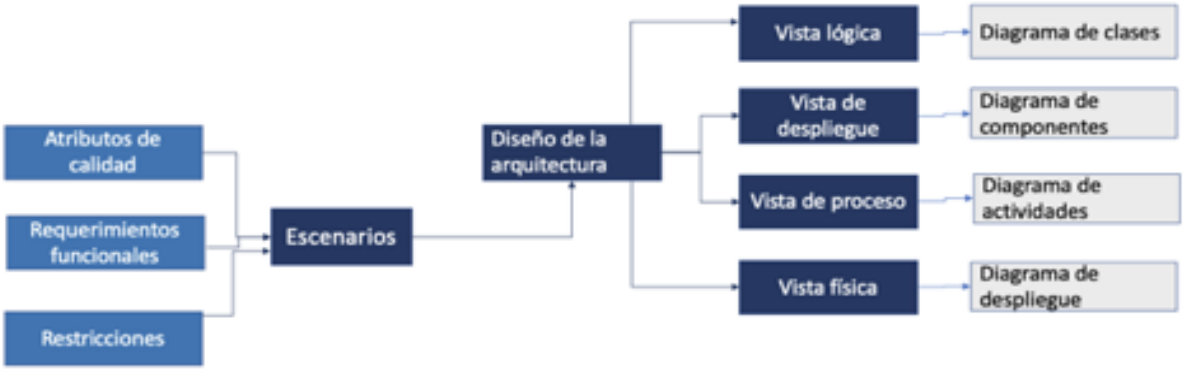

Figura 16. Metodología para el diseño del Meta-modelo de arquitectura.

Capítulo 4

Meta-modelo de arquitectura autonómica para sistemas HCIS (ARTLESS)

En esta tesis se desarrolla un meta-modelo de arquitectura llamado: **Autonomic meta-aRchitecTure modeL for HCIS mEdical diagnoSiS** (ARTLESS), para el diseño de ARTLESS se ha seguido la metodología mostrada en la Sección 3.3. Se han establecido los principales drivers arquitectónicos como son: requerimientos funcionales, no funcionales y restricciones. Se ha realizado el levantamiento de requerimientos, en este caso no se tiene un cliente para realizar el levantamiento de requerimientos, debido a que es un modelo genérico. Por lo tanto, se ha realizado el estado del arte del Capítulo 2, basado en los objetivos planteados para esta tesis, dando como resultado el levantamiento de requerimientos en base al estado del arte.

El concepto de meta-modelo de arquitectura se define como un lenguaje utilizado para describir modelos de arquitectura, en este contexto, definimos un meta-modelo como la herramienta para diseñar un modelo de arquitectura para un sistema de software en particular. Para entender mejor qué es un meta-modelo de arquitectura se han definido los niveles de abstracción.

4.1 Concepto de meta-arquitectura

En el contexto de esta tesis se definen los niveles de abstracción que están jerarquizados y que se usarán para definir el meta-modelo de arquitectura autonómica para sistemas HCIS, se presentan a continuación.

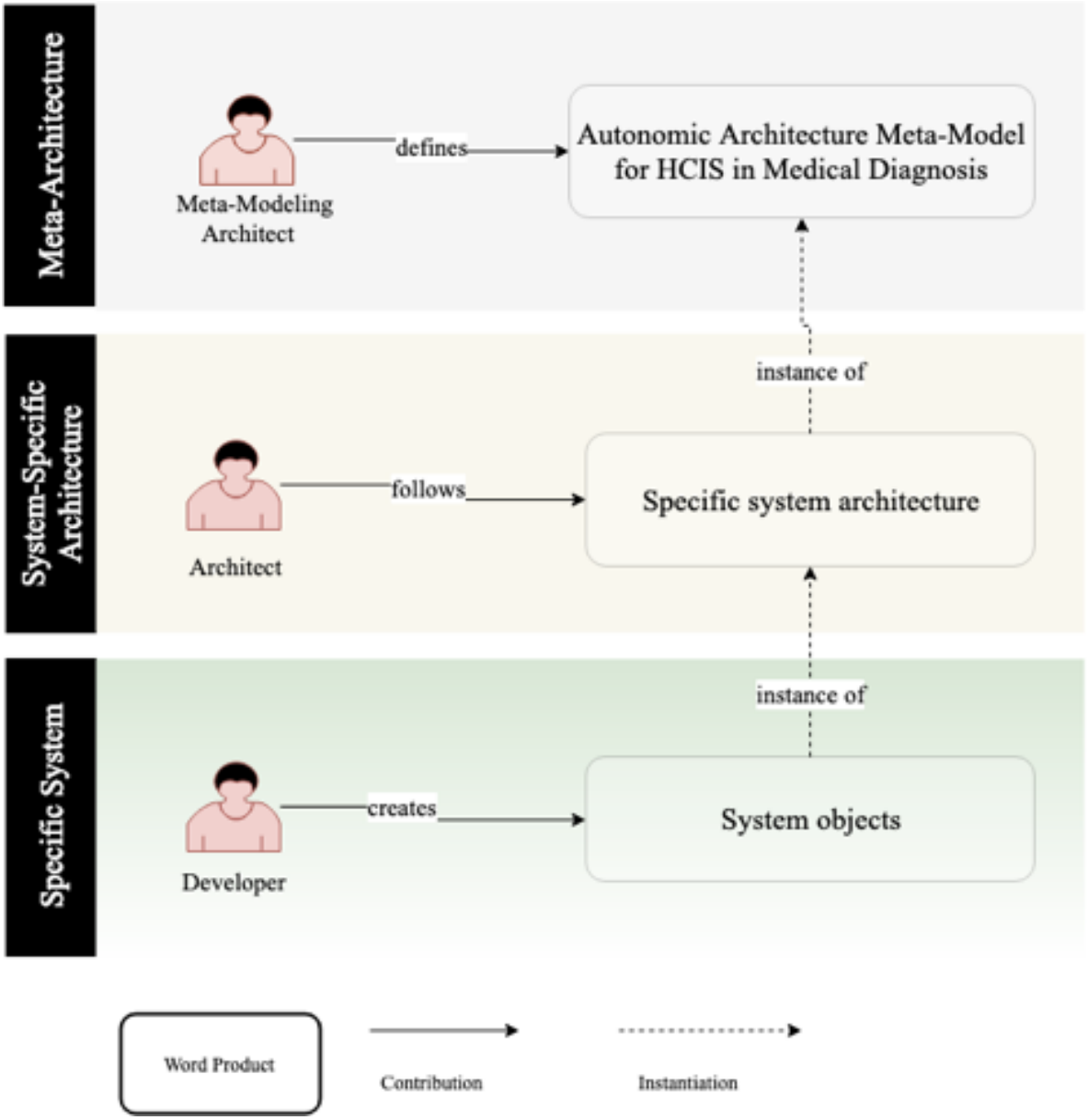

Figura 17. Niveles de abstracción.

Niveles de abstracción de la meta-arquitectura para HCIS

1. **Nivel Meta-Architecture** (**Meta-Arquitectura,** Figura 17). En este nivel se describe la meta-arquitectura para sistemas HCIS, la cual recoge características comunes de sistemas de apoyo al diagnóstico médico. Específicamente sistemas que identifican y clasifican diagnósticos de pacientes con base a requisitos específicos del equipo médico con el objetivo de ingresar a los pacientes identificados a proyectos de experimentación médica. Las características comunes que se abstraen en este nivel son los requerimientos funcionales, no-funcionales y restricciones propias del diagnóstico médico. En este nivel también se incorporan características autonómicas (Sección 3.2), con base en los resultados del estado del arte y que son aplicables y necesarios en los sistemas de cuidado

de la salud. El objetivo de una meta-arquitectura es; apoyar al arquitecto de sistemas para que diseñe arquitecturas específicas autonómicas con requisitos funcionales, no-funcionales y restricciones específicas para el diagnóstico médico. En este sentido, la meta-arquitectura funciona en un nivel de abstracción superior al de un diseño de una arquitectura de un sistema (Nivel 2, **System-Specific Architecture**). Por lo tanto, varias arquitecturas se podrán generar con elementos comunes a todas ellas, a partir de la meta-arquitectura.

2. **Nivel System-Specific Architecture (Sistema Específico de Arquitectura**, Figura 17**),** en este nivel se diseña la arquitectura del sistema objetivo, es decir, dicho diseño se obtiene a partir de instanciar la meta-arquitectura por medio de un proceso de instanciación. En el proceso se deben considerar las entradas propias del sistema específico que se quiere implementar a partir de la arquitectura específica. Por lo tanto, este diseño tendrá los requisitos funcionales y no-funcionales que son comunes al meta-modelo, más los requisitos específicos al sistema objetivo. La implementación de este diseño de arquitectura dará como resultado el sistema requerido por el usuario (Nivel 3, **Specific System**).

3. **Nivel Specific System (Sistema Específico**, Figura 17**)** se encuentra el sistema de software objetivo. Este sistema es implementado por el equipo de desarrollo, a partir del diseño de arquitectura específica que se ha creado en el nivel **System-Specific Architecture**. Este sistema tendrá elementos comunes de los sistemas autonómicos de apoyo al diagnóstico médico, pero tendrá aspectos muy específicos del proyecto de desarrollo en cuestión y que han sido considerados desde su arquitectura en el nivel **System-Specific Architecture**.

4.2 Decisiones de diseño de ARTLESS

En la Sección 3.2 se ha introducido la metodología para desarrollar arquitecturas de software. Como primer paso se tiene el levantamiento de requerimientos de arquitectura, el cual está enfocado en encontrar el nicho de oportunidad y describir la problemática del sistema de software. Como punto inicial para el diseño de ARTLESS se realizó un documento de Visión y Alcance, en el cual se establecen objetivos, requerimientos funcionales y no funcionales que, en principio, servirán como guía para estipular las principales self-features (auto-características) que deberán considerarse para la propuesta inicial de la meta-arquitectura.

El documento de Visión y Alcance establece un conjunto de objetivos base para el diseño de los meta-componentes. Los objetivos fueron planteados en base a 2 decisiones fundamentales para realizar el diseño de los meta-componentes: 1) estado del arte (Capítulo 2) realizado para arquitecturas autonómicas de software de apoyo al diagnóstico médico, el cual mostró que los sistemas HCIS actuales son deficientes en términos de gestión de información, seguridad, almacenamiento e infraestructura que ayude a modificar, instalar, mantener y diseñar nuevos componentes. 2) con base al objetivo general de la tesis, el cual nos pide diseñar un meta-modelo de arquitectura y al objetivo particular dos (Sección 1.2), el cual nos enfoca a realizar el diseño de la meta-arquitectura para una especialidad médica.

A continuación, se muestran los principales objetivos plasmados en el documento de Visión y Alcance:

Obj-1. Apoyar el diseño de arquitectura de sistemas HCIS de apoyo al diagnóstico médico mediante el desarrollo de un meta-modelo de arquitectura que proporcione al arquitecto una base para el diseño de sistemas HCIS que incorpora características para apoyar el diagnóstico médico de pacientes.

Obj-2. Proporcionar al arquitecto un mecanismo que guíe la construcción de sistemas HCIS mediante el desarrollo de un proceso que proporcione los mecanismos para instanciar un meta-modelo de arquitectura.

Obj-3. Mejorar el mantenimiento de los sistemas HCIS de apoyo al diagnóstico médico con respecto a la configuración de bases de datos dedicadas a almacenar el registro médico electrónico con la implementación de CA.

Obj-4. Mejorar el mantenimiento de los sistemas HCIS de apoyo al diagnóstico médico con respecto a la detección y reparación de problemas en bases de datos dedicadas a almacenar el registro médico electrónico y a la detección y reparación de fallas en los sistemas HCIS de apoyo al diagnóstico médico.

Obj-5. Mejorar la seguridad de los sistemas HCIS mediante el desarrollo de una self-feature y la implementación de patrones que cubren los componentes de seguridad relacionados con: confidencialidad, autenticación e integridad del ISO/IEC SQuaRE 25010. [24].

Basándonos en los objetivos anteriores a continuación se presentan las 4 self-features correspondientes al paradigma de CA y que se implementarán en el diseño de ARTLESS:

1) Meta-Self-MedicalDiagnosisAnalysis. Está self-feature está enfocada a cubrir el Obj-1 del documento de Visión y Alcance. Esto es una contribución original de esta tesis, ya que como se demostró en el estado del arte (Capítulo 2) no existe una self-feature dedicada a filtrar el diagnóstico médico de pacientes.

2) Meta-Self-MedicalConfiguration. Está enfocado a cubrir el Obj-3 planteado en el documento de Visión y Alcance [28]. Este Meta-S-MC está basado en el self-configuration de IBM para realizar la auto-configuración en sistemas de software. La contribución de esta tesis es un Meta-Self-MedicalConfiguration, para resolver los problemas de configurar los sistemas HCIS de apoyo al diagnóstico médico y a la configuración de bases de datos que almacenan los registros médicos de pacientes.

3) Meta-Self-MedicalHealing. Está enfocado a cubrir el Obj-4 planteado en el documento de Visión y Alcance [28]. Este Meta-S-MH es una propuesta lanzada por IBM para realizar la curación o reparación de sistemas de software de manera autónoma. Pero en esta tesis enfocaremos el componente a reparar a los sistemas HCIS de apoyo al diagnóstico médico y a las bases de datos que contienen el registro médico de pacientes [1].

4) Meta-Self-MedicalSecurity. Está enfocado a cubrir el Obj-5 planteado en el documento de Visión y Alcance. Este Meta-S-MS es una propuesta lanzada por IBM [28] para integrar seguridad en sistemas de software de forma autónoma. En esta tesis enfocaremos este componente a la seguridad de los sistemas HCIS de apoyo al diagnóstico médico en los campos de autenticación, confidencialidad e integridad del ISO/IEC SQuaRE 25010**.**

El objetivo dos se cubrirá mediante el desarrollo de un proceso de instanciación que nos permite crear arquitecturas particulares a partir del ARTLESS.

ARTLESS está conformado por las 4 self-feature antes mencionadas y para realizar el diseño del meta-modelo de arquitectura se utilizará una versión de ADD que es un método utilizado para guiar el diseño de arquitecturas utilizando atributos de calidad. Para nuestro caso en particular se tomarán los primeros cuatro puntos de ADD (Sección 3.2.2.1), debido a que ADD está enfocado principalmente en desarrollar arquitecturas particulares y no meta-modelos, por lo tanto solo se utilizaran los puntos de interés para el diseño de ARTLESS. También se utilizará la plantilla SAQAS (Sección 3.2.1.5) que nos permite documentar escenarios de atributos de calidad autonómico. Recordando que para el diseño de ARTLESS se realizará bajo el paradigma de CA es necesario tener un método que nos ayude a documentar los escenarios autonómicos que serán implementados dentro del meta-modelo, es por esta razón que se ha seleccionado la plantilla SAQAS.

4.3 Diseño de ARTLESS

Siguiendo los puntos de interés del método ADD y utilizando la identificación de escenarios (SAQAS) [29] para documentar los escenarios de las 4 self-features, se realizarán cuatro iteraciones de desarrollo de ADD y SAQAS para el diseño del meta-modelo de arquitectura. Cada iteración se corresponde con la implementación de las 4 self-features autonómicas explicadas en la Sección 4.2 que se van a implementar en el diseño de ARTLESS.

A continuación, se presentan los cuatro meta-componentes de diseño que corresponden a la implementación de las 4 self-features especificadas en la Sección 4.2 para el diseño de ARTLESS. Estos componentes se ubican en el Nivel Meta-Architecture (ver Figura 18), por lo tanto, en el resto del documento nos referiremos a ellos como meta-componentes.

- Meta-Self-MedicalDiagnosisAnalysis: Capacidad que tiene un sistema para la identificación de nuevos pacientes de manera autónoma y de realizar la filtración de pacientes con respecto a su diagnóstico médico.
- Meta-Self-MedicalConfiguration: Es la configuración dinámica en respuesta a cambios en el ambiente.
- Meta-Self-MedicalHealing: La capacidad que tiene el sistema para detectar, diagnosticar y reparar automáticamente el mal funcionamiento de alguna de sus partes.
- Meta-Self-Medicalsecurity: Protección ante posibles ataques internos como externos.

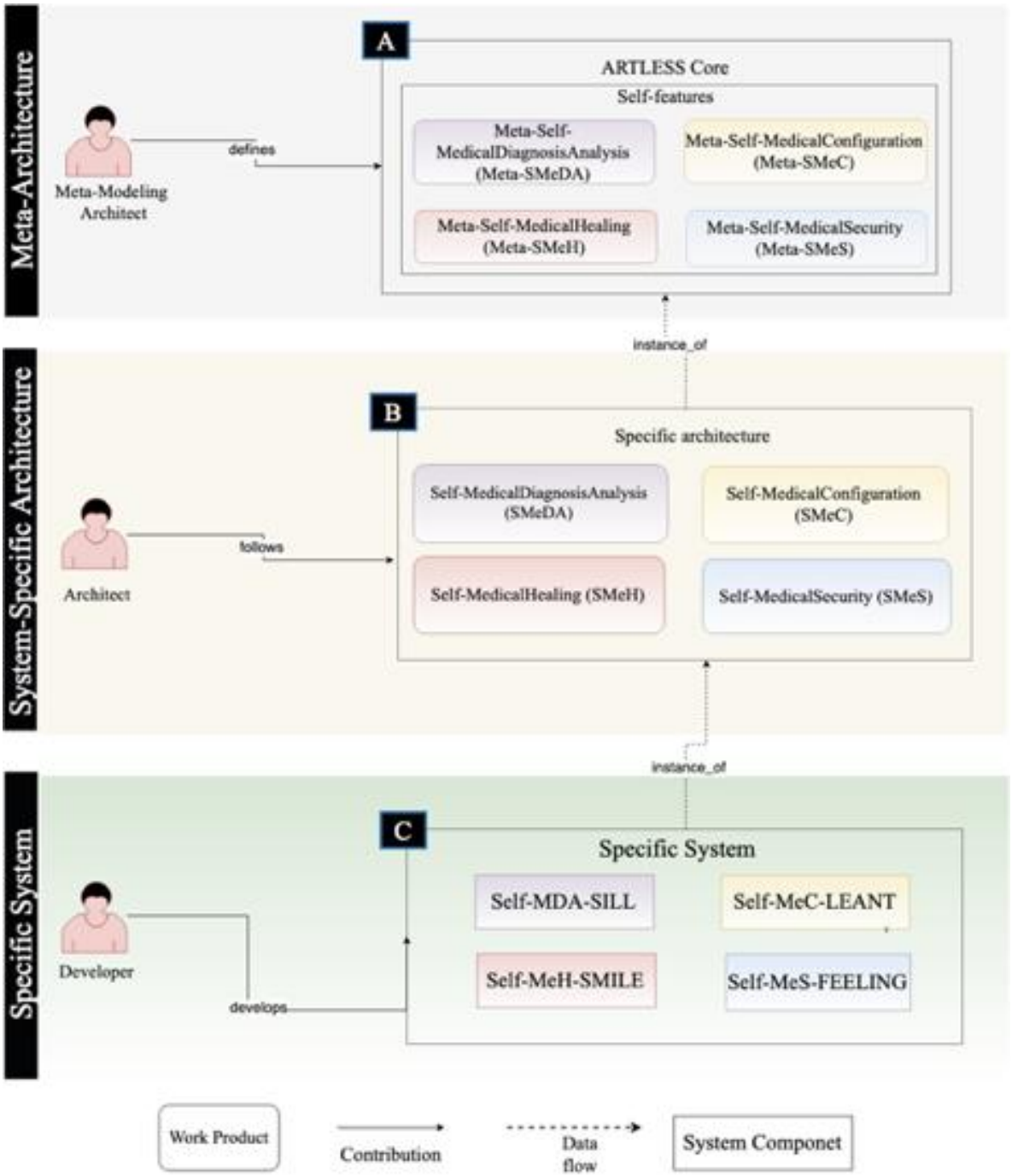

Figura 18. Niveles de abstracción ARTLESS

Los cuatro meta-componentes autonómicos construyen la meta-arquitectura bajo el paradigma de CA [0].

El diseño de la meta-arquitectura implementa de forma concreta las siguientes vistas: vista física (diagrama de despliegue), vista lógica (diagrama de clases, diagrama de secuencia), vista de procesos (diagrama de actividades), vista de despliegue (diagrama de componentes) y vista de escenarios (diagrama de casos de uso).

Para ejemplificar el **Nivel Meta-Architecture** en el que se encuentra ARTLESS es equivalente al nivel de abstracción en que se encuentra la arquitectura de referencia Meta-Object Facility (MOF) [33], la cual es la meta-arquitectura que define el lenguaje de modelado orientado a objetos (UML) [23]. En la actualidad UML se utiliza en la construcción de sistemas de software como lenguaje de modelado.

A continuación, y de acuerdo a los niveles de meta-modelado que se explicaron en la Sección 4.1, se ejemplifica como se utilizara ARTLESS en un proyecto en la industria por medio de la Figura 18. Sin embargo, la especificación de cómo crear estas instancias de forma muy detallada se encuentra en el Capítulo 5.

En el **Nivel System-Specific Architecture** (Figura 18) el arquitecto se encarga de definir el diseño de la arquitectura específica de un sistema de software HCIS de apoyo al diagnóstico médico, a partir de la meta-arquitectura ARTLESS por medio de un proceso de instanciación. Esto potencia el diseño de una arquitectura de un sistema de software de una forma repetible, identificando prontamente errores introducidos por el ingeniero de software en cada actividad del proceso, optimizando el tiempo en el que se realiza la instancia de la arquitectura.

La Figura 18 muestra que el arquitecto del sistema ha realizado una instancia a partir de ARTLESS y como resultado ha obtenido el diseño de una arquitectura específica.

En el **Nivel Specific System** (Figura 18), se ubica el sistema de software que se implementa a partir del diseño de la arquitectura específica obtenida de la instanciación de ARTLESS. El sistema que se ha implementado contiene un conjunto de requerimientos que vienen desde la meta-arquitectura, incluyendo los requerimientos capturados en la etapa del segundo nivel.

El sistema de software es desarrollado y probado por el equipo de desarrollo siguiendo el diseño de la arquitectura. No seguir este diseño repercutirá en tiempo, sistemas deficientes y costo elevado.

El sistema se implementa con el objetivo de satisfacer las necesidades del cliente. En las siguientes secciones se presentará el diseño formal de cada uno de los cuatro meta-componentes desarrollados en esta tesis.

4.3.1 Diseño del Meta-Self-MedicalDiagnosisAnalysis

Este componente se ha diseñado mediante el desarrollo de una serie de vistas, las cuales en términos generales abstraen conceptos comunes de varias arquitecturas y las elevan a un nivel de abstracción superior.

En la Sección 4.3 se ha presentado una definición concreta del componente Meta-S-MDA, ahora se detallará dicha definición. Como primer objetivo del Meta-S-MDA, será la **identificación de nuevos pacientes** potenciales a ser filtrados en base a su diagnóstico médico y a una especialidad médica (por ejemplo: traumatología). Como segundo objetivo, será **filtrar a los pacientes** en base a su diagnóstico médico previamente establecido. Es decir,

los nuevos pacientes que se han seleccionado tienen ciertas características que pertenecen a una especialidad médica, por ejemplo, Traumatología, y en base a esta primera característica se realiza la selección de nuevos pacientes. Posteriormente el diagnóstico médico que se le ha asignado contendrá características específicas del diagnóstico de un paciente. Por ejemplo, si es trauma de pie, tobillo, pierna, brazo, cabeza, etc. Por lo tanto, en base a estas características que son más específicas se realiza la filtración del paciente en base a su diagnóstico médico, asignándole un médico especialista.

Características médicas para apoyar el filtrado de pacientes:

A. Identificar nuevos pacientes:
A. Pacientes que no tienen un registro médico.
B. Pacientes que han estado inactivos por un periodo mayor a 8 semanas.

B. Filtrar pacientes respecto a su diagnóstico médico:
A. Identificar el área médica asignada por el médico tratante, por ejemplo: Angiología, Dermatología, Ginecología, Oftalmología, Urología, Traumatología, etc.
B. Identificar la patología asignada por el médico tratante, por ejemplo: Obesidad, Diabetes, Cáncer, Asma, Gripe, etc.
C. Identificar características específicas del diagnóstico, por ejemplo: Fracturas, Luxaciones, Esguinces, traumatismos medulares, etc.

A continuación, se muestra la selección de atributos de calidad enfocados a cubrir el objetivo planteado en el documento de Visión y Alcance y en la definición formal del Meta-S-MDA.

4.3.1.1 Atributos de calidad del Meta-Self-MDA

La selección de los atributos de calidad para Meta-S-MDA se realizó tomando tres objetivos: 1) el análisis del estado del arte, 2) definición del meta-componente, 3) definición del atributo de calidad del ISO/IEC SQuaRE 25010. La *Tabla 23* muestra la relación de los atributos de calidad y el meta-componente.

Tabla 23. Meta-S-MDA a atributos de calidad

Meta-Component e	**Atributo de calidad**	**Sub-categoría (s)**	**Justificación**
Meta-Self-MedicalDiagnosisAnalysis	Usabilidad	Efectividad	Se ha seleccionado el atributo de calidad de efectividad ya que el meta Meta-Self-MDA está enfocado a realizar una tarea que es: filtrar pacientes respecto a su diagnóstico médico. Siendo así que este Meta-componente está dedicado en realizar una tarea en específico.
	Disponibilidad	Confiabilidad	Funcionalidad El Meta-Self-MDA es el componente principal para identificar nuevos pacientes y filtrarlos respecto a su diagnóstico médico. Este componente tiene que estar disponible en todo momento, ya que el

			registro de los pacientes en un hospital se realiza las 24 horas del día. Por otra, parte el Meta-Self-MDA debe ser confiable y evitar posibles fallas, ya que, si llegase a fallar, no se filtraría ningún paciente y habría una lista de pacientes en espera de ser filtrados. Este problema podría repercutir a nivel de los pacientes en pérdidas de tiempo y dinero.
Compatibilidad	Interoperabilidad		El Meta-Self-MDA debe implementar interoperabilidad mediante el desarrollo de diferentes interfaces que faciliten la comunicación con otros sistemas y que den soporte a diferentes estándares médicos. Por ejemplo, cuando un nuevo paciente llega a un hospital se realiza un proceso para realizar la persistencia del paciente, primero se ingresan sus datos personales (nombre, edad, peso, nss, etc), después se ingresa con el médico tratante, el cual es el encargado de otorgar un diagnóstico y persistir la información del paciente, esta información que se ha guardado en la base de datos es la que ocupará el Meta-Self-MDA y si la información se ha persistido bajo un estándar médico y el Meta-Self-MDA no implementa ese estándar, no se podrá realizar el objetivo para el cual fue diseñado.

4.3.1.2 Vista de escenarios: Meta-S-MDA

Principalmente el diseño de la arquitectura comienza por el levantamiento de requerimientos, más específicamente por: objetivos, necesidades, características particulares del sistema y atributos de calidad. Los requerimientos son transformados en un conjunto de casos de uso o escenarios. Para el diseño del Meta-S-MDA se ha comenzado a desarrollar la vista de escenarios.

Diagrama de casos de uso

De la definición mostrada en la Sección 4.3.1 se han obtenido los casos de uso primarios del componente Meta-S-MDA. La Figura 19 ilustra la relación entre los actores y los casos de uso primarios y secundarios. El caso de uso (CU-0) Filtrar el diagnóstico de pacientes ilustra un caso de uso primario, que se encuentra en un primer nivel del diagrama.

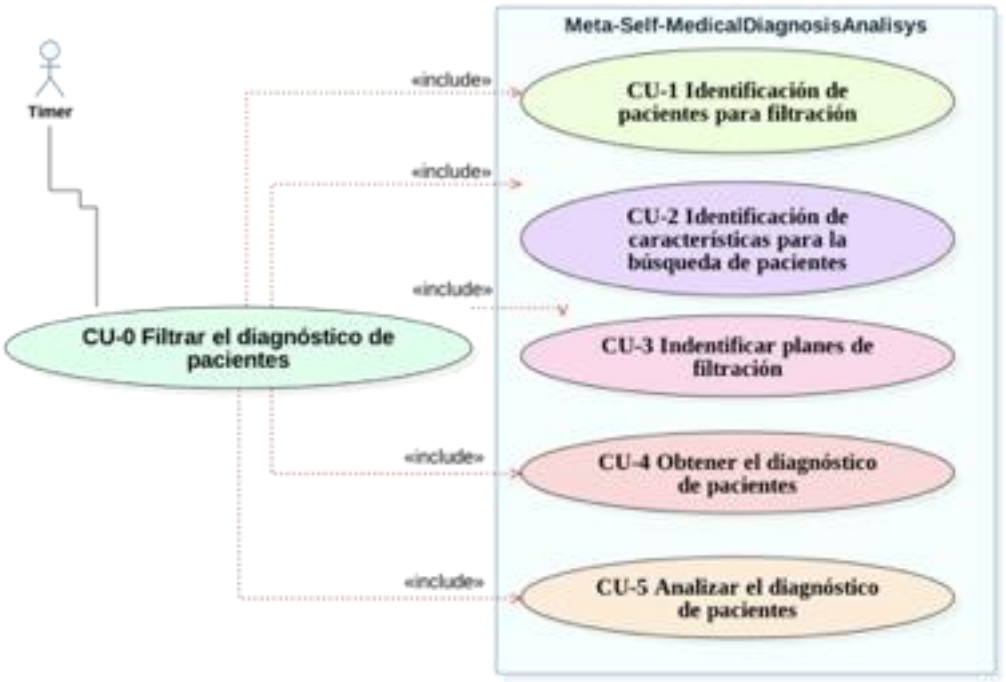

Figura 19. Diagrama de casos de uso del Meta-S-MDA.

Tomando como referencia el CU-0 de la Figura 19, mostraremos el entorno de operación en el cual se podría aplicar dicho caso de uso.

Basado en la información brindada por la Secretaría de Salud del Estado de México, tenemos que un hospital recibe un promedio de 700 a 800 pacientes por día en un horario de 7:00 AM a 7 PM. El primer proceso que se realiza en un hospital es la toma de información personal del paciente, como es: nombre completo, número de seguro social, peso, estatura, edad. Estos datos son ingresados a una base de datos de forma digital, pero también se realiza de forma manual persistiendo la información en formularios, que posteriormente son archivados. El siguiente proceso que se realiza es la revisión del paciente con un médico, este es el encargado de asignar un diagnóstico y en caso de que el paciente requiera de atención especializada, el médico canalizará al paciente con el personal correspondiente. En los hospitales el proceso de asignación de médico especialista tarda un promedio de 4 meses (información brindada por la Secretaría de salud del Estado de México).

El caso de uso que estamos proponiendo tiene la finalidad de identificar nuevos pacientes potenciales de una especialidad médica y de filtrar el diagnóstico asignado por el médico, es decir, una vez que se le ha asignado un diagnóstico a un paciente, este se podrá filtrar en base a las características obtenidas a partir de dicho diagnóstico y con la finalidad de evitar la intervención humana, se utilizará la computación autonómica. Por lo anterior, el caso de uso se convierte en un escenario autonómico el cual podrá ser documentado utilizando la plantilla SAQAS.

La *Tabla 30* muestra el escenario autonómico, filtrar el diagnóstico de pacientes, el cual indica el funcionamiento de cada uno de los componentes.

Para un mejor entendimiento de cómo funciona un escenario autonómico, se describirán cada uno de los elementos que lo conforman. A continuación, se muestra un ejemplo utilizando el escenario autonómico filtrar el diagnóstico de pacientes: el Sensor detecta que ha habido un cambio en la base de datos (han llegado nuevos pacientes potenciales de ser filtrados), y notifica al Monitor del cambio. Cuando el Monitor es notificado de un cambio se lanza el ciclo autonómico. El Monitor procede a verificar todos los accesos a la Data-Base desde su última fecha de acceso obteniendo los nuevos pacientes. El Monitor obtiene una lista de pacientes que envía al Analyzer, este se encarga de recuperar y filtrar toda la información de los pacientes, sintetizar y generar reportes de cada uno de ellos, además de realizar la selección de pacientes potenciales para la asignación de un médico especializado. El Analyzer envía la información al Planner, para que seleccione un plan de acuerdo con el análisis recibido. Una vez seleccionado el plan, se envía al Executor el cual ejecuta el plan y cierra el ciclo autonómico.

Tabla 24 Plantilla SAQAS para el Meta-Self-MedicalDiagnosisAnalysis Component

Source	Médico	
Stimulus	Filtrar el diagnóstico de pacientes	
Artifact	EMR (Electronic Medical Record)	
Environment	Tiempo de ejecución	
Response	Sensor	Detecta que ha habido un cambio en EMR-Data-Base y envía una notificación al Monitor.
	Monitor	• Cuando recibe la notificación del Sensor lanza el ciclo autonómico. • Verifica todos los accesos al EMR-Data-Base desde su última fecha de acceso obteniendo los nuevos pacientes por su idPatient. • Envía una lista de pacientes al elemento Analizar.
	Analyzer	• Recibe la lista. • Recuperar la información de los pacientes y la sintetiza generando un reporte de cada uno de los pacientes. • Envía la información al elemento Planner
	Planner	• Selecciona el plan adecuado para realizar la filtración de pacientes de acuerdo con el análisis recibido. • Envía el plan al Executor.
	Executor	Ejecuta el plan
	Knowledge	Guarda los planes del Planner y los reportes generados por el Analyzer

La Figura 20 muestra la ejecución del escenario autonómico filtrar el diagnóstico de pacientes de forma descriptiva. Como primer paso se observa a un paciente proporcionando información general. Como segundo paso el médico se encarga de asignar un diagnóstico y como paso final se observa el procedimiento por el cual tiene que pasar el diagnóstico médico

asignado al paciente mediante la ejecución del ciclo autonómico, como se observa en la Figura 20 se presentan los 5 elementos del ciclo autonómico y la comunicación que existe entre cada uno de los elementos, además se indica que se puede establecer la comunicación con sistemas externos, esto se realizará siempre bajo un estándar médico de intercambio de información. Por ejemplo: HL7 (Health Level Seven) / CDA, DICOM (Digital Imaging and Communication in Medicine), SNOMED-CT, CIE 10 (Clasificación Internacional de Enfermedades), con el objetivo de evitar problemas de comunicación entre sistemas.

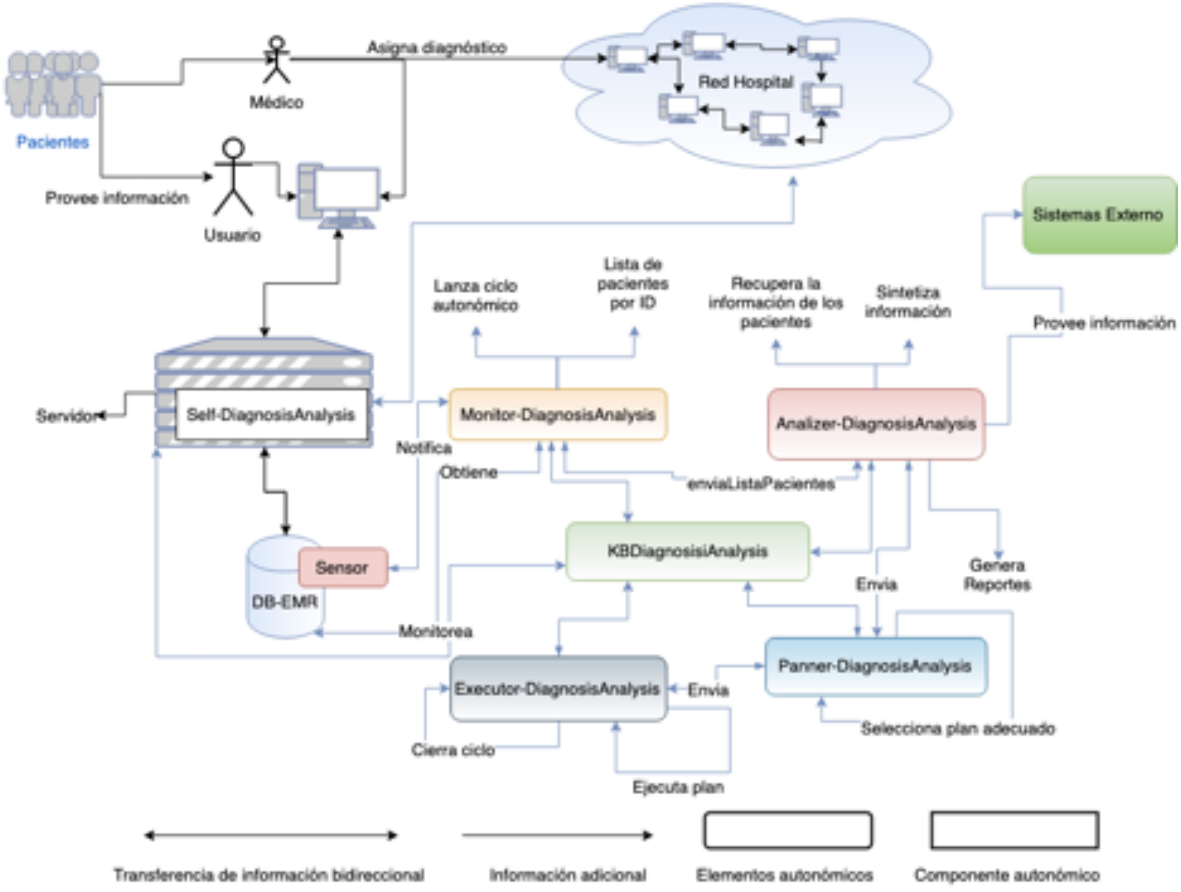

Figura 20. Filtrar el diagnóstico de pacientes.

4.3.1.3 Vista lógica: Meta-S-MDA

Para completar el diseño de la arquitectura autonómica se diseña la vista lógica del Meta-S-MDA, enfocada principalmente en ver el dominio del problema en forma de clases u objetos, es decir, se descompone el problema en una serie de abstracciones claves (clases). Esta descomposición se realiza para potenciar el análisis funcional e identificar mecanismos y elementos de diseño comunes a diversas partes, en este caso apoya principalmente al escenario autonómico filtrar el diagnóstico de pacientes.

Diagrama de clase del Meta-S-MDA

La Figura 21 representa el diagrama de clases asociado al Meta-Self-MDA, el diagrama está conformado por un conjunto de clases enfocadas en dar soporte al escenario autonómico

filtrar el diagnóstico médico de pacientes. A continuación, se describe cada una de las clases de alto nivel que conforman el diagrama.

Class Human: contiene las características generales asociadas a una persona de acuerdo con la clasificación de OpenGALEN [34].

Class Patient: hereda de la clase humano los términos generales como son: nombre, sexo, edad, etc. Además, agrega información particular del paciente, como es: número de seguro social, diagnóstico médico, correo electrónico, número de teléfono, entre algunos otros términos y métodos que complementa la información del paciente.

Class DAO-Patient: se encarga de realizar las acciones de persistencia de los pacientes.

Class Studies: contiene la clasificación de los estudios asociados a un paciente.

Class Diagnostic: contiene la información del diagnóstico como es: nombre del diagnóstico, paciente asociado, fecha de asignación y descripción del diagnóstico.

Class Condition: contiene información del estado del paciente a partir del diagnóstico asignado.

Class Doctor: contiene la información asociada a un médico como es: nombre, edad. sexo, entre algunos otros términos.

Class Sensor-MedicalDiagnosisAnalysis: detecta los movimientos registrados en la base de datos y cada vez que se registra un movimiento notifica al Monitor (Ver Apéndice A).

Class Monitor-MedicalDiagnosisAnalysis: es el encargado de lanzar el ciclo autonómico, esta clase recibe notificaciones del Sensor. También se encarga de enviar notificaciones a la clase Analyzer.

Class Analyzer-MedicalDiagnosisAnalysis: se encarga de analizar el diagnóstico de pacientes y generar reportes que envía a la clase planner.

Class DAO-Analyzer: encargada de guardar los reportes generados por la clase Analyzer.

Class Planner-MedicalDiagnosisAnalysis: encargada de seleccionar el plan adecuado de acuerdo con el reporte generado por la clase Analyzer, la clase P-MDA también se encarga de cerrar el ciclo autonómico.

Class DAO-Planner: encargada de guardar los planes de clasificación de pacientes.

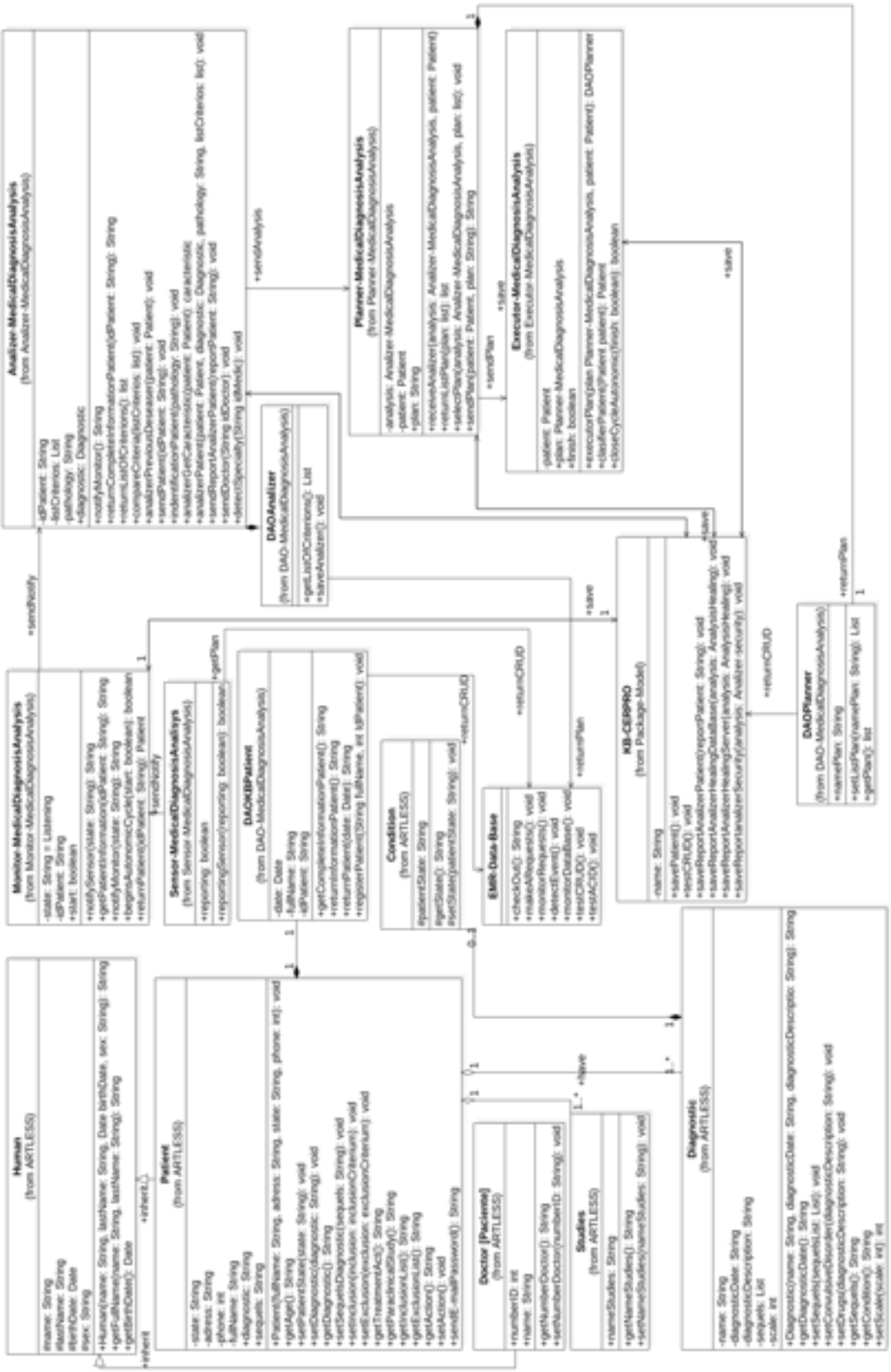

Figura 21. Diagrama de clases del Meta-S-MD

Diagrama de secuencia del Meta-S-MDA: *filtrar el diagnóstico de pacientes*

La Figura 22 representa el diagrama de secuencia asociado al escenario autonómico *filtrar el diagnóstico de pacientes*. Nótese que el diagrama de secuencia representa un caso autonómico, por lo que el usuario principal no está representado por un administrador o un usuario del sistema, como se muestra en la Figura 22. El Timer toma el lugar del usuario principal y en un momento determinado comienza la ejecución con la clase Sensor. Recordando que un diagrama de secuencia muestra cómo interactúa un conjunto de objetos en tiempo de ejecución. A continuación, se describe cada uno de los eventos dentro del diagrama de secuencia.

Eventos de la ejecución del escenario autonómico *filtrar el diagnóstico de pacientes* representado en tiempo de ejecución:

1. En un momento determinado el Timer comienza la ejecución.
2. El Sensor detecta un evento en la base de datos.
3. El Sensor detecta un cambio en la base de datos.
4. El Sensor envía una notificación al Monitor.
5. El Monitor recibe la notificación y lanza el ciclo autonómico.
6. El Monitor pide al DAOPatient la lista de pacientes desde su última fecha de acceso.
7. El DAOPatient realiza una petición a la base de datos.
8. La base de datos regresa la lista de pacientes al DAO.
9. El DAOPatient regresa la lista de pacientes por ID al Monitor.
10. El Monitor envía la lista de pacientes por ID al Analyzer.
11. El Analyzer pide al DAOPatient la información completa de la lista de pacientes que tiene.
12. El DAOPatient realiza una petición a la base de datos.
13. La base de datos regresa la información solicitada al DAOPatient.
14. El DAOPatient regresa la información al Analyzer.
15. Analiza la información y genera reportes.
16. Envía el reporte generado al Planner.
17. El Planner hace una petición al DAOPlanner para obtener los planes.
18. El DAOPlanner regresa la información solicitada.
19. El Planner selecciona el plan en base al informe recibido por el Analyzer.
20. El Executor ejecuta el plan.
21. El Executor cierra al ciclo autonómico.

El diagrama de casos de uso (ver Sección 4.3.1.2) tiene un caso de uso primario el cual involucra a 5 casos de uso secundarios, estos dan soporte al caso de uso primario. Es decir, es necesario realizar la construcción de los 5 casos de uso, para desarrollar el caso de uso primario. Como primer paso, los 5 casos de uso secundarios serán transformados en casos de uso autonómicos haciendo uso de la plantilla SAQAS y posteriormente se podrá pasar a la construcción de los diagramas de secuencia con elementos autonómicos, para finalmente realizar el caso de uso primario.

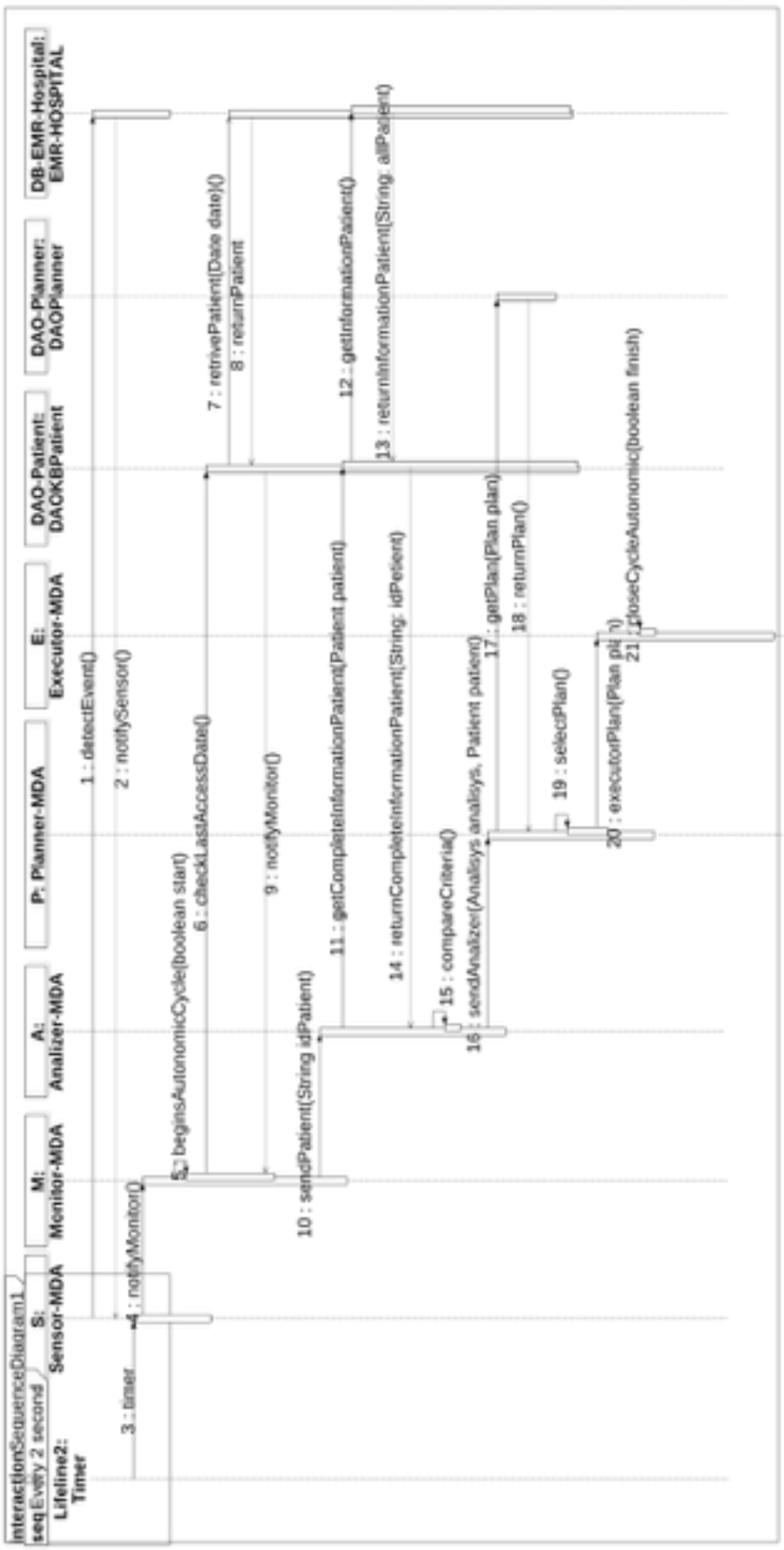

Figura 22. Diagrama de secuencia del Meta-S-MDA.

4.3.1.4 Vista de procesos: Meta-S-MDA

Diagrama de actividades: Meta-S-MDA

El diagrama de actividades del Meta-S-MDA tiene como propósito modelar el comportamiento del sistema, más específicamente modelar el escenario autonómico, *filtrar el diagnóstico de pacientes*. La Figura 23 muestra el diagrama de actividades. Como entrada del diagrama de actividades se recibe un nuevo paciente y se realiza el registro. El Sensor encargado de monitorear el registro de nuevos pacientes detecta un evento y notifica al Monitor, este recibe una notificación y se encarga de obtener la lista de pacientes desde su última fecha de acceso y la envía al Analyzer. El Analyzer recupera la información de los pacientes, la sintetiza y genera reportes, los cuales envía al Planner. El Analyzer puede establecer la comunicación con sistemas externos, esto se representa por medio de un rombo vacío (ver *Tabla 23*). El Planner recupera el reporte generado por el Analyzer y selecciona el plan adecuado, este lo envía al Executor. Este ejecuta el plan seleccionado, esta última acción se representa por una flecha hacia el mismo elemento Executor. También el diagrama muestra un elemento central llamado: base de conocimiento el cual establece una comunicación bidireccional y es el encargado de que el sistema aprenda.

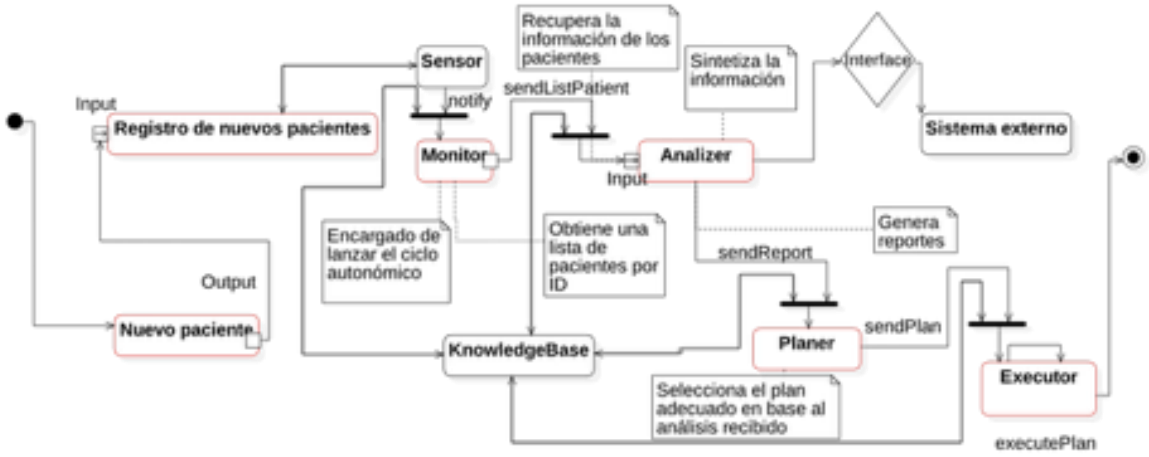

Figura 23. Diagrama de actividades del Meta-S-MDA.

4.3.1.5 Vista de despliegue: Meta-S-MDA

La vista de despliegue del Meta-S-MDA se centra en la organización de la arquitectura, la cual aplica el dicho divide y vencerás. El diseño de la arquitectura se representa en partes pequeñas que pueden ser desarrollados por uno o un grupo pequeño. Los sub-sistemas se organizan en una jerarquía de capas, cada una de las cuales brinda una interfaz bien definida hacia las capas superiores.

Diagrama de paquetes del Meta-S-MDA

El objetivo de crear un diagrama de componentes de alto nivel del Meta-Self-MDA es tener una visión más clara de todos los elementos genéricos que componen la arquitectura,

organizando la arquitectura en capas, agrupando los elementos autonómicos y detallando las relaciones de dependencia entre ellos.

El mecanismo de agrupación por el cual se agrupa el diagrama de componentes se llaman paquetes, más estrictamente los paquetes y sus dependencias son elementos de los diagramas de casos de uso, vistos en la Sección 4.3.1.2 el diagrama de componentes del Meta-S-MDA es una extensión del diagrama de casos de uso y del diagrama de clases.

La Figura 24 muestra el diagrama de componentes asociado al Meta-S-MDA, que contiene cuatro paquetes. Un paquete se representa mediante un símbolo con forma de 'carpeta' en el que se coloca el nombre en la pestaña y el contenido del paquete dentro de la 'carpeta'. En los casos en que no sea visible el contenido del paquete se podrá colocar en lugar del nombre.

Si el paquete tiene definido un estereotipo, este se representa encima del nombre entre el símbolo << ... >>, y si se definen propiedades, se representan debajo del nombre y entre llaves. Recordando que un paquete es una agrupación de elementos, bien sea casos de uso, clases o componentes. Los paquetes pueden contener a su vez otros paquetes anidados que en última instancia contendrá alguno de los elementos anteriores, como se puede ver en la Figura 24, en el paquete 1, 2, 3 y 4 contienen paquetes anidados y estos contienen un conjunto de componentes y clases.

La Figura 24 también establece una arquitectura en capas fomentando la alta cohesión y el bajo acoplamiento, ya que la primera capa está enfocada en soportar las reglas de negocio, la segunda en establecer la comunicación con la base de datos, la tercera enfocada en enviar notificaciones a través de un sensor y la última dedicada a realizar la persistencia de la información.

La primera capa del diagrama de componentes del Meta-S-MDA contiene un componente dentro de un paquete y dentro de este, se tienen 4 componentes más que contienen cuatro clases, estas son las encargadas de ejecutar el ciclo autonómico. Como se puede observar, las clases establecen dependencias entre ellas, representadas por medio de una flecha punteada, como en el diagrama de clase, pero ahora un hecho interesante es que el Sensor y Monitor se encuentran en paquetes diferentes, por lo que para establecer la comunicación se ha destinado un puerto especializado.

La segunda capa muestra el paquete donde se encuentran los DAO's encargados de realizar la persistencia de la información, como se muestra en la Figura 24. La comunicación con la capa de negocio se establece por medio de una interfaz bien definida y la comunicación con las bases de datos se establece por medio de una dependencia simple indicada por una flecha punteada.

La tercera capa corresponde al paquete donde se encuentra el Sensor, encargado de enviar notificaciones de cualquier evento que suceda en la base de datos. Las notificaciones serán enviadas por el puerto 1.

La cuarta y última capa se encuentra la base de conocimiento y la base de datos del hospital encargados de proporcionar la información.

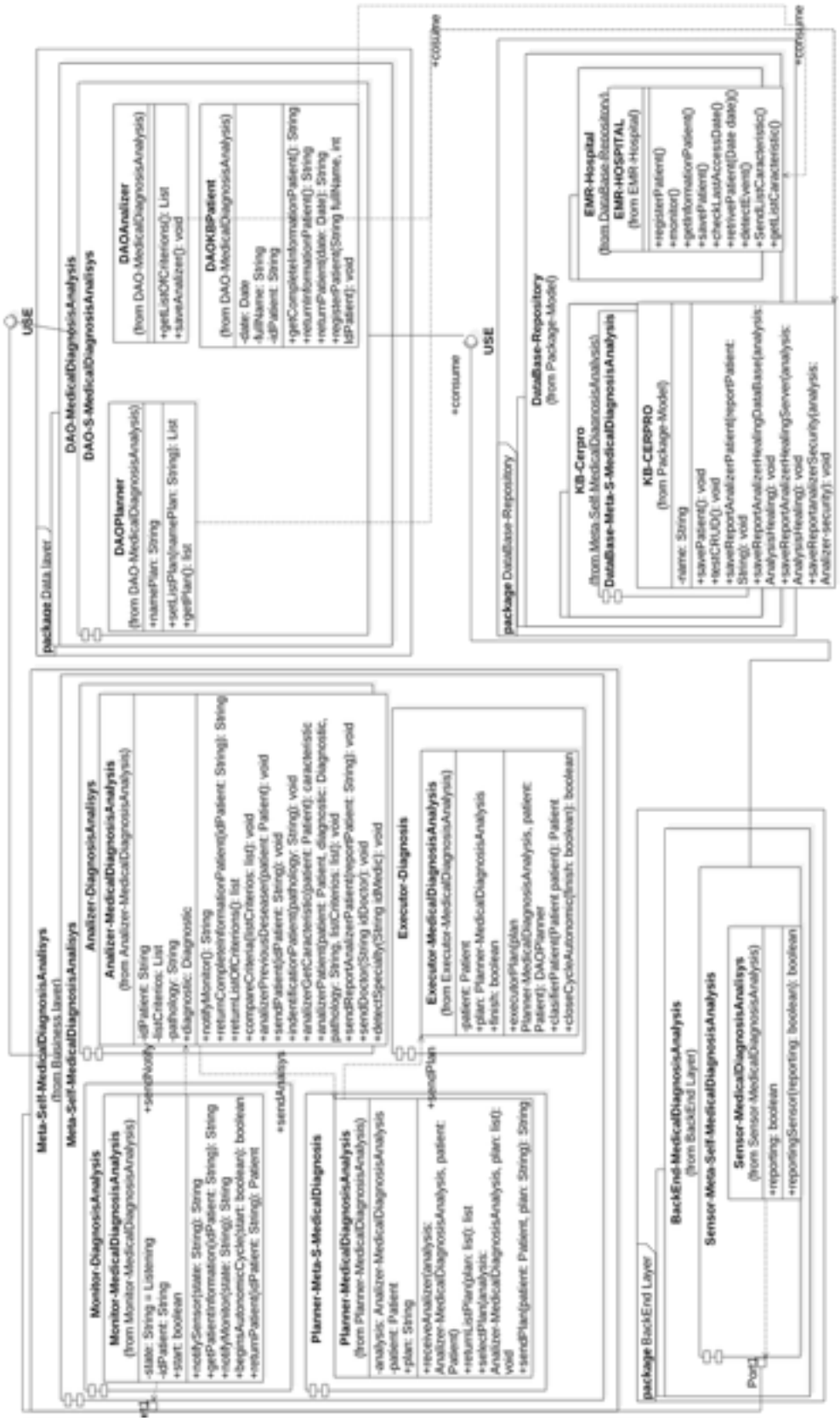

Figura 24. Diagrama de componentes del Meta-S-MDA.

4.3.1.6 Vista física del Meta-S-MDA

Tomando en cuenta los atributos de calidad asociados al Meta-Self-MDA, tales como la disponibilidad, confiabilidad (tolerancia a fallos) y la escalabilidad, se mostrará el diseño de la arquitectura para satisfacer dichos atributos.

Diagrama de implantación del Meta-S-MDA

El diagrama de implantación del Meta-Self-MDA permite visualizar la arquitectura física del hardware, software y todos los artefactos que se implementan dentro de la arquitectura como los patrones. El diagrama del Meta-S-MDA puede entenderse como lo contrario de los escenarios autonómicos, ya que ilustran la forma física del sistema, en lugar de representar conceptualmente los usuarios y dispositivos que interactúan dentro de la arquitectura.

La Figura 25 representa el diagrama de implantación del Meta-Self-MDA, el cual contiene 6 nodos. Un nodo puede contener objetos, instancias y componentes. Además, permiten establecer relaciones entre nodos mediante un canal de comunicación. A continuación, se describe el funcionamiento de cada uno de los nodos:

Nodo DAO-Meta-S-MDA: nodo encargado de atender las peticiones de cada uno de los componentes encontrados en el nodo Meta-S-MDA. Este nodo también implementa un patrón llamado Pausa-Reanudar indicada en el diagrama por un componente en color rojo, que conecta al DAO y a la base de datos. El objetivo de implementar esta táctica es evitar la pérdida de tiempos prolongados (Tiempo > 10 min) al obtener información de pacientes. Por ejemplo, supongamos que en un momento determinado el DAO ha realizado una petición a la base de datos de un archivo médico, este tiene un peso de 3 GB, ahora el archivo comienza a enviarse, pero en un momento la computadora ha tenido una falla y se ha apagado, el archivo llevaba un tiempo de descarga de 20 minutos, por lo que cuando se reinicie la computadora el archivo empezará a descargarse de nuevo desde el principio, la aplicación de Pausa-Reanudar nos permite evitar este tipo de situaciones, reanudando la descarga del archivo desde dónde se quedó descargando.

Nodo Back-End-Meta-S-MDA: encargado de detectar eventos en la base de datos y notificar al componente Monitor encontrado dentro del nodo Meta-S-MDA.

Nodo DataBase-Meta-S-MDA: encargado de proporcionar la información requerida por el DAO.

Nodo Meta-S-MDA: encargado de ejecutar y finalizar el ciclo autonómico, dentro de este nodo se encuentran 4 componentes encargados de ejecutar la funcionalidad del sistema, más específicamente ejecutar el escenario autonómico *filtrar el diagnóstico de pacientes*.

Nodo Patrón-Mediador: encargado de establecer la comunicación con otros sistemas, sin generar fuertes acoples.

Nodo Another-System: establece la conexión con el sistema principal.

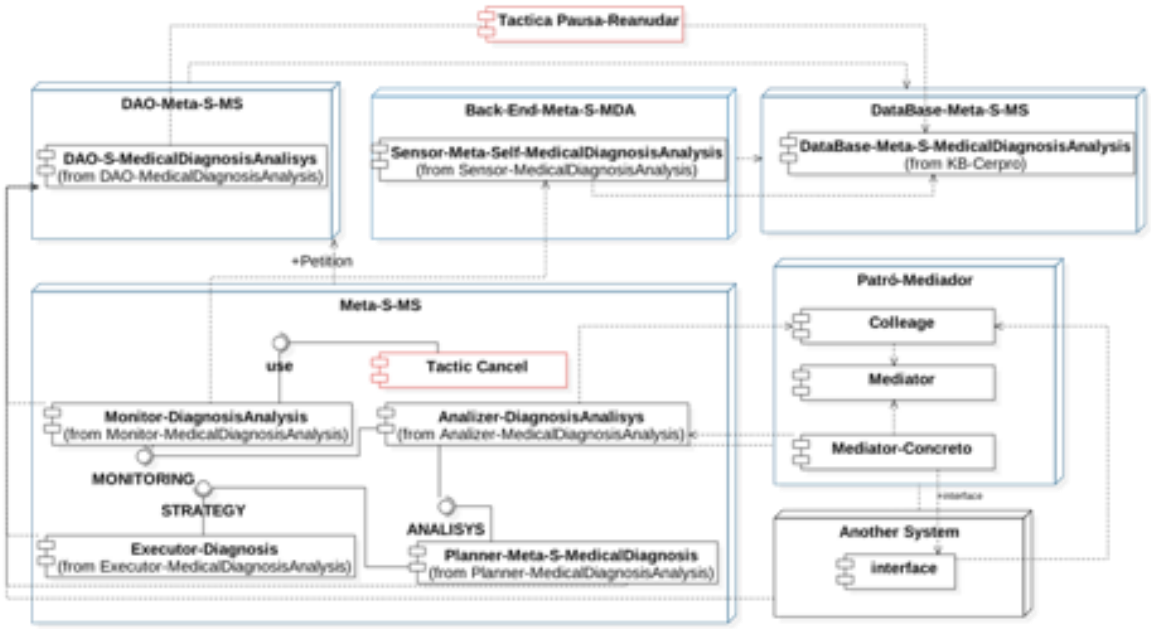

Figura 25. Diagrama de implantación del Meta-S-MDA.

4.3.2 Diseño del Meta-Self-MedicalSecurity

Basado en el análisis del estado del arte encontramos que los sistemas de información para el cuidado de la salud son deficientes en términos de seguridad. Esto debido al poco personal calificado para trabajar con sistemas de seguridad y de salud, sumado a esto encontramos que los fondos dedicados a solucionar problemas de seguridad para los sistemas médicos son insuficientes. Un artículo publicado por Symantec en 2018 sobre el tema "Cyber Security and Healthcare: an Evolving Undestanding of Risk", advierte que los ciber riesgos de atención médica están en aumento, esto en parte porque los datos médicos se han convertido en objetos de gran valor, poca seguridad y actualización que ofrecen los sistemas de hospitales.

Tomando en cuenta lo anterior mencionado, se decidió diseñar un componente autonómico enfocado a la seguridad de la información médica. En la Sección 4.3 se ha proporcionado una definición de seguridad, pero ahora detallaremos los campos que estará abarcando el meta-componente.

Autenticación: el grado en que se puede demostrar que la identidad de un sujeto o recurso es la indicada. Es decir, el Meta-S-MS verificará y autenticará que los usuarios que intenten acceder al sistema tengan las credenciales para realizarlo.

Confidencialidad: el grado de protección contra la divulgación no autorizada de datos o información, ya sea accidental o deliberada. Es decir, el Meta-S-MS protegerá la información de pacientes de modo que el sistema no permite realizar copias de información de ningún tipo.

Integridad: el grado en que un sistema o componente impide la modificación de programas o datos informáticos. Es decir, el Meta-S-MS, protegerá la información de pacientes, de modo que la información que se persista en la base de datos no se pueda modificar, al menos que el usuario cuente con el nivel de acceso para realizar dicho proceso.

Siguiendo el método de diseño ADD presentado en la Sección 3.2.2.1, se realizará el diseño del Meta-S-MS enfocado a cubrir los campos de seguridad antes mencionados. Este componente se ha diseñado mediante el desarrollo de una serie de vistas, las cuales en términos generales abstraen conceptos comunes de varias arquitecturas y las elevan a un nivel superior.

4.3.2.1 Atributos de calidad del Meta-S-MS

La selección de los atributos de calidad para el Meta-S-MS se realizó tomando tres objetivos: 1) el análisis del estado del arte, 2) definición del meta-componente, 3) definición del atributo de calidad. La *Tabla 25* muestra la relación de los atributos de calidad y el meta-componente.

Tabla 25. Meta-S-MS a atributos de calidad

Meta-Self-MedicalSecurity	Seguridad	Confidencialidad	En un hospital la protección de la información de los pacientes es muy importante ya que la filtración de la información se podría utilizar para fines perjudiciales a los pacientes. Por lo que es importante que la información médica de los pacientes esté protegida contra la divulgación no autorizada.
		Autenticación	El Meta-Self-MS debe implementar autenticación ya que en un hospital se manejan grandes cantidades de información y no todos los usuarios deben de tener acceso a esta. Supongamos que un usuario malintencionado intenta ingresar al sistema, este usuario primero deberá autenticarse al sistema y en el caso de que el acceso sea denegado el usuario será bloqueado inmediatamente.
		Integridad	El Meta-Self-MS debe implementar integridad ya que los datos de los pacientes no deben ser modificados al menos que sea por el personal autorizado.
	Disponibilidad	Confiabilidad	El Meta-Self-MS es el encargado de proteger el sistema de posibles ataques internos como externos, por lo que este componente no debe fallar. En el momento que este componente falle, todo el sistema quedará vulnerable a ataques. Es por esta razón por la que el Meta-Self-MS debe implementar el atributo de seguridad de confiabilidad.
	Usabilidad	Manejo de errores	El Meta-Self-MS debe realizar el manejo de errores debido a que, si en un momento de operación el sistema HCIS de apoyo al diagnóstico médico sufre un error, el sistema quedara vulnerable a ataques.

4.3.2.2 Vista de escenarios: Meta-S-MS

El diseño de la arquitectura comienza por levantar requerimientos asociados a la seguridad de los sistemas de salud, más específicamente se buscan los objetivos particulares, las necesidades, características y atributos de calidad. Definir los casos de uso es una buena guía para obtener las características principales de un sistema médico enfocado al diagnóstico de pacientes. Por lo que para el diseño del Meta-S-MS se ha comenzado desarrollando una vista de escenarios.

Diagrama de casos de uso

De la definición en la Sección 4.3, el estado del arte (Sección 2) y los atributos de calidad, se han obtenido los casos de uso asociados al componente Meta-S-MS. La Figura 26 ilustra las relaciones de dependencia entre los actores y los casos de uso. Estas relaciones están representadas por medio de una flecha punteada.

Los casos de uso mostrados en la Figura 26 están enfocados a la seguridad de los sistemas médicos. A continuación, se muestra la descripción detallada de los casos de uso.

1. Realizar la autorización de usuarios, este caso de uso pertenece al campo de autenticación de seguridad. Para explicar mejor el funcionamiento de este caso de uso se plantea el siguiente ejemplo. Supongamos que, si un usuario desea tener acceso a la base de datos de un sistema médico, el usuario primero deberá autenticarse al sistema. Sí desea tener acceso a la información, en caso de que el usuario no cumpla con el nivel de acceso requerido para obtener información el sistema se bloqueará la cuenta de usuario.

2. Realizar la identificación de usuarios que han ingresado al sistema**,** este caso de uso pertenece al campo de confidencialidad de seguridad. Para explicar mejor el funcionamiento de este caso de uso se plantea el siguiente ejemplo. Supongamos que un usuario ha roto la seguridad del sistema médico y ha ingresado al sistema. El sistema verificará permanentemente las credenciales de los usuarios que han ingresado al sistema, de haber encontrado anomalías el sistema bloqueará al usuario temporalmente.

3. Realizar la autenticación de usuarios para la obtención de información, este caso de uso pertenece al campo de confidencialidad de seguridad. Para explicar mejor el funcionamiento de este caso de uso se plantea el siguiente ejemplo. Supongamos que un usuario que se ha autenticado al sistema desea obtener información de los pacientes, pero este no cuenta con el nivel de acceso requerido para obtener la información, por lo que, si el usuario intenta obtener más información de la que se le permite, se bloqueara temporalmente.

4. Realizar la detección de usuarios que han roto la seguridad del sistema, este caso de uso pertenece al campo de confidencialidad de seguridad. Para explicar mejor el funcionamiento de este caso de uso se plantea el siguiente ejemplo. Supongamos que un usuario no identificado ha accedido al sistema, y ha empezado a realizar copias de información de los pacientes, el sistema permite la detección y autenticación de todos los usuarios que accedan a la base de datos, y si un usuario no es autenticado se bloquea de forma inmediata.

5. Realizar la autenticación de usuarios para la modificación de información, este caso de uso pertenece al campo de confidencialidad e integridad. Para explicar mejor el funcionamiento de este caso de uso se plantea el siguiente ejemplo. Supongamos que un usuario que se ha autenticado al sistema accede a la base de datos y empieza a realizar modificación de información, el sistema detecta y verifica las credenciales del usuario que está modificando la información, y en caso de que el usuario no cuente con el nivel de acceso permitido se bloquea temporalmente.

Figura 26. Diagrama de casos de uso del Meta-S-MS.

Se ha realizado el levantamiento de requerimientos basado en atributos de calidad, estado del arte y la definición del Meta-S-MS, pero para tener un sistema autonómico con la mínima intervención humana los requerimientos deberán ser planteados como escenarios autonómicos. Por lo anterior, los casos de uso se mapearán a escenarios autonómicos, los cuales podrán ser documentados utilizando la plantilla SAQAS.

La *Tabla 26* muestra el escenario autonómico, realizar la autorización de usuarios y describe el funcionamiento de cada uno de los componentes que dan soporte al caso de uso.

Tabla 26. Plantilla SAQAS: Escenario-realizar la autorización de usuarios

Source	**User**	
Stimulus	Un usuario ha accedido al sistema del hospital e intenta acceder a la base de datos del hospital, pero ha olvidado su contraseña de acceso. El sistema solo permite realizar tres intentos para acceder, después de realizar los tres intentos el sistema verifica que sea un usuario con acceso, en caso de que se verifique correcto el sistema envía un formulario de recuperación de contraseña, en caso contrario bloqueara la cuenta del usuario.	
Artifact	Base de Datos	
Environment	Runtime	
Response	Sensor	El Sensor está escuchando cada vez que se produce un acceso o un intento de acceso a la base de datos a través del sistema HCIS.

		El Sensor envía notificaciones al componente Monitor cada vez que se produce un acceso a la base de datos o cada vez que se ha pasado el número de internos permitidos.
	Monitor	El componente Monitor se encarga de recibir las notificaciones enviadas por el Sensor, las procesa y monitorea por un periodo de tiempo el componente seleccionado. Sí detecta una anomalía en el sistema. Por ejemplo, se ha sobrepasado el número de intentos de acceso por un usuario a la base de datos, notifica al componente Analyzer.
	Analyzer	El componente Analyzer recibe las notificaciones del Monitor y las procesa, analiza la problemática encontrada y genera un reporte que envía al componente Planner. Por ejemplo, el usuario que ha olvidado su contraseña ha realizado los tres intentos de acceso a la base de datos, el Analyzer analiza si el usuario tiene acceso o no al sistema mediante la obtención de su dirección IP.
	Planner	Recibe el reporte generado por el Analyzer y en base a este se selecciona el plan más adecuado. Por ejemplo, el usuario que ha realizado más de tres intentos de acceso a la base de datos, el Analyzer ha identificado que es un usuario que tiene acceso al sistema, por lo que el plan será enviar un formulario para que recupere su contraseña, en caso contrario se bloquearía la dirección IP.
	Executor	Recibe el plan enviado por el Planner y lo ejecuta.
	Knowledge	Guarda el plan de mitigación y las características de los ataques y aprende sobre posibles ataques futuros.
Campo: Autenticación	El grado en que se puede demostrar que la identidad de un sujeto o recurso es la indicada	

La Figura 27 muestra la ejecución del escenario autonómico; *realizar la autorización de usuarios*, de forma descriptiva se observa a un usuario malintencionado que ha realizado la suplantación de identidad por lo que el sistema lo ha dejado acceder. El usuario ha accedido al sistema, pero no a la base de datos. Cuando ha intentado acceder a la base de datos, el sistema ha denegado el acceso, el usuario ha realizado el intento más de tres veces, el Sensor que está escuchando cualquier evento que suceda en la base de datos ha notificado al Monitor, este ha comenzado a monitorear al usuario desde el segundo intento de acceso. El Monitor ha notado anomalías de un usuario que desea acceder y ha enviado una notificación al Analyzer, este se encarga de analizar la situación y genera un reporte que envía al Planner. Este recibe el reporte del usuario que ha ingresado al sistema, se ha detectado que es un usuario sin credenciales para tener acceso al sistema, por lo que el Planner ha seleccionado el plan de bloquear al usuario permanentemente, el plan se ha enviado al Executor, el cual se encarga de bloquear al usuario.

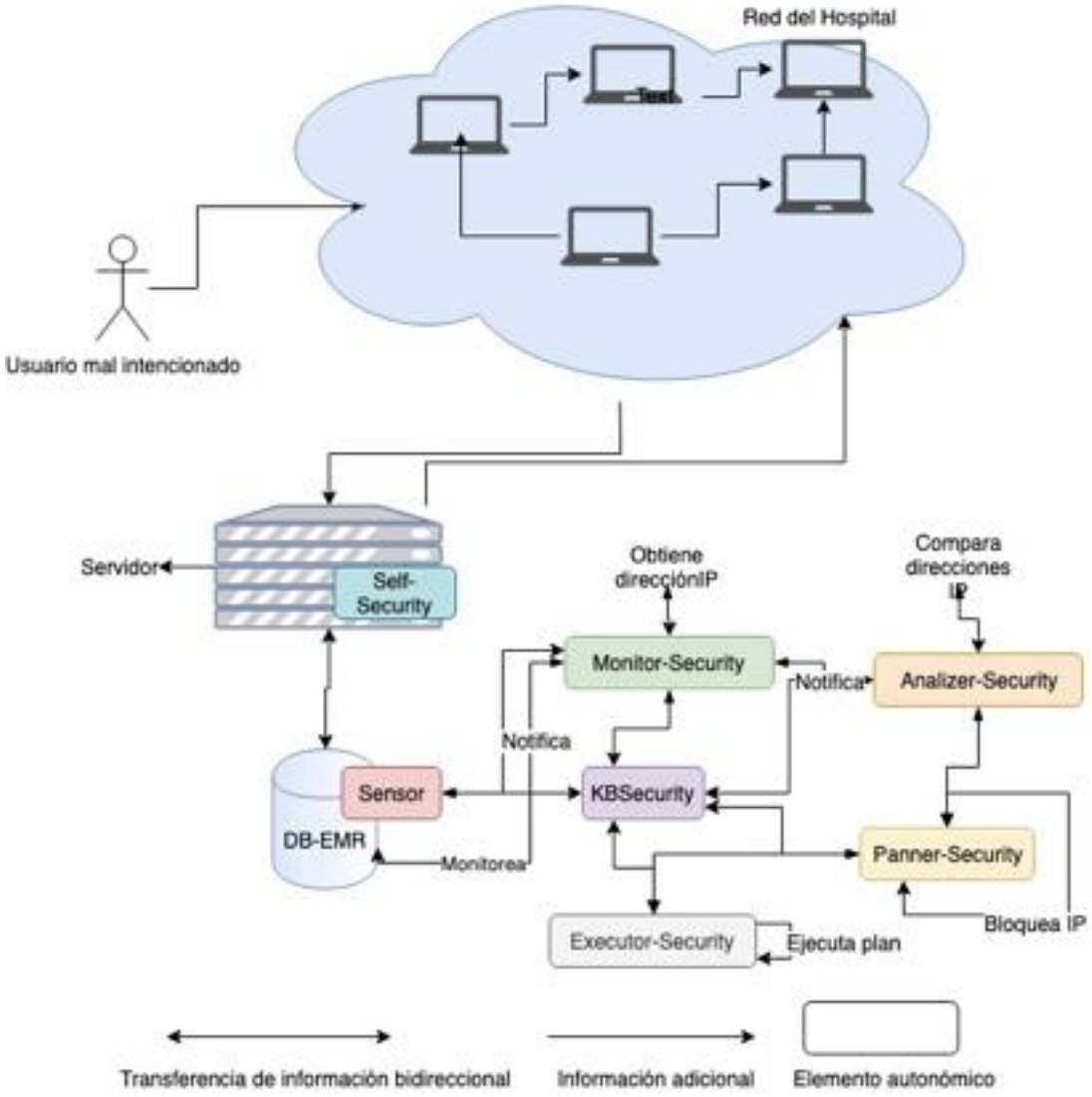

Figura 27. Escenario autonómico de seguridad.

4.3.2.3 Vista lógica: Meta-S-MS

La vista lógica del componente Meta-S-MS está enfocada principalmente en ver el dominio del problema en forma de clase u objetos, es decir, se descompone el problema en una serie de clases. Esta descomposición se realiza principalmente para identificar mecanismos y elementos de diseño que ayuden a la solución de problemas de arquitectura.

Diagrama de clase del Meta-S-MS

El diagrama de clases del Meta-S-MS, ilustrado en la Figura 28, muestra las clases necesarias para implantar los escenarios autonómicos.

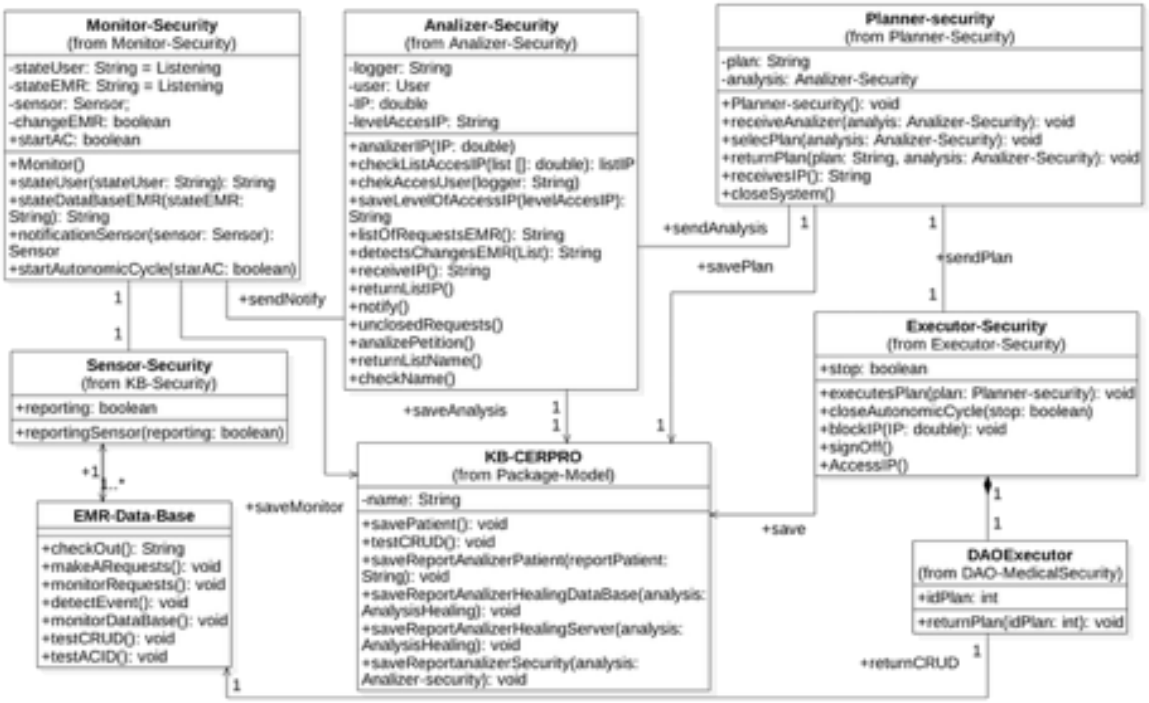

Figura 28. Diagrama de clases del Meta-S-MS.

Class Sensor-Security: encargado de detectar cualquier evento dentro de la base de datos (ver Apéndice A).

Class Monitor-Security: encargado de recibir las notificaciones enviadas por el Sensor, monitorear a los usuarios que han ingresado al sistema y notificar al Analyzer al encontrar una anomalía.

Class DAOMonitor: encargado de persistir las notificaciones enviadas por el Sensor, las notificaciones generadas son guardadas por un periodo de tiempo (30 días).

Class Analyzer-Security: encargado de recibir las notificaciones enviadas por el Monitor y analizar a los usuarios que han accedido al sistema o a la base de datos, en respuesta a las acciones realizadas por los usuarios genera reportes de estos mismos y los envía al Planner.

Class Planner-Security: recibe el análisis enviado por el Analyzer y seleccionar el plan adecuado en base al reporte obtenido, una vez que ha seleccionado el plan lo envía al Executor.

Class Executor-Security: encargado de recibir los planes enviados por el Planner y ejecutarlos, esta clase también se encarga de finalizar el ciclo autonómico.

Class EMR-Data-Base: contiene la información del EMR de los pacientes del hospital.

Class KB-CERPRO: encargado de proporcionar la inteligencia del sistema autonómico.

Diagrama de secuencia del Meta-S-MS: realizar la autenticación de usuarios

A manera de ejemplo se presenta el diagrama de secuencia asociado con el escenario autonómico: realizar la autenticación de usuarios, esto con el objetivo de ver cómo interactúan los elementos autonómicos.

La Figura 29 representa un diagrama de secuencia de un escenario autonómico asociado al Meta-S-MS. A continuación, se describe cada una de las interacciones del diagrama de secuencia.

1. En un momento determinado el timer comienza la ejecución.
2. El Sensor encargado de detectar eventos dentro de la base de datos.
3. Se envía una notificación al Sensor.
4. El Sensor envía una notificación al Monitor.
5. El Monitor al recibir una nueva notificación lanza el ciclo autonómico.
6. El Monitor, monitorea las peticiones realizadas a la base de datos por el usuario que se está monitoreando y envía una notificación al Analyzer.
7. Analiza las transacciones realizadas por el usuario y hace una petición para obtener su dirección IP.
8. Retorna la dirección IP del usuario.
9. El Analyzer verifica sí la IP del usuario tiene acceso al sistema y genera un reporte que envía al Planner.
10. El Planner recibe el reporte generado por el Analyzer.
11. El Planner manda una petición de recuperación de planes.
12. Regresa una lista de planes.
13. El Planner selecciona el plan adecuado en base al reporte generado por el Analyzer.
14. El Planner envía el plan al Executor.
15. El Executor recibe el plan y lo ejecuta.
16. El Executor cierra el ciclo autonómico.

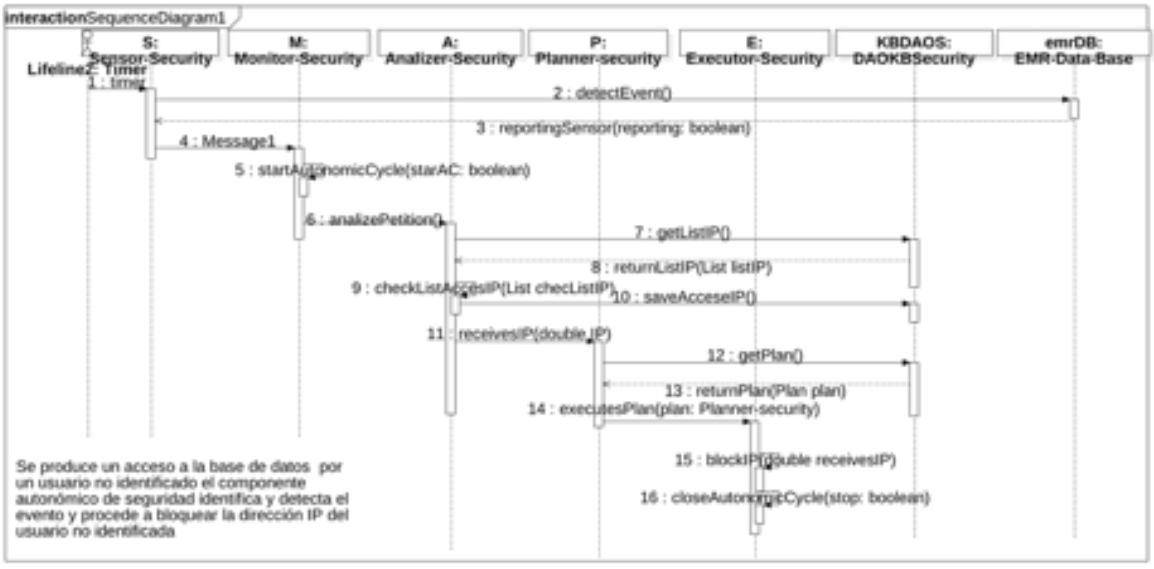

Figura 29. Diagrama de secuencia del Meta-S-MS.

Como se muestra en el diagrama de casos de uso de la Sección 4.3.2.2 existen 5 casos de uso primarios, los cuales se transforman en escenarios autonómicos y dan soporte al Meta-S-MS. Es decir, es necesario realizar la construcción de los 5 casos de uso primarios, como primer paso, serán transformados en casos de uso autonómicos haciendo uso de la plantilla SAQAS y posteriormente se podrá pasar a la construcción de los diagramas de secuencia con elementos autonómicos.

4.3.2.4 Vista de procesos: Meta-S-MS

Diagrama de actividades: Meta-S-MS

El diagrama de actividades del Meta-S-MS tiene como propósito modelar el comportamiento del sistema y, más específicamente, modelar el escenario autonómico, realizar la autenticación de usuarios. La Figura 30 muestra el diagrama de actividades. Como entrada del diagrama de actividades se tiene un nuevo usuario que quiere ingresar al sistema del hospital sin tener acceso, el usuario ha violado la seguridad del sistema médico y ha accedido. El usuario desea tener acceso a la base de datos del hospital por lo que realiza una inyección SQL a la base de datos. El Sensor que monitorea cualquier evento en la base de datos ha detectado un nuevo evento, el Sensor envía una notificación al Monitor. Este al recibir la notificación lanza el ciclo autonómico y monitoriza a los usuarios que se encuentran en el sistema HCIS de apoyo al diagnóstico médico, el Monitor detecta anomalías con algunos de los usuarios y solicita su dirección IP, el Monitor envía la lista de direcciones IP al Analyzer para que analice las últimas transacciones realizadas por los usuarios. El Analyzer genera reportes de las transacciones realizadas por cada uno de los usuarios, y envía los reportes al Planner. Este selecciona el plan adecuado en base al reporte, en este caso el reporte generado indica que un usuario no cuenta con las credenciales para tener acceso al sistema por lo que al ejecutar el plan. El usuario será bloqueado permanentemente del sistema, finalmente el Executor se encarga de cerrar el ciclo autonómico.

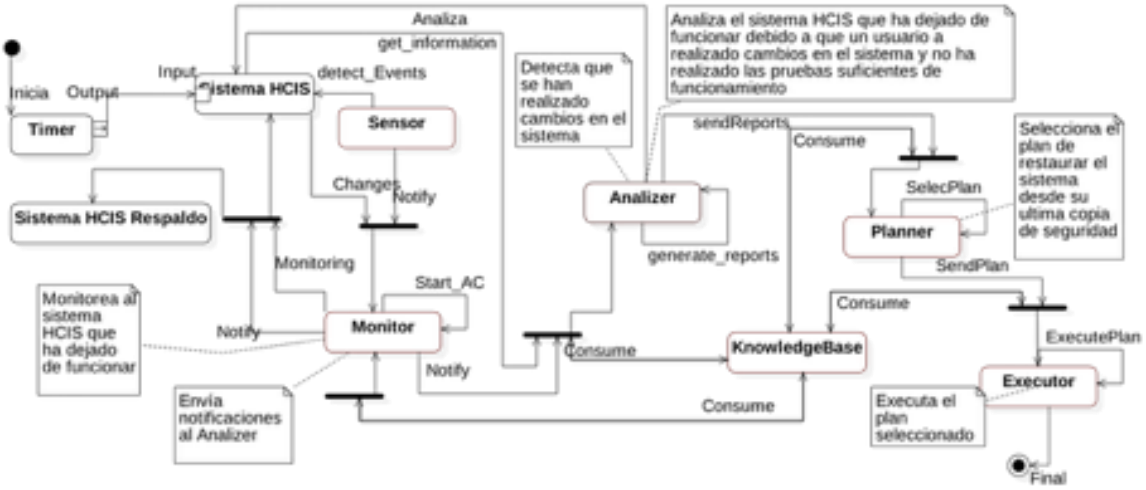

Figura 30. Diagrama de actividades del Meta-S-MS

4.3.2.5 Vista de despliegue: Meta-S-MS

La vista de despliegue del Meta-S-MS organiza la arquitectura en capas, aplicando el dicho de divide y vencerás.

Diagrama de paquetes del Meta-S-MS

El objetivo de crear un diagrama de paquetes de alto nivel del Meta-Self-MS es tener una visión más clara de todos los elementos que componen la arquitectura.

Para el diseño del Meta-S-MS se establece una arquitectura en capas la cual fomenta el principio de diseño de alta cohesión y el bajo acoplamiento, principalmente el diagrama de paquetes del Meta-S-MS está conformado por cuatro capas; la primera capa está encargada de soportar las reglas de negocio, es decir, se encarga de la funcionalidad del sistema, llevando a cabo los escenarios autonómicos propuestos para este componente. La primera capa contiene cuatro meta-componentes que a su vez contienen las cuatro clases encargadas de la seguridad de los sistemas HCIS de apoyo al diagnóstico médico. Como se puede observar (ver Figura 31), las clases contienen una serie de métodos enfocados a cubrir la seguridad de los sistemas HCIS y a ejecutar el ciclo autonómico. Es decir, las clases establecen un ciclo de comunicación entre ellas, es importante mencionar que si las clases que se encuentran en esta primera capa no establecieran un ciclo no se podría ejecutar el ciclo autonómico, la relación que establecen las clases es una relación de asociación indicada por una flecha.

La segunda capa muestra el paquete donde se encuentran los DAO's encargados de realizar la persistencia de la información del Meta-S-MS, como se muestra en la Figura 31. Esta capa está conformada por tres clases que se comunican con la capa de negocio por medio de una interface y la comunicación con las bases de datos se establece por medio de una dependencia simple se indica por una flecha punteada.

La tercera capa corresponde al paquete BackEnd-Layer donde se encuentra el Sensor-Security, encargado de enviar notificaciones de cualquier evento que suceda en la base de datos. (ver Figura 31).

La cuarta y última capa se encuentra en la base de conocimiento encargada de proporcionar el aprendizaje al componente Meta-S-MS y la base de datos del hospital encargados de proporcionar la información.

4.3.2.6 Vista física del Meta-S-MS

Tomando en cuenta los atributos de calidad asociados al Meta-Self-MS, tales como la seguridad, disponibilidad y usabilidad, se mostrará el diagrama de implantación con el objetivo de satisfacer los atributos de calidad.

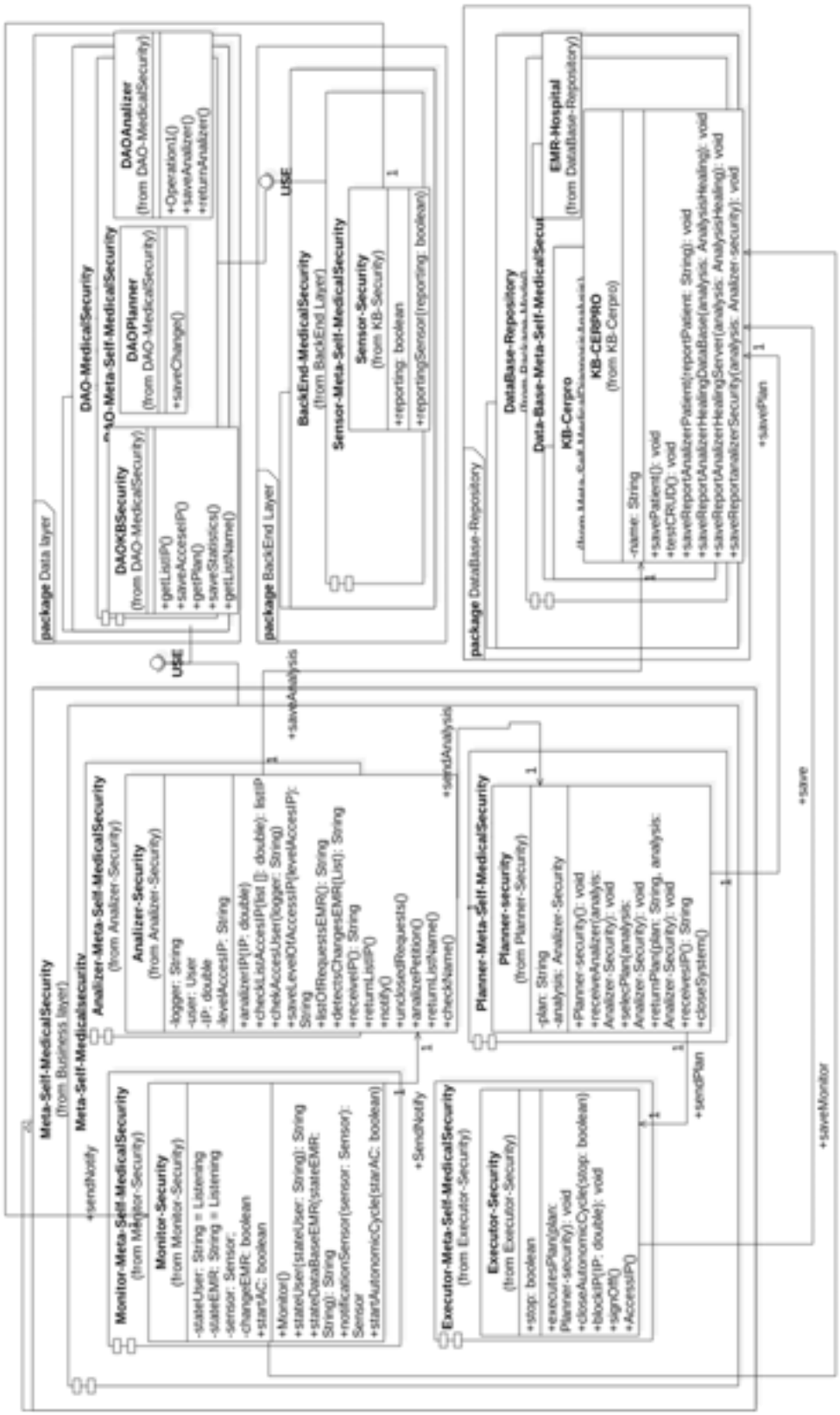

Figura 31. Diagrama de paquetes del Meta-S-MS.

Diagrama de implantación del Meta-S-MS

El diagrama de implantación del Meta-Self-MS permite visualizar la arquitectura física del hardware, software y todos los artefactos que se implementan dentro de la arquitectura como los patrones. La Figura 32 representa el diagrama de implantación del Meta-Self-MS, el cual contiene 8 nodos, recordando que un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional y generalmente tiene memoria y capacidad de procesamiento. A continuación, se describe el funcionamiento de cada uno de los nodos:

Nodo Meta-Self-MedicalSecurity: nodo encargado de la seguridad del sistema HCIS, dentro de este nodo se encuentra implementado el patrón MAPE-K, este se encarga de implementar un proceso que monitorea el sistema HCIS de cualquier posible ataque. Este nodo se encuentra conectado con el DAO el cual le permite tener acceso a la base de datos y al BackEnd el cual le permite escuchar cualquier evento que suceda dentro del sistema HCIS.

Nodo DAO-Meta-S-MS: nodo encargado de atender las peticiones de cada uno de los componentes encontrados dentro del nodo Meta-Self-MedicalSecurity.

Nodo Back-End: nodo encargado de detectar cualquier evento dentro del sistema HCIS, cuando este ha detectado un evento, envía una notificación al Monitor.

Nodo DataBase-Meta-S-MS: nodo encargado de proporcionar la información requerida por el DAO de seguridad.

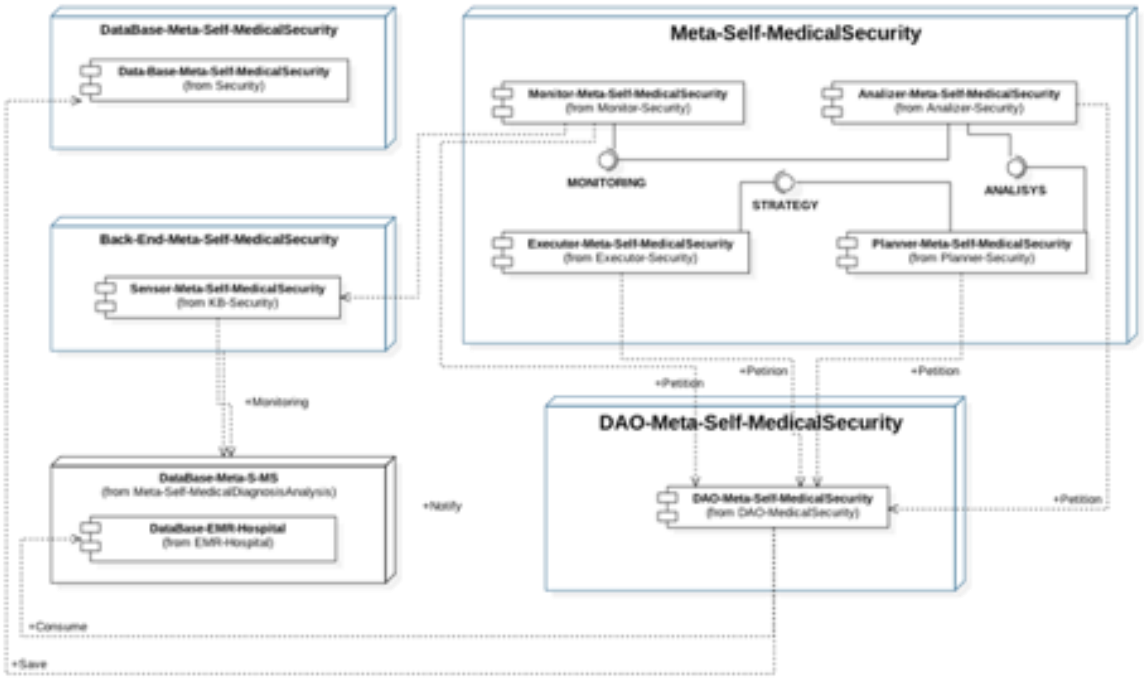

Figura 32. Diagrama de implantación del Meta-S-MS.

4.3.3 Diseño del Meta-Self-MedicalHealing

Mediante el análisis del estado del arte, los objetivos planteados en la Seccion 4.2 y la definición del M-S-MH en la Sección 4.3 se determina que los sistemas de información para el cuidado de la salud son deficientes en mantenibilidad y modificabilidad, esto debido al poco personal calificado enfocado a proporcionar mecanismos por los cuales los sistemas HCIS puedan ser modificados de manera adecuada, sumado a esto encontramos la falta de arquitecturas y de poca documentación sobre los sistemas HCIS, que ayuden a modificar y a reparar los sistemas.

Tomando en cuenta lo anterior, se decidió diseñar un componente autonómico enfocado a la auto-curación o auto-reparación de los sistemas de información para el cuidado de la salud. Siguiendo la metodología presentada en el Capítulo 3 se realizará el diseño del Meta-Self-MedicalHealing enfocado a cubrir el campo de mantenibilidad más específicamente la auto-reparación de sistemas de apoyo al diagnóstico médico y sus bases de datos.

Este componente se ha diseñado mediante el desarrollo de una serie de vistas, las cuales en términos generales abstraen conceptos comunes de varias arquitecturas y las elevan a un nivel superior.

4.3.3.1 Atributos de calidad del Meta-S-MH

La selección de los atributos de calidad para Meta-S-MH se realizó tomando tres objetivos: 1) el análisis del estado del arte, 2) definición del meta-componente, 3) definición del atributo de calidad. La *Tabla 27* muestra la relación de los atributos de calidad y el meta-componente.

Tabla 27. Meta-S-MH a atributos de calidad

Meta-Componente	**Atributo de calidad**	**Sub-categoría (s)**	**Justificación**	
Meta-Self-MedicalHealing	Disponibilidad	Confiabilidad	Tolerancia a fallos	El objetivo del Meta-Self-MH es detectar, diagnosticar y reparar automáticamente el mal funcionamiento por lo que si un componente falla lo tiene que reparar, pero si el Meta-Self-MH falla no se puede auto-reparar solo, por lo que necesita implementar el atributo de calidad de tolerancia a fallas mediante la implementación de un patrón o táctica.
			Recuperabilidad	El Meta-Self-MH debe implementar recuperabilidad ya que la información de los

				pacientes es muy importante, por ejemplo, supongamos que la base de datos donde se encuentran los registros médicos de los pacientes se ha dañado y se ha perdido gran cantidad de información, esto sería un gran problema ya que el hospital podría enfrentarse a demandas por parte de los pacientes, por lo que es importante que tenga un método de recuperación de información.
	Mantenibilidad	Modificabilidad	Riesgo de daño	El Meta-Self-MH que es un componente enfocado a reparar fallas en el sistema es importante que implemente mantenibilidad más específicamente en el contexto de riesgo de daño ya que sin un componente llegase a fallar ya se tendría contemplada esa falla y se podría reparar más fácilmente o dar mantenimiento antes de que fallase.
	Usabilidad	Manejo de errores	Es importante considerar que el Meta-Self-MH implemente el atributo de calidad de usabilidad más específicamente la sub-categoría de manejo de errores, ya que los usuarios podrían generar errores bajo ciertas circunstancias de uso del sistema	

4.3.3.2 Vista de escenarios: Meta-S-MH

El diseño del Meta-S-MH comienza por realizar el levantamiento de requerimientos asociados a la mantenibilidad y modificabilidad de los sistemas de información de apoyo al diagnóstico médico, más específicamente se busca delimitar los objetivos, necesidades, características y atributos de calidad. Definir los casos de uso proporciona una guía para obtener las principales características del Meta-S-MH enfocado a los sistemas HCIS.

Diagrama de casos de uso

De la definición en la Sección 4.3.1, el estado del arte y los atributos de calidad, se han obtenido los casos de uso asociados al componente Meta-S-MH. La Figura 33 ilustra los dos casos de uso primarios asociados al componente, esta muestra las relaciones de dependencia entre los actores (Timer) y los casos de uso.

Los casos de uso mostrados en la Figura 33 están enfocados a la auto-curación de los sistemas médicos. A continuación, se muestra la descripción detallada de los casos de uso.

1. Reparar la base de datos: este caso de uso se enfoca a reparar la base de datos, para explicar mejor el funcionamiento se plantea el siguiente ejemplo. En un momento de ejecución un usuario que tiene acceso a la base de datos ha ingresado a esta y la ha modificado algunos datos de pacientes, el usuario ha realizado algunas pruebas para asegurarse que la base de datos sigue funcionando, la base de datos ha respondido y el usuario ha guardado los cambios, cuando la base de datos ha sido puesta en ejecución, se ha detectado que no realiza las peticiones indicadas por los usuarios, el sistema detecta esta situación, compara las bases de datos actual con la versión anterior y corrige el error.

2. Reparar el servidor que contiene al HCIS: este caso de uso está enfocado en diagnosticar y reparar el servidor que contiene al sistema HCIS. En un momento determinado el sistema HCIS ha dejado de funcionar, algunas de las posibles fallas que se tienen son: usar un hosting compartido, picos de tráfico, sobrecarga de recursos, fallas de seguridad y no usar sistemas de cache. Para explicar mejor el funcionamiento se plantea el ejemplo siguiente: en un momento en tiempo de ejecución se han realizado muchas peticiones al servidor y se ha caído, el sistema detecta esta situación y selecciona las herramientas necesarias para solucionar este problema.

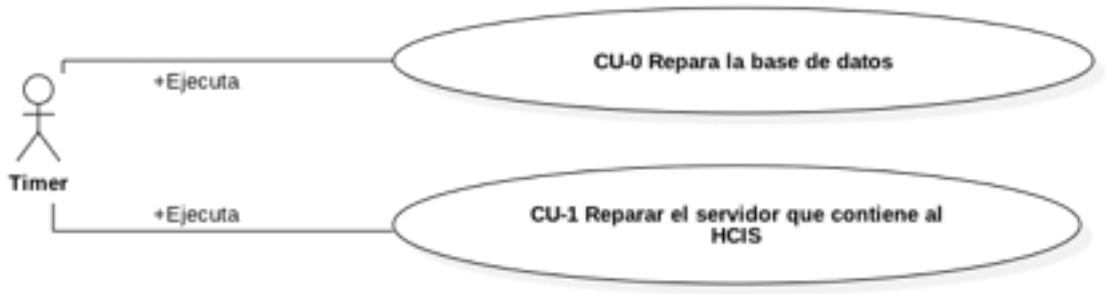

Figura 33. Diagrama de casos de uso.

El diagrama de casos de uso muestra los dos casos de uso más relevantes en el diseño del Meta-S-MH. Como se muestra en la Figura 33, el timer representa al actor y es el encargado de ejecutar los casos de uso en un momento determinado, el timer mantiene una relación de asociación con los casos de uso indicados con una línea y representado con el estereotipo <+ejecuta>. Estos dos casos de uso, junto con los atributos de calidad obtenidos del ADD, serán mapeados a escenarios autonómicos, esto con la finalidad de tener un meta-componente que implemente una arquitectura autonómica.

La *Tabla 28*, muestra el escenario autonómico, reparar la base de datos, y describe el funcionamiento de cada uno de los componentes que dan soporte al caso de uso.

Tabla 28. Plantilla SAQAS del Meta-Self-MedicalHealing.

Source	Usuario	
Stimulus	Un usuario con acceso al sistema ha ingresado a la base de datos y la ha dañado. El sistema detecta que la base de datos no responde a las peticiones y la ponen fuera de servicio, el sistema configura la base de datos espejo para que el sistema funcione con normalidad mientras se repara la base de datos dañada, una vez que se ha reparado se configura y se pone en funcionamiento.	
Artifact	EMR	
Environment	Tiempo de ejecución	
Response	Sensor	Envía notificaciones de la base de datos al Monitor
	Monitor	Recibe notificaciones del Sensor y monitorea la base de datos. Envía notificaciones al Analyzer de cualquier cambio en la base de datos
	Analyzer	Analiza los cambios realizados. Recibe las notificaciones enviadas por el Monitor y en base a estas realiza el análisis. En este caso analiza las peticiones del usuario a la base de datos y el por qué no se pueden ejecutar. En base al análisis realizado genera un reporte y lo envía el componente Planner.
	Planner	Se suspende la operación de la base de datos por un periodo de 30 segundos mientras entra el respaldo, Recibe el reporte generado por el Analyzer y selecciona el plan más adecuado. Se compara la base de datos dañada con la de repuesto y se realiza la copia de la información. Se ejecutan las peticiones realizadas por el usuario anteriormente y se activa la base de datos de nuevo.
	Executor	Se ejecutan los planes.
	Knowledge	Guarda los planes de contingencia, consultas y estadísticas generadas por el Analyzer para su posterior utilización.
Self-Healing	Los sistemas de computación autónomos detectan, diagnostican y reparan los problemas localizados producidos por defectos en el software o en el equipo físico, usando conocimientos sobre la configuración del sistema, realizando diagnósticos, analizando la información de archivos de registro, etcétera.	

La Figura 34 muestra la ejecución del escenario autonómico: reparar la base de datos de forma autónoma, como primer paso se observa a un usuario que se ha autenticado al sistema y ha accedido a la base de datos. El usuario ha modificado la base de datos y ha realizado algunas pruebas para verificar el correcto funcionamiento y ha guardado los cambios. En un momento determinado en tiempo de ejecución otro usuario ha realizado una petición a la base de datos, pero esta no ha respondido y se ha generado una excepción. El Sensor que se encuentra en el servidor de base de datos se ha dado cuenta que se ha generado un error y ha notificado al Monitor, con el cual establece una relación bidireccional para la transferencia de información indicada por una flecha con dos puntas. El Monitor lanza el ciclo autonómico, monitorea la base de datos y notifica al Analyzer, este envía pruebas a la base de datos y en base a estas genera un análisis de las posibles fallas que pueda tener el servidor de base de datos, el Analyzer genera un reporte y lo envía al Planner. Este se encarga de seleccionar el plan adecuado para reparar la base de datos, el plan seleccionado es enviado al Executor, esté ejecuta el plan y cierra el ciclo autonómico.

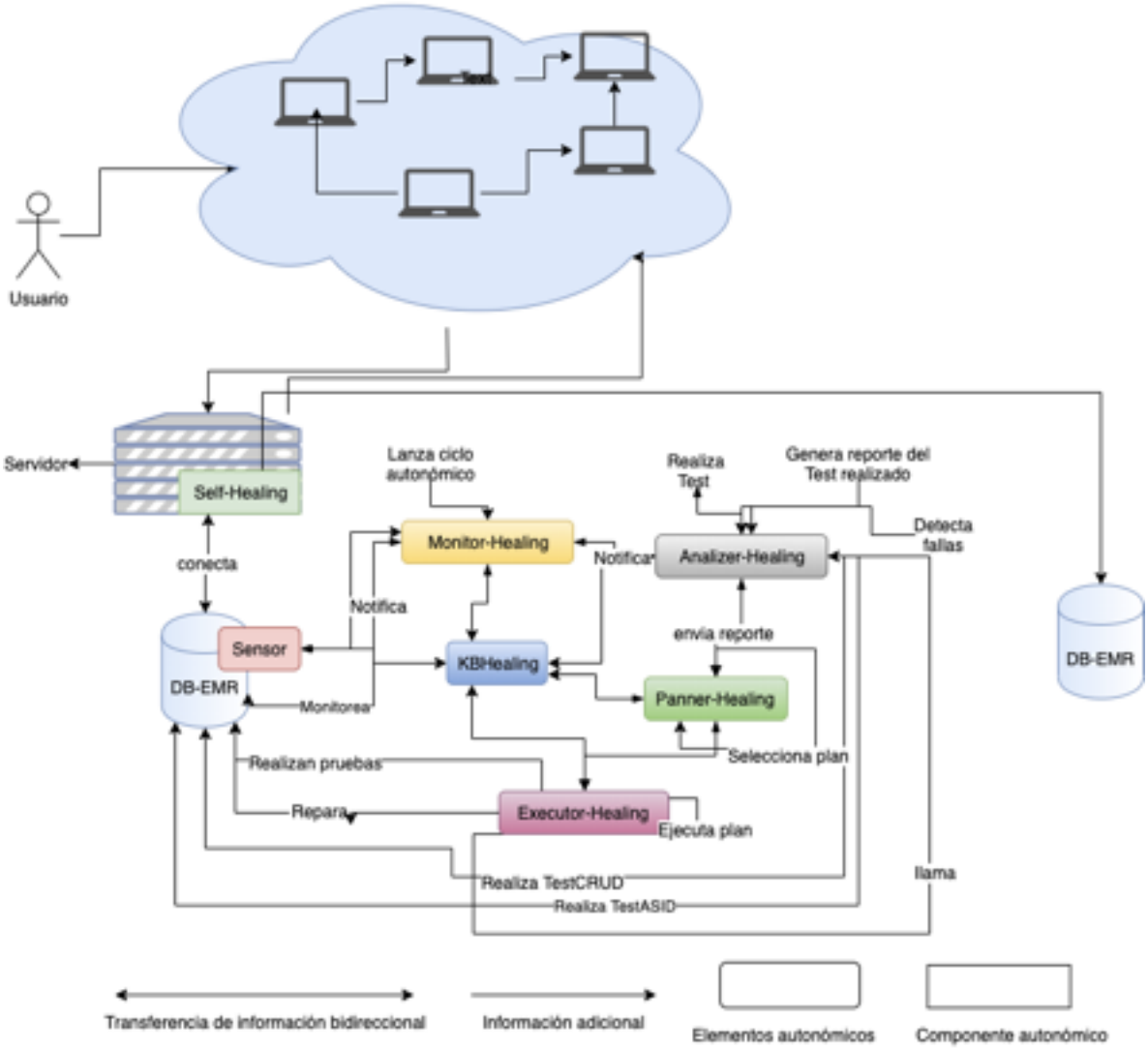

Figura 34. Reparar la base de datos de forma autónoma.

4.3.3.3 Vista lógica: Meta-S-MH

Para el diseño del Meta-S-MH se desarrolla la vista lógica, enfocada principalmente en ver el dominio del problema en forma de clases u objetos. También realizar esta vista nos permite identificar mecanismos y elementos de diseño comunes a los escenarios autonómicos: reparar la base de datos y reparar el servidor que contiene al sistema HCIS.

Diagrama de clases del Meta-S-MH

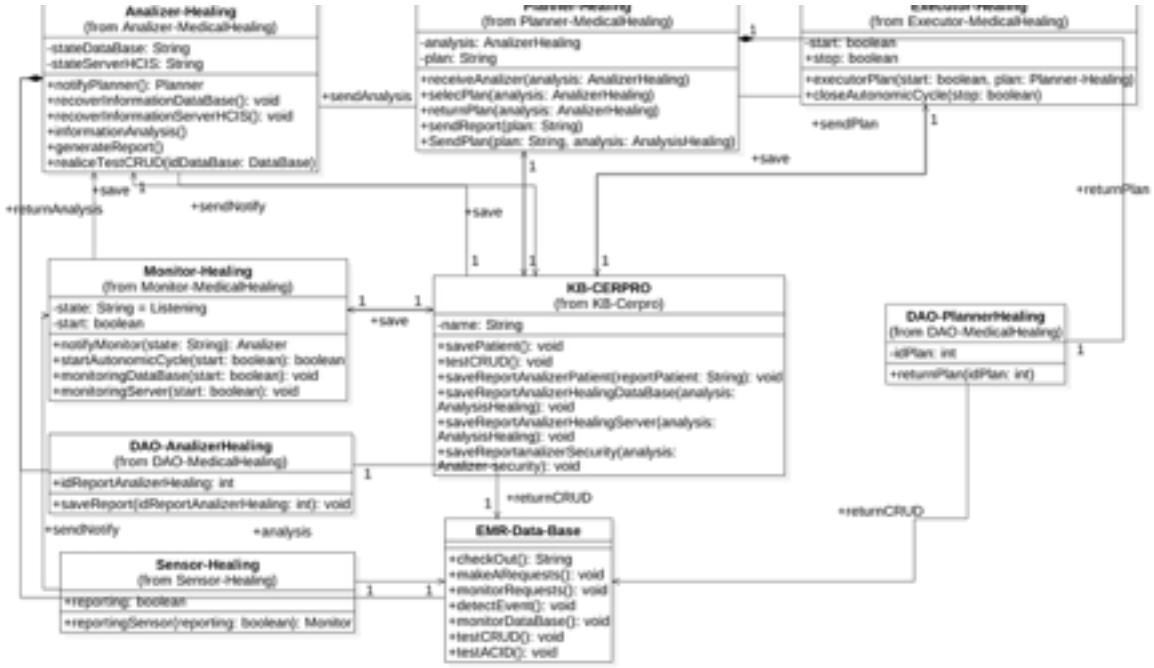

Figura 35. Diagrama de clases del Meta-S-MH.

La Figura 35 representa el diagrama de clases de alto nivel del Meta-S-MH, en esta figura se pueden apreciar las cinco clases pertenecientes al patrón MAPE-K, las cuales dan soporte a los escenarios autonómicos descritos en el Anexo A, a continuación, se describe cada una de las clases que conforman el diagrama.

Class Sensor-Healing: encargado de detectar cualquier evento dentro de la base de datos y de enviar notificaciones al Monitor (ver Apéndice A).

Class Monitor-Healing: encargado de iniciar el ciclo autonómico y de monitorear la base de datos de cualquier error que pueda surgir.

Class Analyzer-Healing: encargado de analizar los cambios en la base de datos, de realizar las pruebas que llevan al diagnóstico del mal funcionamiento de la base de datos, y generar reportes de las posibles fallas encontradas.

Class DAO-Analyzer-Healing: encargado de guardar los reportes que genera el Analyzer y de guardar los test que aplica el Analyzer para verificar el funcionamiento de la base de datos.

Class Planner-Healing: encargado de seleccionar el plan adecuado en base al reporte generado por el Analyzer.

Class DAO-Planner-Healing: encargado de guardar los planes que ocupa el Planner.

Class KB-Cerpro: encargado de generar nuevo conocimiento y proporcionar la inteligencia al sistema autonómico, la base de conocimiento mantiene una relación bidireccional con todos los componentes involucrados en realizar el funcionamiento autonómico.

Class EMR-Data-Base: encargado de proporcionar toda la información relacionada con los pacientes.

Diagrama de secuencia del Meta-S-MH: reparar la base de datos de forma autonómica.

A manera de ejemplo se presenta el diagrama de secuencia asociado con el escenario autonómico: reparar la base de datos de forma autonómica, esto con el objetivo de ver cómo interactúan los elementos en tiempo de ejecución. Recordando que un diagrama de secuencia muestra la interacción de un conjunto de objetos a través del tiempo. La Figura 36 representa un diagrama de secuencia del escenario autonómico asociado al Meta-S-MH. La descripción detallada solo se presenta para un diagrama de secuencia. A continuación, se describe cada una de las interacciones del diagrama de secuencia.

1. En un momento de operación el timer comienza la ejecución.
2. El Sensor ha detectado un evento en la base de datos.
3. El Sensor recibe una notificación de dicho evento.
4. El Sensor notifica al Monitor.
5. El Monitor lanza el ciclo autonómico.
6. El Monitor monitorea la base de datos.
7. Detecta anomalías dentro de la base de datos y recibe una notificación.
8. Envía una notificación al Analyzer.
9. Hace la primera prueba a la base de datos, realiza el test CRUD.
10. Hace la segunda prueba a la base de datos, realiza el test ACID.
11. Recupera la información en base a los test realizados.
12. Realiza un análisis de la información recuperada.
13. Genera un reporte en base a la información.
14. Envía la información al Planner.
15. El Planner realiza una petición al DAO para recuperar los planes.
16. EL DAO realiza una petición a la base de datos para recuperar los planes.
17. La base de datos envía los Planes al DAO.
18. El DAO envía los planes al Planner.
19. El Planner selecciona el plan en base al análisis realizado.
20. El Planner envía el plan al Executor.
21. El Executor ejecuta el plan.
22. El Executor finaliza el ciclo autonómico.

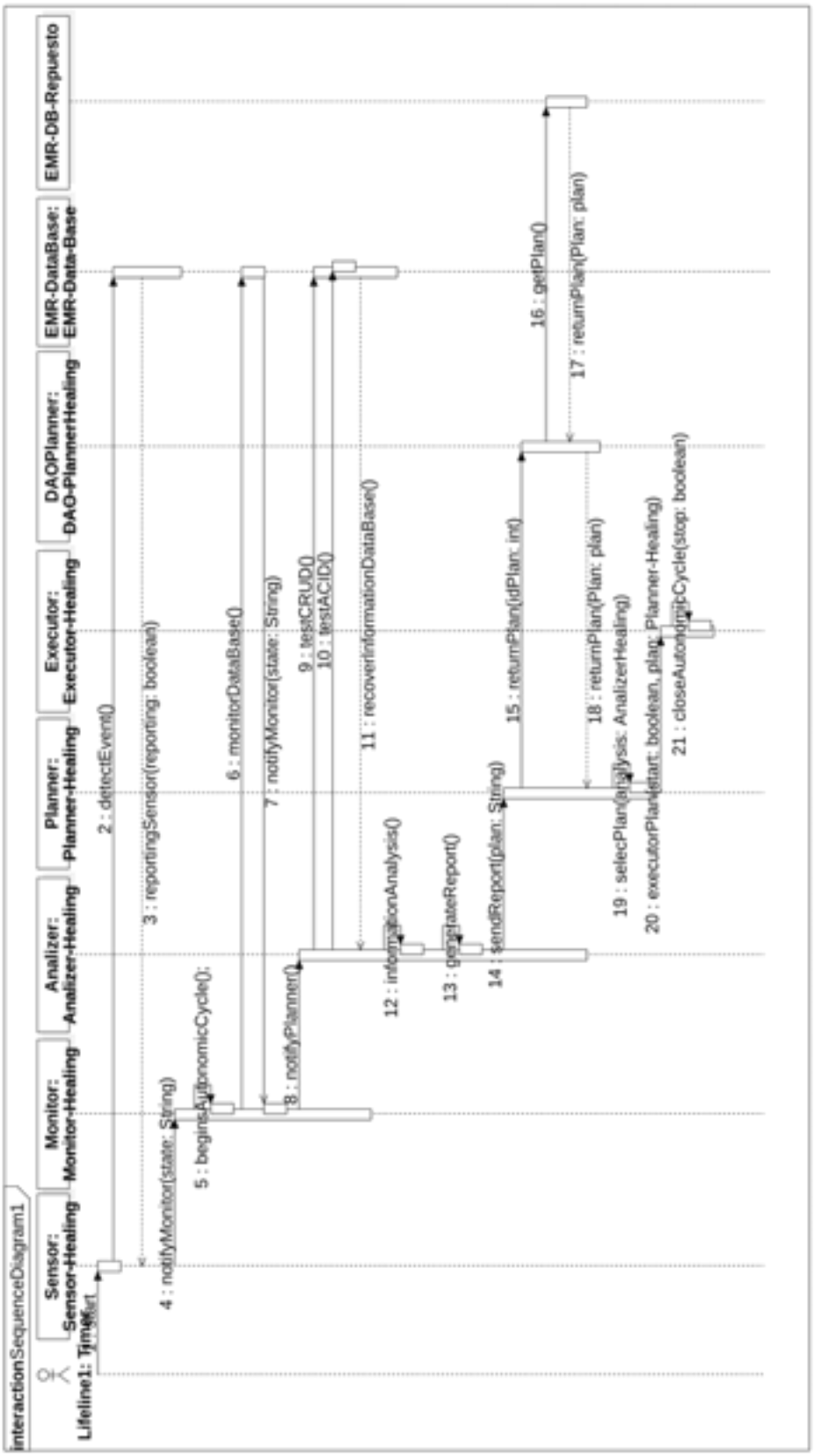

Figura 36. Diagrama de secuencia de un escenario autonómico perteneciente al Meta-S-MH.

Como se muestra en el diagrama de casos de uso de la Sección 4.3.3.2 existen 2 casos de uso primarios, los cuales son transformados en escenarios autonómicos y dan soporte al Meta-S-MH. Es decir, es necesario realizar estos dos escenarios autonómicos, los cuales serán documentados con la plantilla SAQAS.

4.3.3.4 Vista de procesos: Meta-S-MH

Diagrama de actividades: Meta-S-MH

El diagrama de actividades del Meta-S-MH tiene como propósito modelar el comportamiento del sistema, más específicamente en nuestro caso, modelar el escenario autonómico, reparar la base de datos y reparar el servidor que contiene al HCIS. La Figura 37 muestra el diagrama de actividades que pertenece a reparar el servidor que contiene al sistema HCIS. Como entradas del diagrama de actividades tiene un timer que juega el papel de usuario principal, y en un momento de operación normal el servidor que contiene al sistema HCIS se ha caído por sobrecarga de peticiones. El Sensor que se encuentra en el servidor ha detectado que ha ocurrido un cambio en el servidor y envía un mensaje al Monitor. Este al recibir la notificación lanza el ciclo autonómico y monitoriza al servidor por un periodo de 30 segundos y notifica al Analyzer. Este verifica el funcionamiento del servidor y detecta que el problema se ha suscitado dado a que se han realizado más peticiones al servidor de las que podía soportar, el Analyzer genera un reporte y lo envía al Planner. Este envía el plan al Executor el cual ejecuta y cierra el ciclo autonómico.

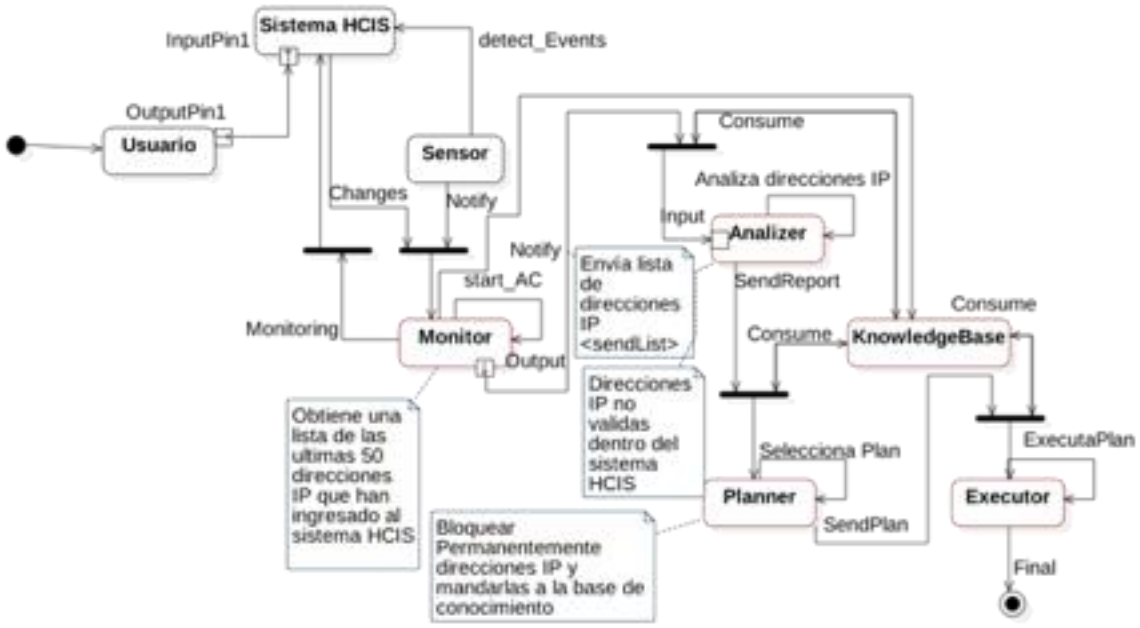

Figura 37. Diagrama de actividades del Meta-S-MH

4.3.3.5 Vista de despliegue: Meta-S-MH

La vista de despliegue del Meta-S-MS al igual que los demás meta-componentes está centrada en una arquitectura en capas en las cuales se implementa el patrón de diseño MAPE-K. Aplicando el dicho de divide y vencerás, el diseño de la arquitectura se representa en partes pequeñas que pueden ser desarrollados por una o más personas.

Diagrama de paquetes del Meta-S-MH

El objetivo de crear un diagrama de paquetes de alto nivel del Meta-Self-MS es tener una visión más clara de todos los elementos que componen la arquitectura, organizando la arquitectura en capas, agrupando los elementos autonómicos y detallando las relaciones de dependencia entre ellos.

Para el diseño del Meta-S-MH se establece una arquitectura en capas la cual fomenta el principio de diseño de alta cohesión y el bajo acoplamiento. La primera capa está encargada de soportar las reglas de negocio, es decir, se encarga de la funcionalidad del sistema, llevando a cabo los escenarios autonómicos propuestos en la Sección 4.3.3.2.

La capa de negocio del Meta-S-MH contiene cuatro componentes, cada uno de estos componentes se corresponde con cada una de las clases necesarias para implementar el ciclo autonómico enfocado a la auto-curación de los sistemas HCIS de apoyo al diagnóstico médico. Las clases implementan los métodos necesarios para la detección de errores y la recuperación del sistema a partir de realizar copias de seguridad.

La capa Data-Layer muestra el paquete donde se encuentran los DAO's encargados de realizar la persistencia de la información. Como se muestra en la Figura 38, la comunicación con la capa de negocio se establece por medio de una interfaz bien definida y la comunicación con las bases de datos se establece por medio de una dependencia simple que es indicada por una flecha punteada.

La capa BackEnd-Layer corresponde al paquete donde se encuentra el Sensor, encargado de enviar notificaciones a la capa de negocio de cualquier evento que suceda en la base de datos, más específicamente envía notificaciones al Monitor.

La capa Data Base corresponde a la base de conocimiento encargada de proporcionar el aprendizaje al componente autonómico y la base de datos del hospital encargados de proporcionar la información de los pacientes a la capa de negocio.

4.3.3.6 Vista física del Meta-S-MH

Tomando en cuenta los atributos de calidad asociados al Meta-Self-MH, tales como la disponibilidad, mantenibilidad y usabilidad, se muestra el diagrama de implantación con el objetivo de satisfacer dichos atributos.

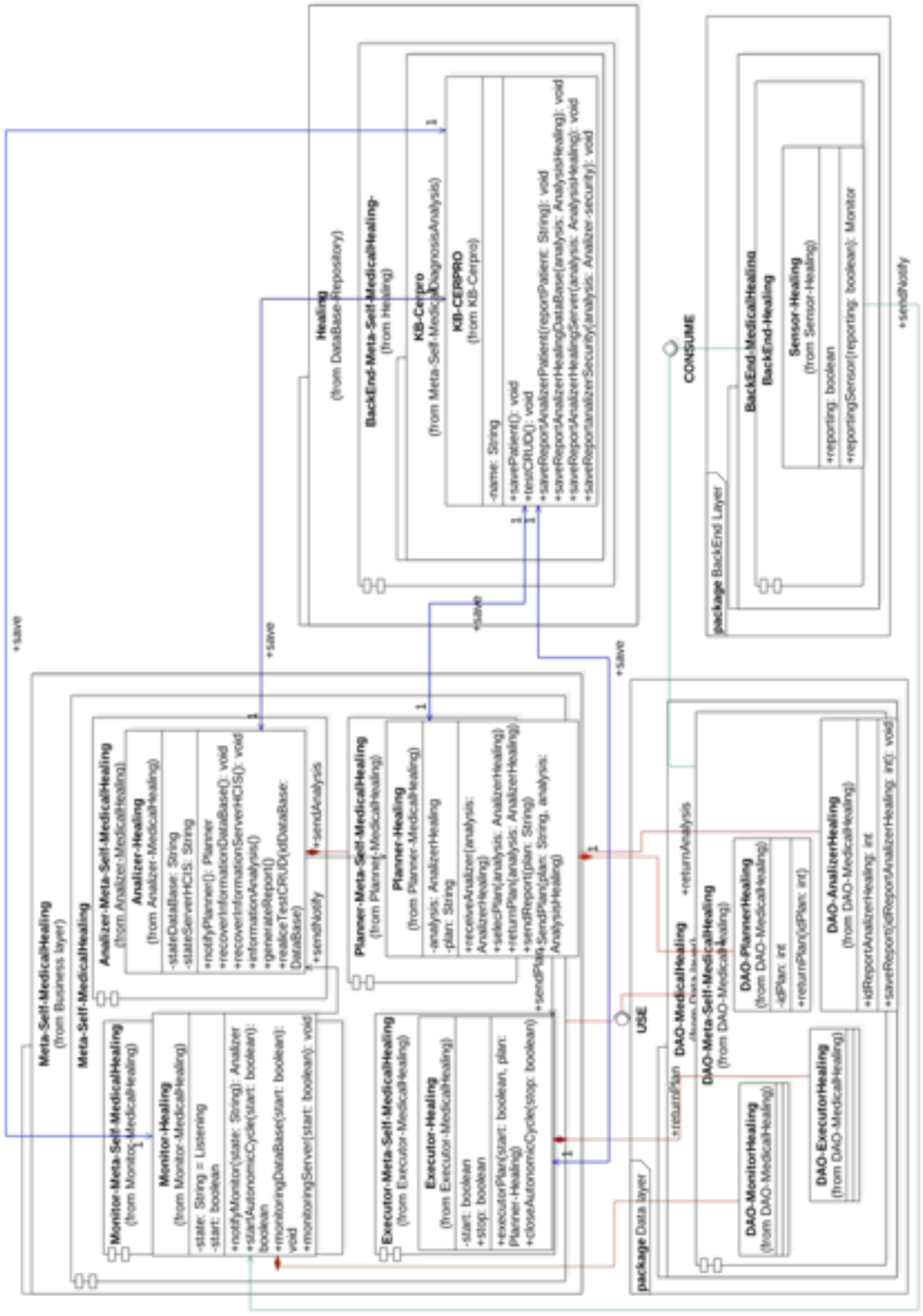

Figura 38. Diagrama de paquetes del Meta-S-MH.

Diagrama de implantación del Meta-S-MH

El diagrama de implantación del Meta-Self-MH permite visualizar la arquitectura física del hardware, software y todos los artefactos que se implementan dentro de la arquitectura como los patrones. El diagrama del Meta-S-MH ilustra la forma física del sistema, en lugar de representar conceptualmente los usuarios y dispositivos que interactúan dentro de la arquitectura.

La Figura 39 representa el diagrama de implantación del Meta-Self-MH, el cual contiene 9 nodos, recordando que un nodo puede contener objetos, instancias y componentes. Además, permite establecer relaciones entre nodos mediante un canal de comunicación como se muestra en la *Figura 39*. A continuación, se describe el funcionamiento de cada uno de los nodos:

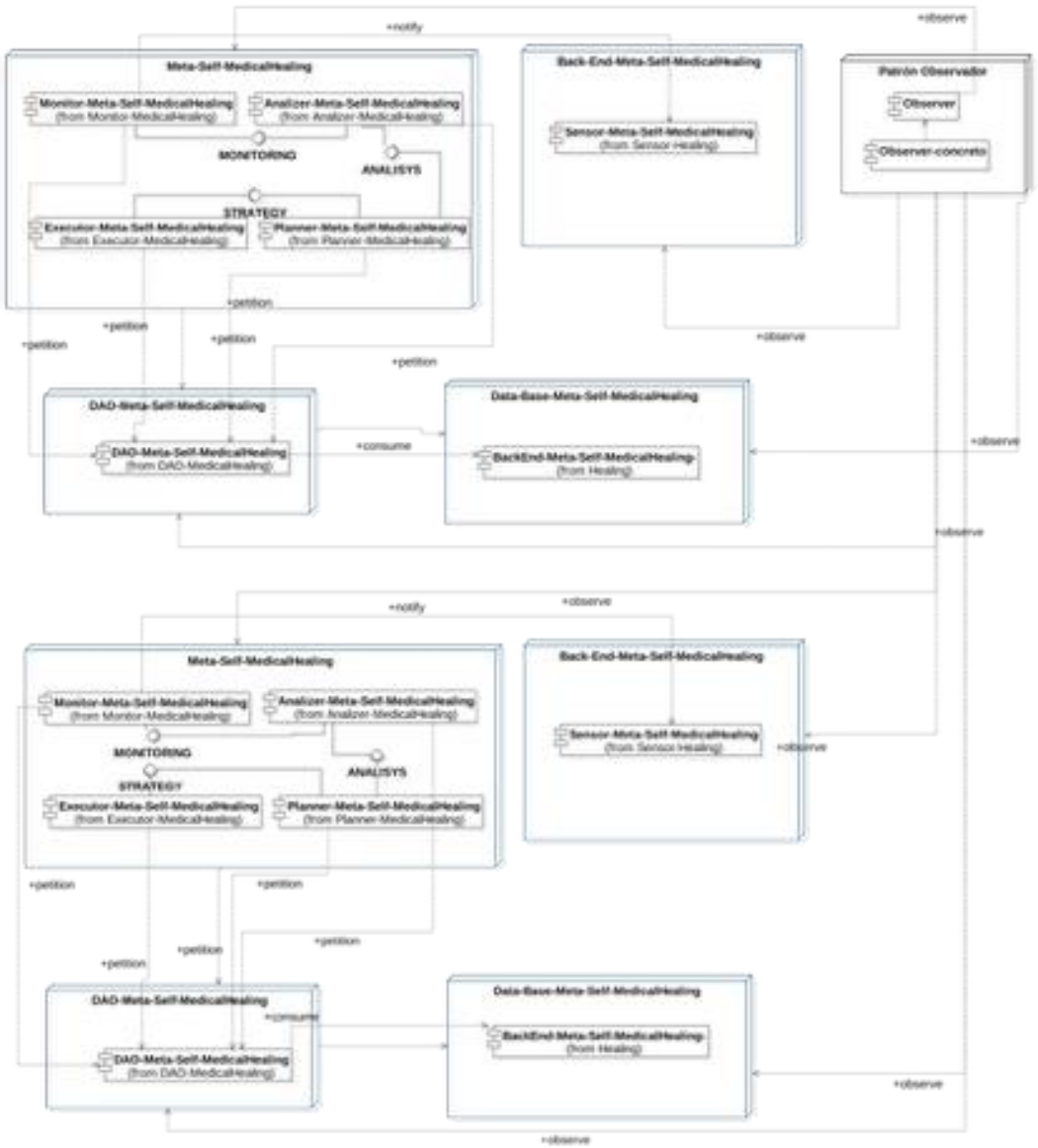

Figura 39. Diagrama de implantación del Meta-S-MH.

Nodo-Meta-Self-MedicalHealing: nodo encargado de comenzar y finalizar el ciclo autonómico, dentro de este nodo se encuentran cuatro componentes encargados de reparar o curar las fallas que se encuentren en el sistema HCIS de apoyo al diagnóstico médico como son: falla en la base de datos y en el servidor que contiene al HCIS.

Nodo DAO-Meta-Self-MedicalHealing: nodo encargado de atender todas las peticiones de cada uno de los componentes encontrados dentro del nodo Meta-Self-MedicalHealing.

Nodo BackEnd-Meta-Self-MedicalHealing: nodo encargado de detectar eventos en la base de datos y en el servidor que contiene el sistema HCIS, cuando detecta un evento envía una notificación al Monitor.

Nodo DataBase-Meta-Self-MedicalHealing: encargado de proporcionar la información requerida por el DAO-Healing.

Nodo Patrón observador: nodo encargado de monitorear el funcionamiento de cada uno de los nodos pertenecientes al meta-componente, en caso de que uno de los nodos falle el observador se encargará de notificar y despertar al nodo que se encuentra en standby.

Nodo Fuera de servicio: este nodo nos permite poner a un nodo fuera de servicio temporalmente mientras es diagnosticado y reparado por el ciclo autonómico.

4.3.4 Diseño del Meta-Self-MedicalConfiguration

Mediante el análisis del estado enfocado a los sistemas HCIS de apoyo al diagnóstico médico y a los objetivos planteados en la Sección 4.2 encontramos los principales problemas que afrontan estos sistemas, encontrando que son deficientes en términos mantenibilidad. Más específicamente se encontró que no existen arquitecturas o modelos de arquitecturas que proporcionen los elementos suficientes para realizar configuraciones y modificaciones a los sistemas médicos, siendo este un problema, ya que los sistemas médicos deben de buscar actualizarse constantemente debido al alto valor de información que manejan. Es por estas razones que se decidió diseñar un meta-componente autonómico enfocado a configurar los sistemas médicos. El diseño del Meta-S-MC se realizará siguiendo el método de diseño presentado en el Capítulo 3.

Este componente se ha diseñado mediante el desarrollo de una serie de vistas, las cuales en términos generales abstraen conceptos comunes de varias arquitecturas y las elevan a un nivel superior.

4.3.4.1 Atributos de calidad del Meta-S-MC

La selección de los atributos de calidad para Meta-S-MC se realizó tomando tres objetivos: 1) el análisis del estado del arte, 2) definición del meta-componente, 3) definición del atributo de calidad. La *Tabla 29* muestra la relación de los atributos de calidad y el meta-componente.

Tabla 29. Meta-S-MC a atributos de calidad

Meta-Componente	Atributo de calidad	Sub-categoría (s)	Justificación
Meta-Self-MedicalConfiguration	Usabilidad	Eficiencia	El Meta-Self-MC implementa una sub-categoría de usabilidad la cual es eficiencia. La razón de implementar este atributo de calidad es brindar al usuario final un mejor sistema. Por ejemplo, supongamos que la base de datos donde se encuentran los registros médicos de los pacientes ha sufrido un error, por lo que debe de configurarse la base de datos que se encuentra en standby. Esta deberá configurarse en el menor número de transacciones y el menor tiempo posible para que no se vea afectado el sistema.
	Mantenibilidad	Modificabilidad	El objetivo Meta-Self-MC es hacer cambios en el ambiente, por lo que si se desea realizar un cambio tanto en arquitectura como en el sistema tiene que ser fácil de realizar y en el menor tiempo posible, por lo tanto, es importante que el Meta-Self-MC implemente el atributo de calidad de mantenibilidad y más específicamente que pueda implementar modificabilidad para que a la hora de realizar cambios no se introduzcan errores ni degradación del sistema o arquitectura.

4.3.4.2 Vista de escenarios: Meta-S-MC

El diseño de la arquitectura del Meta-S-MC comienza por levantar requerimientos asociados a la configuración y modificación de los sistemas de salud, más específicamente se buscan los objetivos particulares, las necesidades, características y atributos de calidad.

Diagrama de casos de uso

De la definición en la Sección 4.3.1, el estado del arte y los atributos de calidad, se han obtenido los casos de uso asociados al componente Meta-S-MC. La Figura 40 ilustra las relaciones de dependencia entre los actores (timer), y los casos de uso.

Los casos de uso mostrados en la Figura 40 están enfocados a la seguridad de los sistemas médicos. A continuación, se muestra la descripción detallada de los casos de uso.

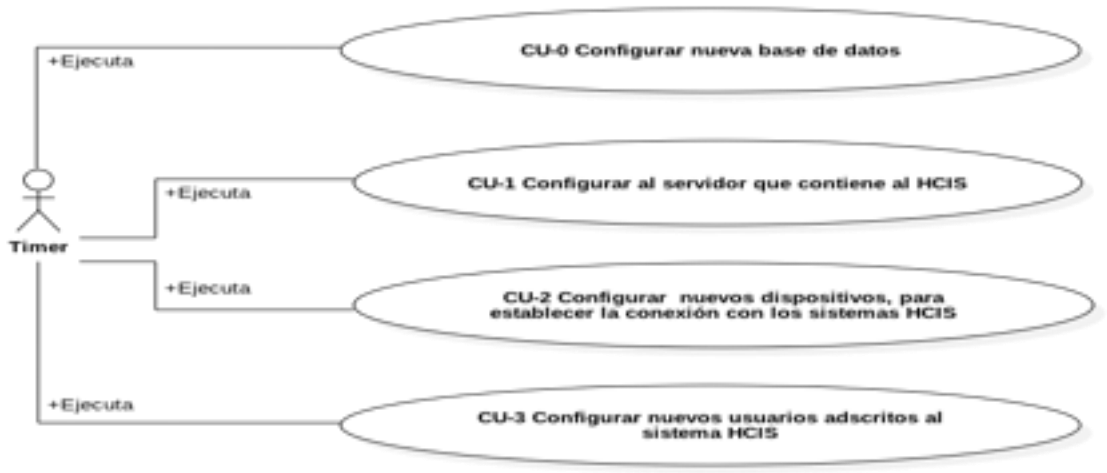

Figura 40. Diagrama de casos de uso del Meta-S-MC.

1. Configurar una nueva base de datos: este caso de uso está enfocado en configurar una base de datos que se encuentre en standby. Para explicar mejor este caso de uso planteamos el siguiente escenario: en un momento de operación el sistema HCIS está funcionando con normalidad, un usuario adscrito al sistema ha realizado una consulta a la base de datos, el usuario ha esperado un tiempo de aproximadamente de 1 minuto y el sistema ha arrojado un error. El Meta-Self-MedicalHealing ha detectado esta situación y ha puesto en standby la base de datos que se encuentra dañada, al mismo tiempo ha enviado un mensaje a Meta-S-MC para que active la base de datos que se encuentra en standby. El Meta-S-MC configura la nueva base de datos y la activa en un tiempo menor a 3 minutos.

2. Configurar el servidor que contiene al sistema HCIS: este caso de uso está enfocado a configurar un nuevo servidor en caso de que se presente una falla en el servidor principal. Para explicar mejor este caso de uso planteamos el siguiente ejemplo: en un momento de operación el servidor que contiene el sistema HCIS de apoyo al diagnóstico médico ha dejado de funcionar. El Sensor que se encuentra en el servidor ha detectado un nuevo evento y ha notificado al Monitor, este ha verificado el funcionamiento del servidor y ha comprobado que ha dejado de funcionar por lo que debería poner en ejecución el servidor que se encuentra en standby, se ha despertado al servidor y se han realizado las configuraciones necesarias para ponerlo en ejecución.

3. Configurar nuevos dispositivos: para establecer comunicación con los sistemas HCIS, este caso de uso está enfocado a configurar nuevos dispositivos que se quieren comunicar con el sistema HCIS, para esto el sistema verifica que los dispositivos que se desee comunicar intercambien información bajo un estándar médico como el HL7 (Health Level Seven) [36].

4. Configurar nuevos usuarios adscritos al sistema HCIS, este caso de uso está enfocado a realizar la configuración de nuevos usuarios dentro del sistema médico, como son: invitados (es aquel que llega a la página pero no se identifica como usuario registrado), miembros de la comunidad (se identifican como tales mediante un nombre y una

contraseña en la página de inicio, y que pueden añadir noticias, nuevos enlaces a páginas, ver y subir fotos, leer los foros y participar en ellos, acceder a documentos almacenados), usuarios registrados (son aquellas personas que han creado usuarios en el sitio web y podrán acceder a contenidos especiales. Adicionalmente tienen la posibilidad de colaborar con nuevos documentos a la biblioteca virtual, eventos al calendario de actividades y personas o instituciones al directorio), y administradores (pueden añadir módulos con nuevas funciones, actualizar el portal, dar de alta o baja a usuarios).

Se ha realizado el levantamiento de requerimientos basado en atributos de calidad, estado del arte y la definición del Meta-S-MC, pero para tener un sistema autonómico con la mínima intervención humana los requerimientos deberán ser planteados como escenarios autonómicos. Por lo anterior, los casos de uso se mapean a escenarios autonómicos, los cuales podrán ser documentados utilizando la plantilla SAQAS.

La *Tabla 30*, muestra el escenario autonómico: configurar una nueva base de datos, y describe el funcionamiento de cada uno de los componentes que dan soporte al caso de uso.

Tabla 30. Plantilla SAQAS: Escenario-configurar una nueva base de datos

Source	Usuario	
Stimulus	En un momento de operación se han realizado varias consultas a la base de datos, esta no ha respondido en el tiempo esperado y se ha caído, el sistema ha detectado que la base de datos ha fallado y ha notificado a la base de datos espejo para que se configure y se ponga en funcionamiento.	
Artifact	EMR, Servidor.	
Environment	Tiempo de ejecución	
Response	Sensor	Detecta cambios en el ambiente y envía notificaciones al Monitor cuando a sucedido un cambio.
	Monitor	Cuando recibe notificación del Sensor lanza el ciclo autonómico. Monitorea a la base de datos del EMR y al servidor que contiene al software del HCIS.
	Analyzer	Caso 1: Recibe notificaciones del componente Monitor y analiza la situación detectada por el Monitor. En el caso de fallo de la base de datos, el Analyzer realiza peticiones para cerciorarse del fallo de la base de datos, y en base al análisis realizado genera un reporte que envía al Planner.
	Planner	Recibe el reporte del componente Analyzer y en base a este selecciona el plan más adecuado. Se configura la base de datos que se encuentra en suspensión y se reactivan los servicios.
	Executor	Ejecuta el plan.
	Knowledge	

		Guarda los planes del Planner y las estadísticas generadas por el Analyzer para su posterior aprendizaje.

La Figura 41 muestra la ejecución del escenario autonómico: configurar una nueva base de datos, de forma descriptiva se observa a un usuario que se encuentra dentro del sistema HCIS. Este usuario ha realizado una petición a la base de datos para obtener la información de un paciente, pero ha ocurrido un error y se le ha envía un mensaje de error. El Sensor que se encuentra en la base de datos ha detectado un nuevo evento y ha enviado un mensaje al Monitor, este se encarga de lanzar el ciclo autonómico y monitorea la base de datos, después de cierto tiempo (30 segundos) se da cuenta que no recibe las peticiones de los usuarios y notifica al Analyzer. Este realiza pruebas de funcionamiento y en base a estas pruebas genera un reporte que envía al Planner, este recibe el reporte enviado por el Analyzer y se da cuenta que tiene que poner en funcionamiento la base de datos que se encuentra en standby, por lo que selecciona el plan y los componentes necesarios para poner en ejecución la base de datos. Una vez que ha seleccionado el plan, lo envía al Executor, el cual será el encargado de configurar la nueva base de datos en base a las especificaciones seleccionadas dentro del plan. Una vez que ha terminado de configurar la nueva base de datos, el Executor finaliza la ejecución del ciclo autonómico.

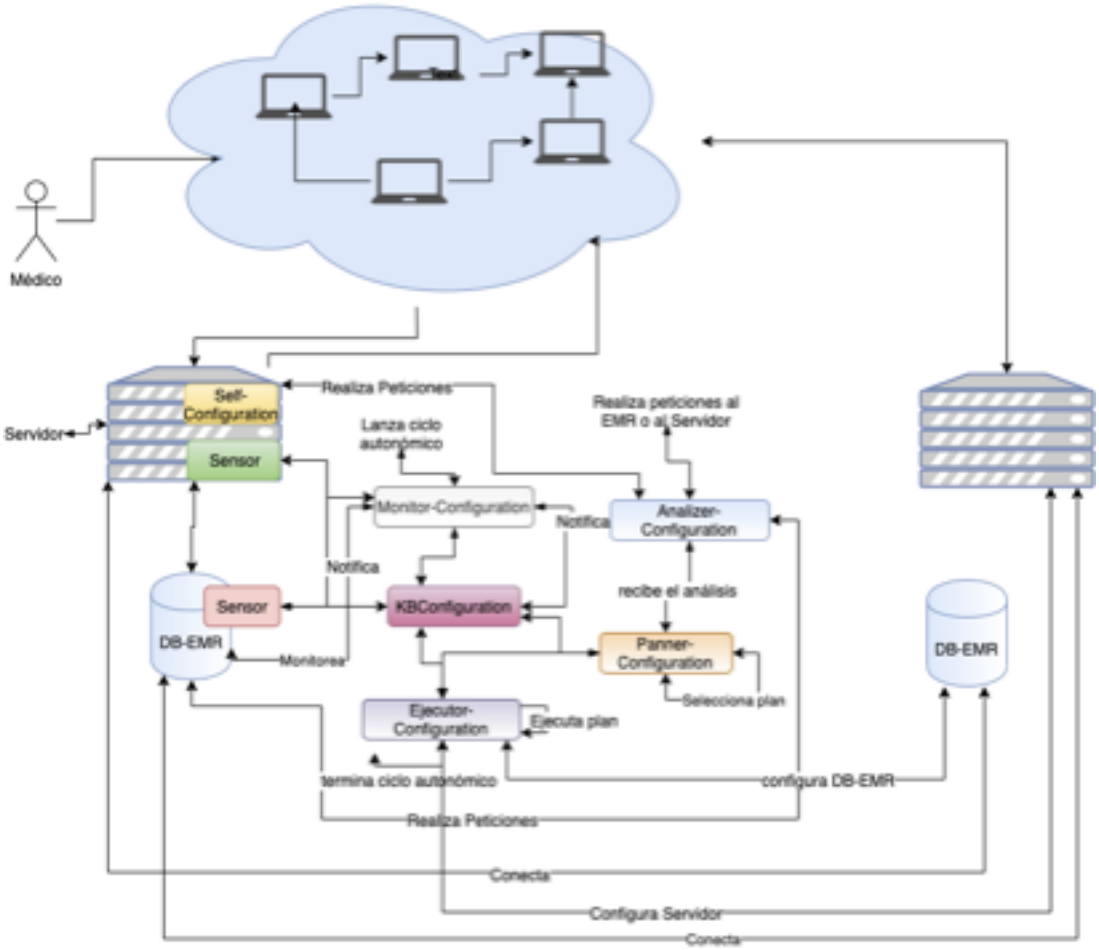

Figura 41. Diagrama configurar una nueva base de datos.

4.3.4.3 Vista lógica: Meta-S-MC

Para complementar el diseño de arquitectura del componente Meta-S-MC se desarrolla la vista lógica, esta se enfoca en ver el dominio del problema en forma de clase, principalmente se identifican mecanismos y elementos de diseño.

Diagrama de clases del Meta-S-MC

El diagrama de clases del Meta-S-MC ilustrado en la Figura 42, muestra todas las clases necesarias para implantar los escenarios autonómicos.

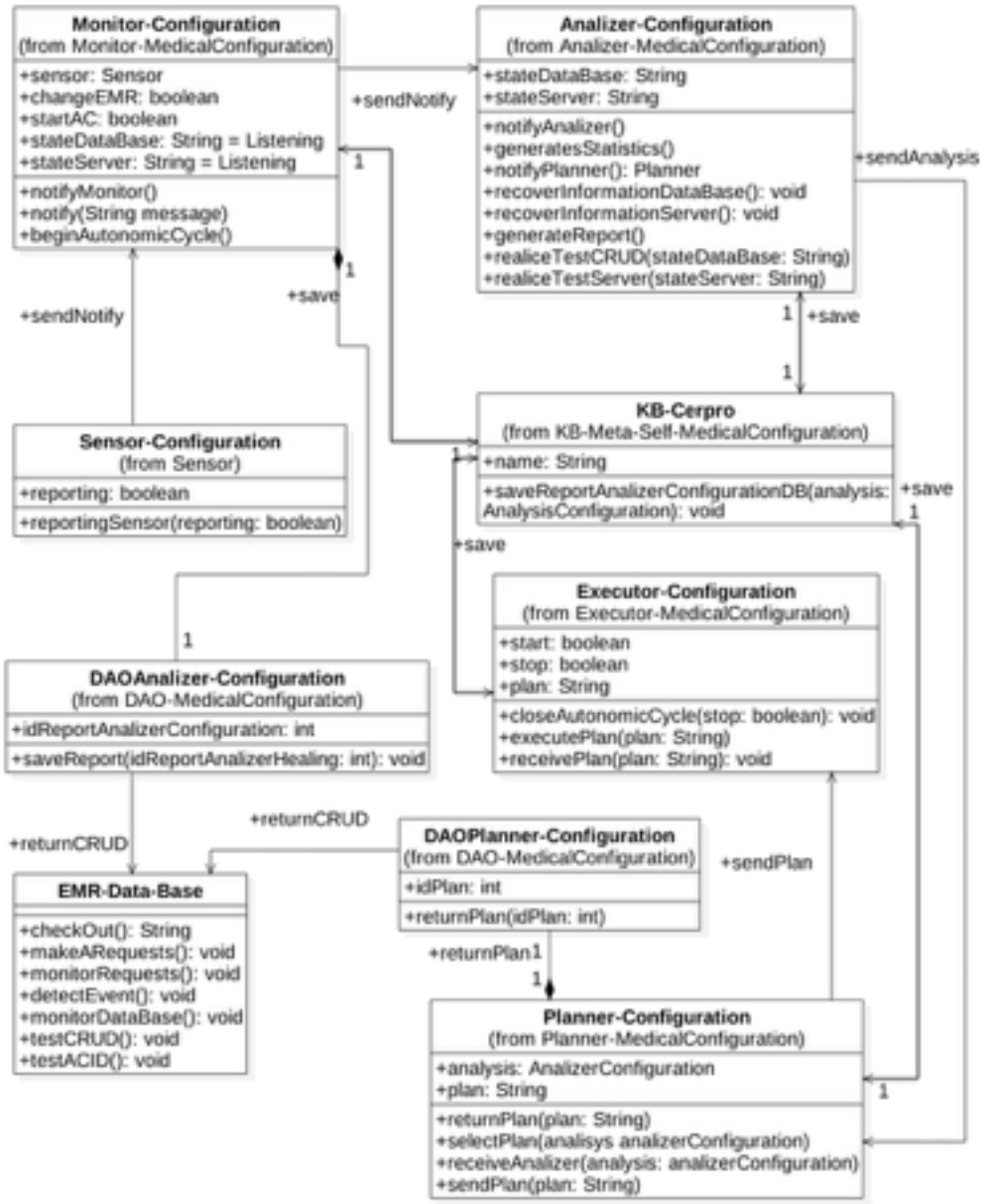

Figura 42. Diagrama de clase de alto nivel del Meta-Self-MedicalConfiguration.

Class Sensor-Configuration: encargado de detectar cualquier evento dentro de la base de datos y del servidor que contiene al sistema HCIS, envía notificaciones al Monitor (Ver Apéndice A).

Class Monitor-Configuration: encargado de recibir las notificaciones enviadas por el Sensor, monitorear la base de datos y el servidor que contiene al HCIS de posibles fallas, si detecta alguna falla envía una notificación al Analyzer.

Class Analyze-Configuration: encargado de recibir las notificaciones enviadas por el Monitor, y analizar el funcionamiento de la base de datos del sistema HCIS y del servidor que contiene al sistema de posibles fallas. Una vez terminado el análisis genera un reporte el cual es enviado al Planner.

Class DAOAnalyze-Configuration: encargado de establecer la comunicación entre el Analyzer y la base de datos, también permite guardar los reportes generados por el Analyzer.

Class Planner-Configuration: encargado de recibir el reporte generado por el Analyzer, en base al reporte selecciona el plan adecuado y lo envía al Executor.

Class DAOPlanner-Configuration: encargado de establecer la comunicación entre el Planner y la base de datos, es decir, es el medio por el cual el Planner obtiene los planes a ejecutar, el DAO y el Planner establecen una relación de agregación y hacen uso del estereotipo <returnPlan>.

Class Executor-Configuration: encargado de ejecutar los planes recibidos del Planner y de finalizar el ciclo autonómico una vez que se ha configurado la base de datos, el servidor o un nuevo usuario.

Class EMR-Data-Base: contiene la información de los registros médicos de los pacientes en el hospital.

Class KB-CERPRO: encargado de proporcionar la inteligencia del sistema autonómico.

Diagrama de secuencia del Meta-S-MC: reparar la base de datos de forma autonómica.

A manera de ejemplo se presenta el diagrama de secuencia asociado con el escenario autonómico: configurar una nueva base de datos, esto con el objetivo de ver cómo interactúan los elementos en tiempo de ejecución. La Figura 43 representa un diagrama de secuencia del escenario autonómico asociado al Meta-S-MC. La descripción detallada solo se presenta para un diagrama de secuencia. A continuación, se describe cada una de las interacciones en dicho diagrama.

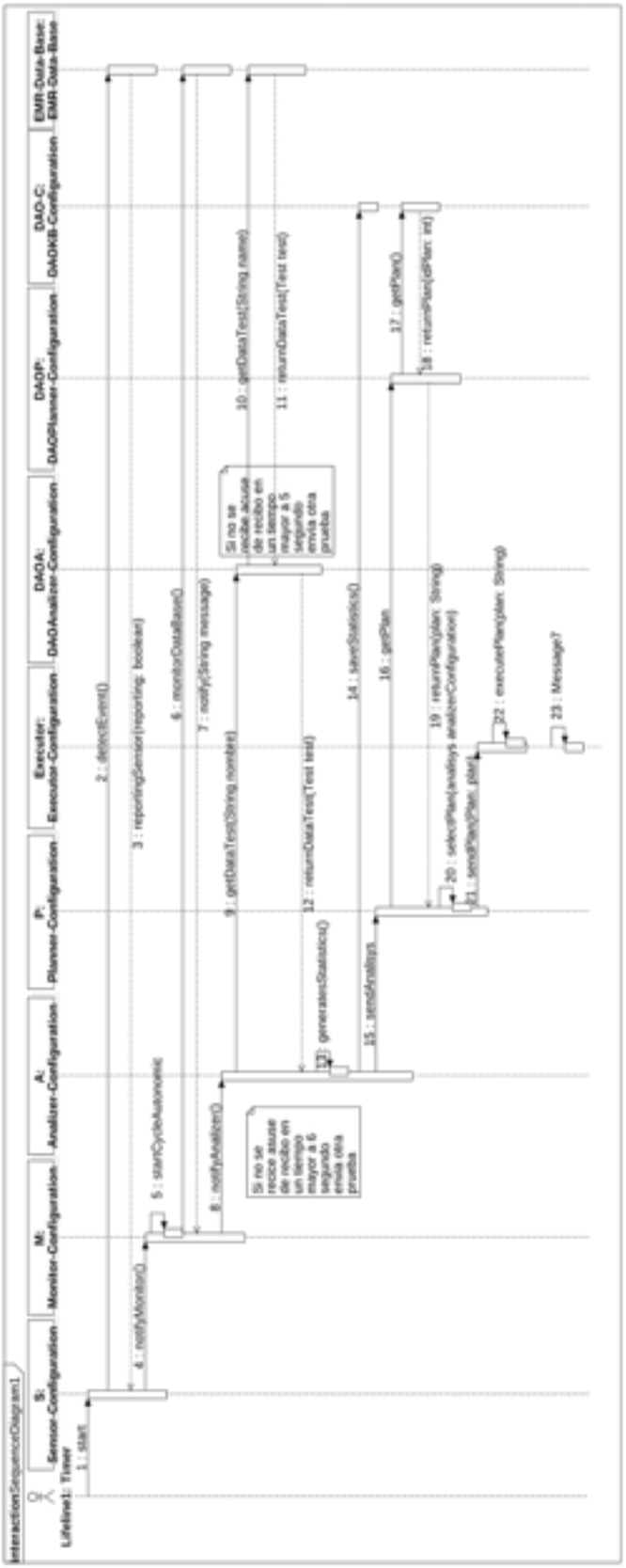

Figura 43. Diagrama de secuencia de un escenario autonómico perteneciente al Meta-S-MC.

Descripción detallada del escenario autonómico configurar una nueva base de datos.

1. En un momento determinado ha ocurrido un error en la base de datos.
2. El Sensor que se encuentra monitoreando la base de datos ha detectado un nuevo evento
3. El Sensor ha realizado un cambio de estado.
4. El Sensor ha enviado una notificación al Monitor.
5. El Monitor ha lanzado el ciclo autonómico.
6. El Monitor se encarga de monitorear la base de datos por un periodo de tiempo.
7. Se han detectado cambios en la base de datos.
8. Se envía una notificación al Analyzer.
9. El Analyzer realiza una petición al DAO para obtener las pruebas que se le van a realizar a la base de datos.
10. El DAO realiza una petición a la base de datos para obtener las pruebas.
11. La base de datos regresa las pruebas solicitadas por el DAO.
12. El DAO regresa las pruebas solicitadas por el Analyzer.
13. El Analyzer genera reportes en base a las pruebas realizadas.
14. El Analyzer guarda las pruebas.
15. El Analyzer envía las pruebas al Planner.
16. El Planner solicita los planes al DAO.
17. El DAO solicita los planes a la base de datos.
18. La base de datos envía los planes al DAO.
19. El DAO recibe los planes y los envía al Planner.
20. El Planner selecciona el plan adecuado en base al reporte generado por el Analyzer.
21. El Planner envía el plan seleccionado al Executor.
22. El Executor recibe el plan y lo ejecuta.
23. El Executor cierra el ciclo autonómico.

Como se muestra en el diagrama de casos de uso de la Sección 4.3.4.2 existen 4 casos de uso primarios, los cuales son transformados en escenarios autonómicos y dan soporte al Meta-S-MC. Es decir, es necesario realizar estos cuatro escenarios autonómicos, los cuales serán documentados utilizando la plantilla SAQAS.

4.3.4.4 Vista de procesos: Meta-S-MC

Diagrama de actividades: Meta-S-MC

El diagrama de actividades del Meta-S-MC tiene como propósito modelar el comportamiento del sistema y, más específicamente en nuestro caso, modelar el escenario autonómico, **configurar una nueva base de datos**. La Figura 44 muestra el diagrama de actividades que pertenece al escenario autonómico. Como primera entrada el diagrama de actividades tiene un **timer** que juega el papel del usuario principal, en un momento de operación normal la base de datos principal ha tenido una falla. El Sensor que se encuentra en la base de datos ha detectado un nuevo evento y ha cambiado de estado notificando al Monitor, este comienza a monitorear la base de datos por un periodo de tiempo y detecta que la base de datos no funciona, por lo que manda una notificación al Analyzer. Este realiza una serie de pruebas para verificar el funcionamiento de la base de datos, en base a las pruebas realizadas genera

un reporte que envía al Planner, este revisa el reporte generado por el Analyzer y selecciona el plan, el cual es detener la base de datos que se encuentra activa y configurar la base de datos que se encuentra en standby. El plan es enviado al Executor para que lo ejecute y realice las configuraciones necesarias para el correcto funcionamiento de la nueva base de datos. El Executor es el encargado de cerrar el ciclo autonómico.

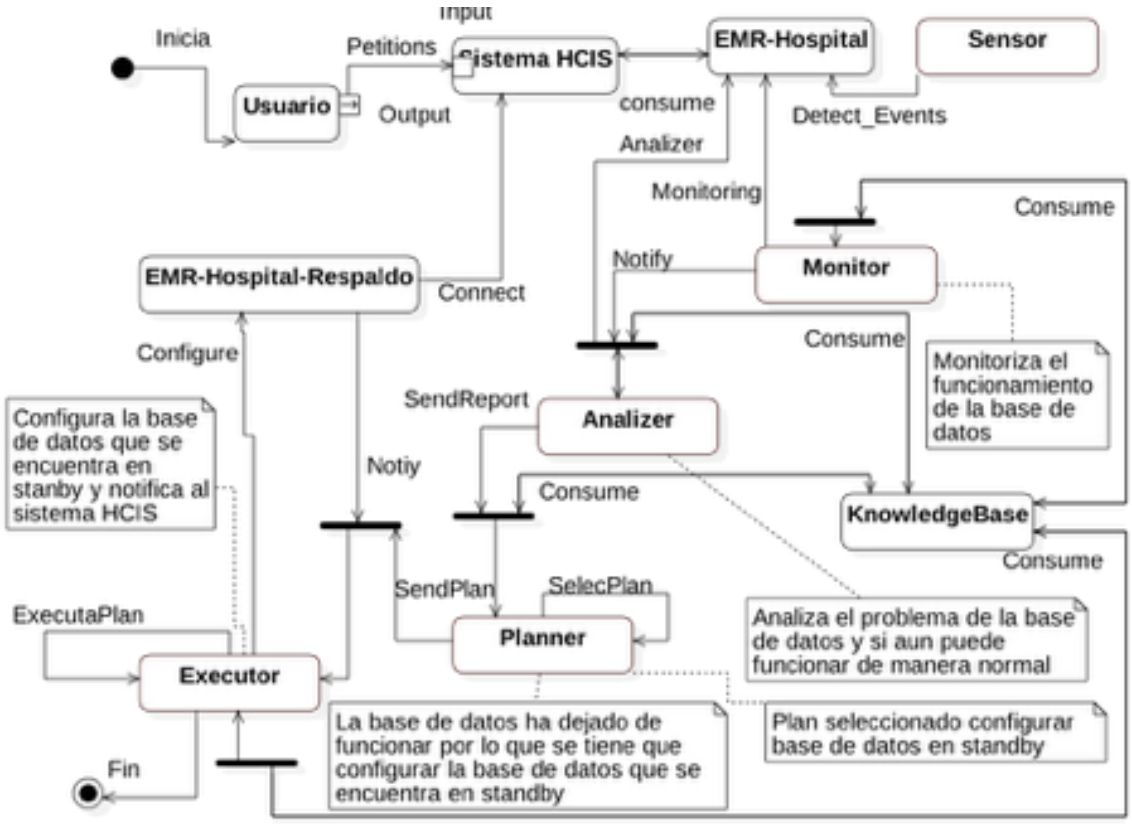

Figura 44. Diagrama de actividades del Meta-S-MC.

4.3.4.5 Vista de despliegue: Meta-S-MC

La vista de despliegue del Meta-S-MC está centrada en una arquitectura en capas en las cuales se implementa el patrón de diseño MAPE-K. El diseño de la arquitectura se representa en una jerarquía de capas, cada una de las cuales brinda una interfaz bien definida que ayuda a establecer la comunicación entre los diferentes niveles.

Diagrama de paquetes del Meta-S-MC

El diagrama de paquetes del Meta-Self-MC muestra una visión más clara de todos los elementos que componen la arquitectura, agrupando los elementos autonómicos y detallando las relaciones de dependencia entre ellos.

Para el diseño del Meta-S-MC se establece una arquitectura en capas la cual fomenta el principio de diseño de alta cohesión y el bajo acoplamiento. La primera capa está encargada

de soportar las reglas de negocio, es decir, se encarga de la funcionalidad del sistema, llevando a cabo los escenarios autonómicos propuestos en la Sección 4.3.4.2.

La capa de negocio del Meta-S-MC contiene cuatro componentes, cada uno de estos contiene una clase perteneciente al patrón MAPE-K, las cuales contienen los métodos necesarios para desarrollar el ciclo autonómico y para lograr el objetivo de configurar de forma autonómica los sistemas HCIS de apoyo al diagnóstico médico y a las bases de datos que contienen el registro médico de pacientes.

La capa Data-Layer muestra el paquete donde se encuentran los DAO's encargados de realizar la persistencia de la información del Meta-S-MH, como se muestra en la Figura 45. La comunicación con la capa de negocio se establece por medio de una interface bien definida y la comunicación con las bases de datos se establece por medio de una dependencia simple que es indicada por una flecha punteada.

La capa BakcEnd-Layer corresponde al paquete donde se encuentra el Sensor, esté verifica frecuentemente cualquier cambio en la base de datos y se encarga de enviar notificaciones de cualquier evento que suceda en la base de datos.

La capa Data Base contiene la base de conocimiento encargada de proporcionar el conocimiento al componente y también contiene la base de datos del hospital encargados de proporcionar la información de los pacientes.

4.3.4.6 Vista física del Meta-S-MC

Tomando en cuenta los atributos de calidad asociados al Meta-Self-MC, tales como la usabilidad y mantenibilidad, se muestra el diagrama de implantación con el objetivo de satisfacer dichos atributos.

Diagrama de implantación del Meta-S-MC

El diagrama de implantación del Meta-Self-MC permite visualizar la arquitectura física del hardware, software y todos los artefactos que se implementan dentro de la arquitectura como los patrones, en lugar de representar conceptualmente los usuarios y dispositivos que interactúan dentro de la arquitectura.

La Figura 46 representa el diagrama de implantación del Meta-Self-MH, el cual contiene 8 nodos. Estos permiten establecer relaciones entre nodos mediante un canal de comunicación. A continuación, se describe el funcionamiento de cada uno de los nodos:

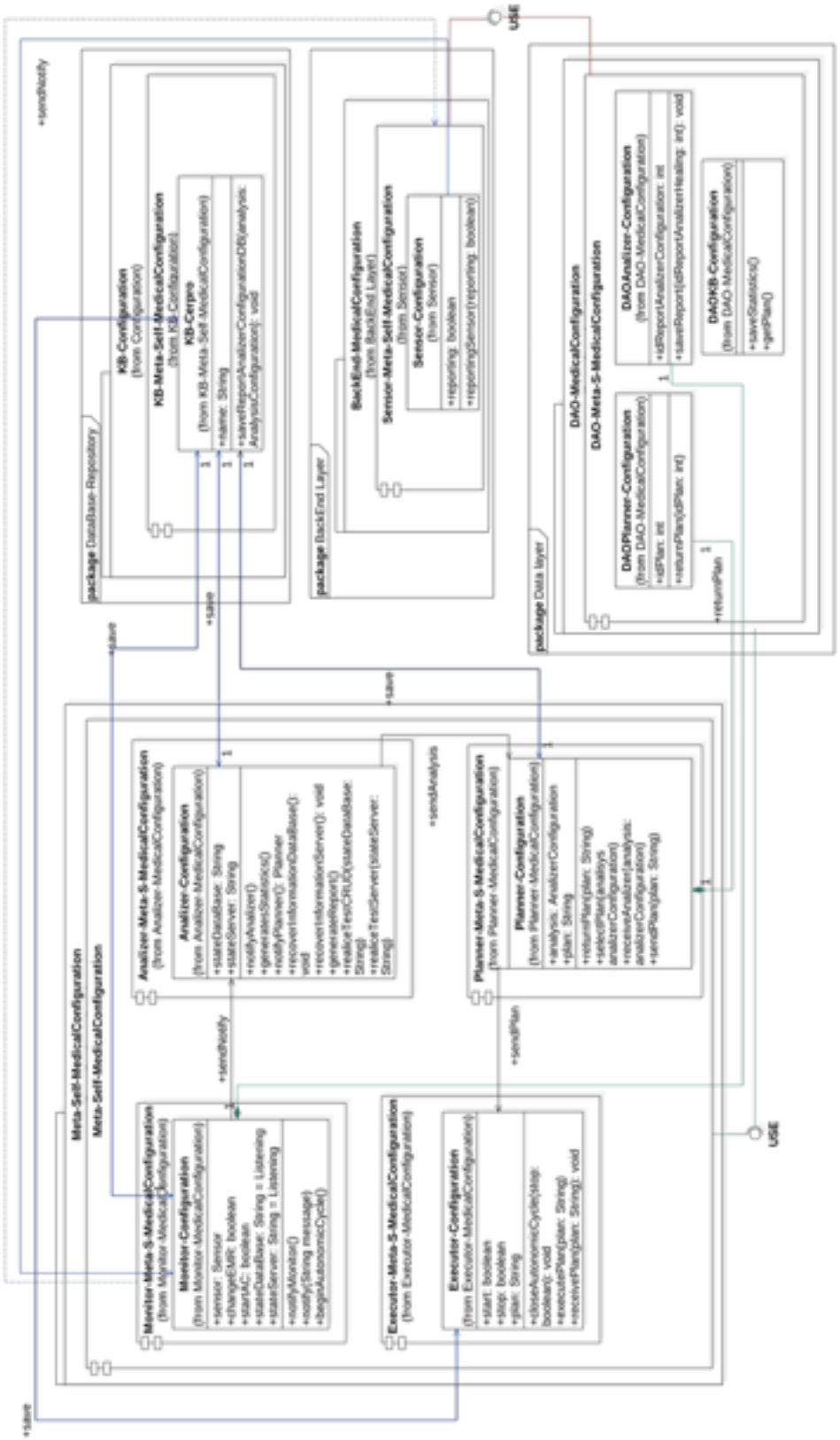

Figura 45. Diagrama de paquetes del Meta-S-MC.

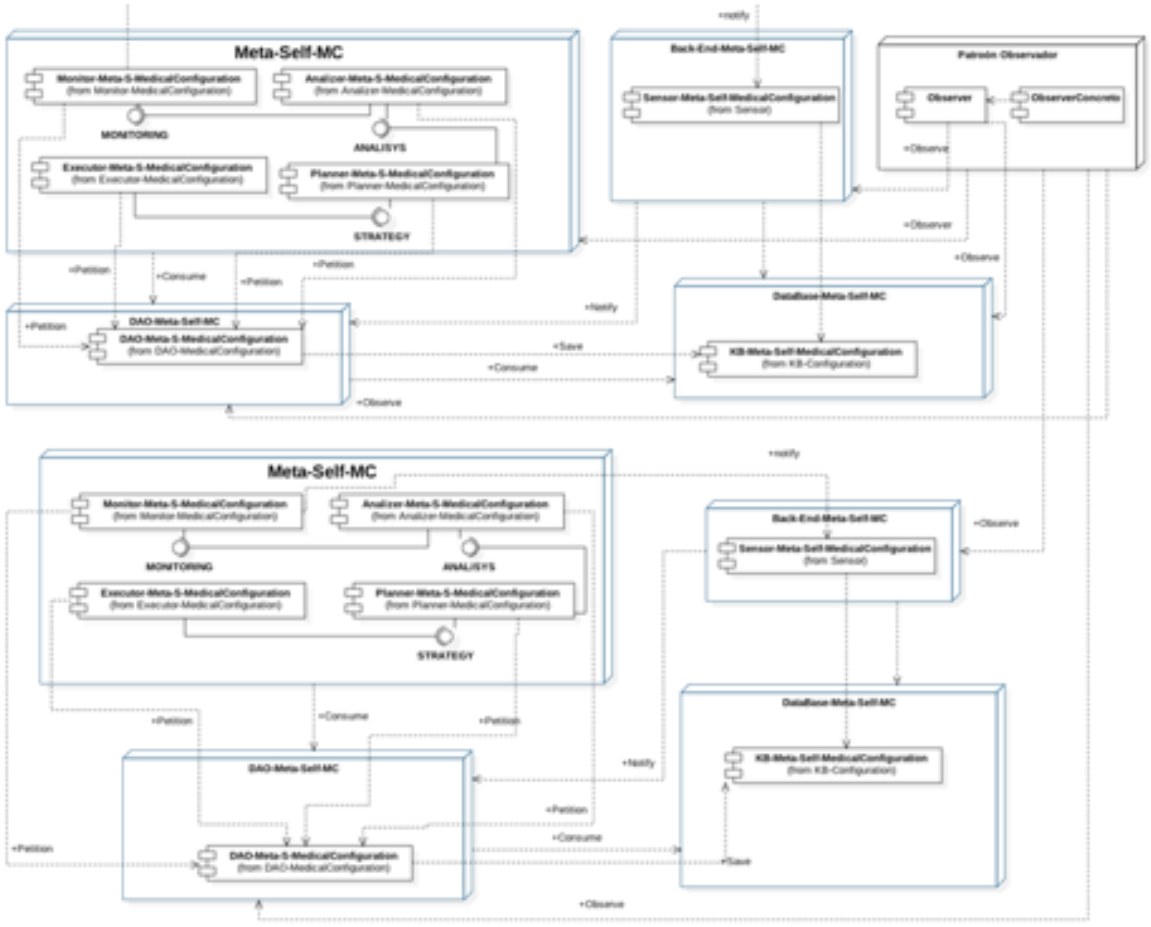

Figura 46. Diagrama de implantación del Meta-S-MC.

Nodo Meta-Self-MC: nodo encargado de comenzar el ciclo autonómico, dentro de este nodo se encuentran cuatro componentes encargados de ejecutar de forma automática la configuración de algunas partes de los sistemas HCIS como son: bases de datos, servidor que contiene al HCIS y la configuración de nuevos usuarios dentro del sistema HCIS.

Nodo DAO-Meta-S-MC: nodo encargado de atender las peticiones realizadas por cada uno de los componentes encontrados dentro del Meta-S-MC.

Nodo DataBase-Meta-Self-MC: nodo encargado de proporcionar la información requerida por el DAO-Configuration.

Nodo BackEnd-Meta-Self-MC: nodo encargado de detectar eventos en la base de datos y en el servidor que contiene al sistema HCIS. Cuando detecta un evento nuevo envía una notificación al Monitor.

Nodo Patrón observador- M-S-MC: nodo encargado de monitorear el funcionamiento de cada uno de los nodos pertenecientes al meta-componente M-S-MC. En caso de que uno de los nodos falle, el observador se encargará de notificar y despertar al nodo que se encuentra en standby.

La Figura 46 representa el diagrama de implantación del Meta-Self-MedicalConfiguration en el cual se observa que se tiene duplicados algunos de los nodos. Esto se debe a la aplicación de la táctica de redundancia activa la cual nos permite cubrir la tolerancia a fallos permitiendo tener nodos redundantes como son: Nodo Meta-Self-MC-Respaldo, Nodo DAO-Meta-S-MC-Respaldo, Nodo DataBase-Meta-Self-MC-Respaldo, Nodo BackEnd-Meta-Self-MC-Respaldo, manteniendo un estado síncrono con los nodos activos.

Capítulo 5

Proceso de instanciación de ARTLESS

Seguir un proceso permite realizar el trabajo de forma repetible, producir sistemas en un menor tiempo ya que para cada tarea dentro del proceso se conocen los tiempos de construcción y promover una mejora continua, permitiendo la eliminación de defectos.

Para **ARTLESS** se ha desarrollado un proceso de instanciación el cual recibe como nombre: inSTantiation procEss foR ARTLESS (STEER, Ver Figura 47). Esto ayuda a realizar el trabajo de forma repetible y predecible, estableciendo una serie de pasos ordenados en el tiempo que el arquitecto tendrá que seguir para implementar el diseño específico de la arquitectura de un sistema de software.

5.1 Mecanismos de instanciación

5.1.1 Primer nivel: Mecanismo de instanciación de la Meta-Architecture

El arquitecto de meta-modelado en el Punto A Figura 47 específica los meta-componentes que conforman a **ARTLESS**. Concretamente los siguientes cuatro: Meta-Self-MedicalDiagnosisAnalysis, Meta-Self-MedicalConfiguration, Meta-Self-MedicalHealing y Meta-Self-MedicalSecurity, cada una de estas meta-self cumple un objetivo diferente y podrá ser instanciada, utilizando el proceso **STEER** que previamente el arquitecto de meta-modelado ha definido para crear arquitecturas específicas a partir de **ARTLESS**.

5.1.2 Segundo nivel: Mecanismo de instanciación de System-Specific Architecture para un Healthcare Information System (HCIS)

En el System-Specific Architecture (Figura 47), el arquitecto de sistemas define la arquitectura específica del sistema de software a partir de ARTLESS siguiendo el proceso **STEER** (Figura 47, Punto C). Este proceso utiliza como entradas los puntos de la sección A y B Figura 47.

El Punto B (Figura 47), define las pre-condiciones que se deben de realizar antes de ejecutar el proceso **STEER**. El cliente junto con el arquitecto habrá desarrollado cada uno de los puntos que se enlistan a continuación:

1. Definen el alcance del proyecto.
2. Definen los requerimientos del sistema.
3. Definen los atributos de calidad.

El proceso de instanciación **STEER** está compuesto de dos actividades principales, utilizadas para crear instancias particulares de la arquitectura **ARTLESS**. Las Actividades 1 y 2 son las siguientes.

Tabla 31. Actividades de STEER

No de Actividad	Nombre de la actividad
Act-1	Realizar la selección de una self-feature.
Act-2	Crear una instancia de un meta-componente.

Para la creación del proceso **STEER** se ha utilizado Eclipse Process Framework (EPF) [32] basado en OpenUp. El EPF proporciona un marco extensible para la ingeniería de procesos de software como son: creación de métodos y procesos, administración de bibliotecas, configuración y publicación de un proceso.

En **STEER** se han definido una serie de pasos ordenados en el tiempo que el arquitecto tendrá que seguir para crear arquitecturas específicas a partir de **ARTLESS**.

El arquitecto del sistema realizará primero la Actividad 1 de esta depende la Actividad 2 como se muestra en la Figura 48. No empezar por la Actividad 1 repercutirá en tiempo de diseño, ya que el arquitecto probablemente no seleccione la self-feature más adecuada para su sistema y tenga que seleccionar otra self. La Actividad 1, está enfocada en dos tareas: 1) mostrar los pasos a seguir para seleccionar una self-feature, 2) mostrar los problemas que resuelve cada meta-componentes de **ARTLESS**. La Actividad 2 está enfocada en crear una instancia de alguno de los meta-componentes: Meta-S-MeDA, Meta-S-MeC, Meta-S-MeS, Meta-S-MeH, utilizando el proceso **STEER**. El arquitecto podrá realizar la selección de 1 hasta un máximo de 4 meta-componentes para implementar en su sistema. El número de componentes seleccionados depende del alcance del sistema de software que establezca el arquitecto junto con el cliente.

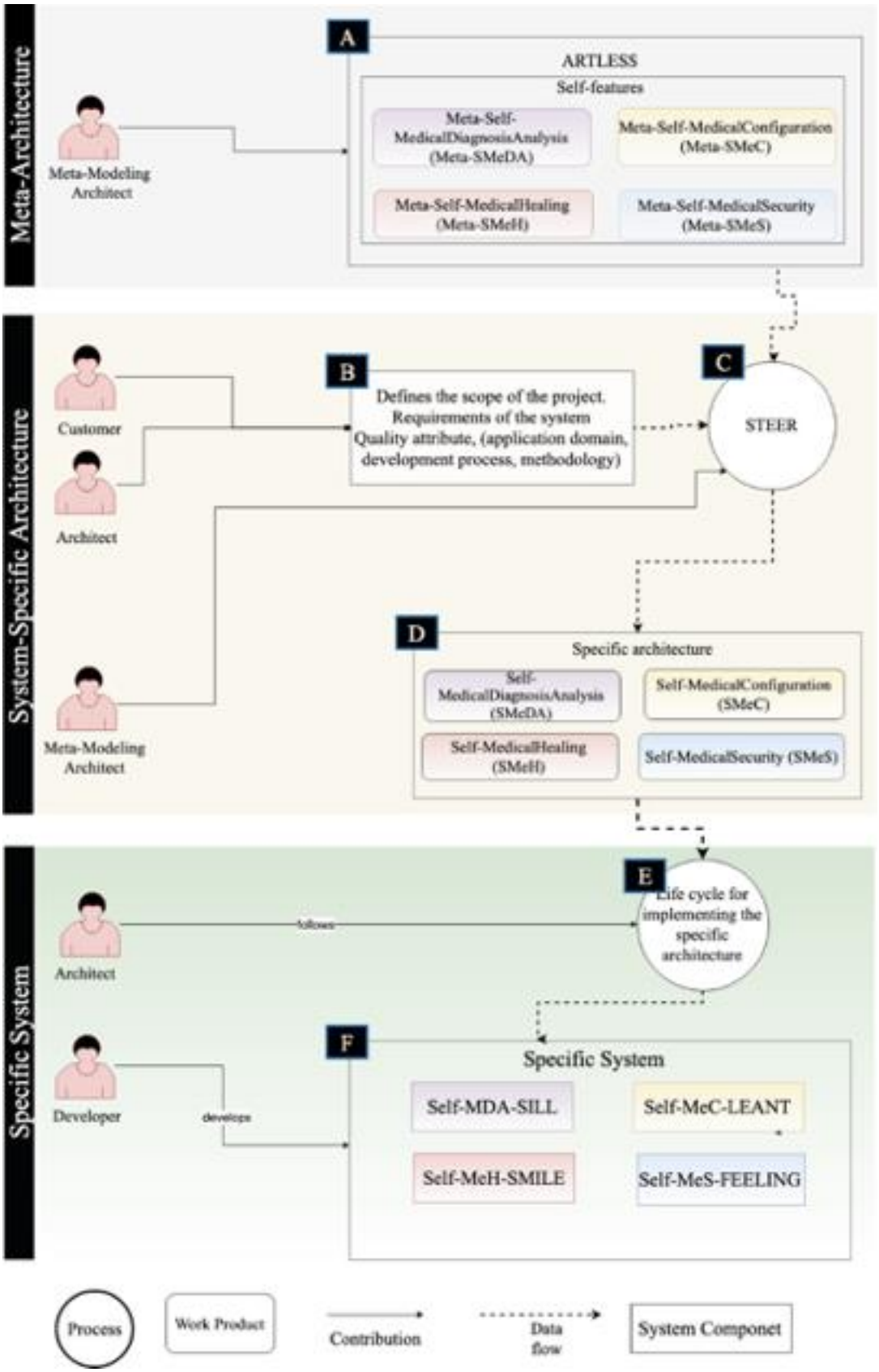

Figura 47. Mecanismos de instanciación.

En esta tesis se propone que el primer meta-componente a implementar sea el Meta-S-MeDA ya que es una contribución para mejorar los sistemas HCIS de apoyo al diagnóstico médico, el componente se enfoca en filtrar pacientes de acuerdo con su diagnóstico médico. Posteriormente se recomienda que se implemente el Meta-S-MS ya que mediante el análisis del estado del arte se encontró que los sistemas HCIS existentes son vulnerables en seguridad por el alto costo y las deficientes arquitecturas para desarrollar sistemas seguros, además de que la seguridad en los sistemas médicos es de importancia por el alto valor de la información de los pacientes. En un tercer lugar se recomienda que se instancie el Meta-S-MeH por el nivel de impacto que pudiera tener si un sistema HCIS llegara a fallar. Finalmente se recomienda que se instancie el Meta-S-MeC ya que la configuración dentro de un sistema HCIS tiene un menor impacto.

El arquitecto del sistema tendrá que seguir los pasos descritos para la Actividad 1 (*Tabla 32*). El tiempo aproximado para realizar la selección de una self-features es de 2 a 3 horas, ya que se tienen que revisar las especificaciones de cada self.

Tabla 32. Selección de self-features

Iteración	Objetivos primarios (Tareas y casos de uso)	Tiempo
E-1		Aprox. 1 hora.
	1.-Analizar los escenarios en los que se puede realizar una instanciación de las self-features.	
	2.-Analizar los atributos de calidad que cubre cada self-feature.	Aprox.1 hora.
	*Para el caso en particular del Meta-S-MC, se plantea tener una infraestructura redundante por lo que si se desea realizar una instancia de este meta-componente se deberá considerar el presupuesto para implementar y desarrollar este componente.	
	3.-Analizar los requerimientos no funcionales que cubre cada self-feature.	Aprox. 1 hora.
	4.-En base a los análisis realizados y a la tabla de mapeo de drivers arquitectónicos a self-feature seleccionar las self a implementar.	

Cuando el arquitecto ha terminado la Actividad 1. Se procede a realizar la selección de un meta-componente (Actividad 2, *Tabla 31*). Siguiendo el proceso **STEER** (ver Figura 48), el arquitecto diseñará la arquitectura específica de un componente autonómico para un sistema HCIS con características genéricas y particulares al sistema que va a construir. El proceso **STEER** (*Tabla 33*), describe todos los pasos necesarios para diseñar arquitecturas específicas a partir de **ARTLESS**.

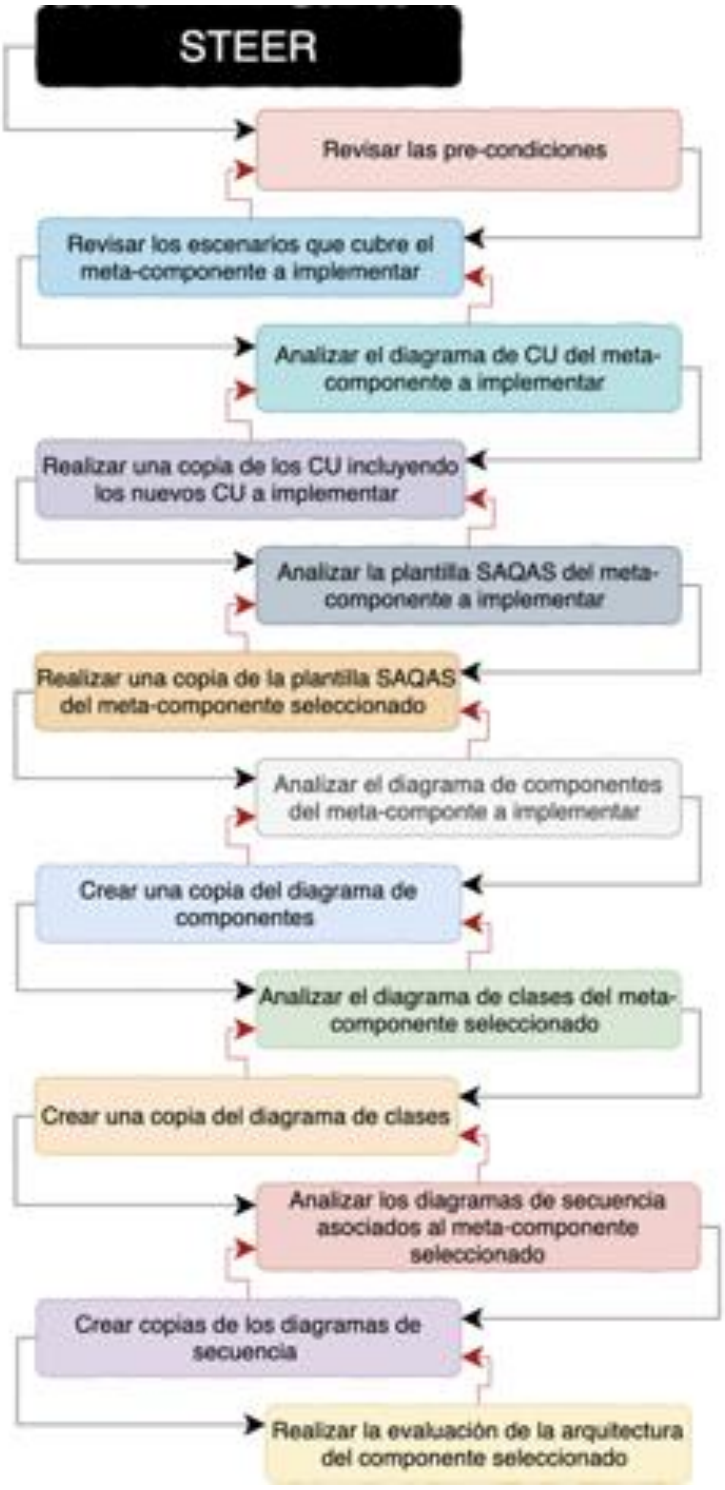

Figura 48. Pasos del proceso STEER.

Tabla 33. Proceso de instanciación de un meta-componente

Iteración	Tareas a realizar	Descripción
E-2	1.-Revisar y realizar las pre-condiciones.	El arquitecto del sistema junto con el cliente define el alcance del proyecto.

2.-Revisar los escenarios que cubre el meta-componente a implementar.	En un segundo paso se tiene que realizar una revisión de los escenarios que cubre cada meta-componente. Esto se realiza con el objetivo de proporcionar más información al arquitecto de cada componente, y de esta forma el arquitecto puede rectificar la selección del meta-componente.
3.- Analizar el diagrama de casos de uso.	En un tercer paso se analizará el diagrama de caso de uso del meta-componente seleccionado. Este diagrama sirve para especificar la comunicación y el comportamiento de un sistema mediante la interacción con los usuarios y contiene la descripción de las actividades que se tendrán que realizar.
4.- Realizar una copia del diagrama de casos de uso, incluyendo los casos de uso a implementar.	Una vez que se han seleccionado las actividades a realizar, se desarrolla el diagrama de casos de uso, con los usos seleccionados se incluyen las características propias del sistema a realizar.
5.-Analizar la plantilla SAQAS [7], del meta-componente.	Hasta este punto se han seleccionado los usos del sistema, pero no se ha mencionado como se implanta la solución. Para esto tomamos la plantilla SAQAS la cual describe la solución autonómicamente mediante la aplicación MAPE-K. Estudiar la solución del problema de la plantilla SAQAS brindará al arquitecto el conocimiento necesario para crear sus propias soluciones autonómicas de las características particulares del sistema a implantar.
6.-Realizar una copia de la plantilla SAQAS del meta-componente seleccionado.	Una vez que se ha comprendido cómo desarrollar la solución mediante la ejecución del patrón MAPE-K, se realizará una copia de la plantilla SAQAS con los escenarios a implementar. Aquí el arquitecto podrá agregar la solución a sus características particulares siguiendo el patrón MAPE-K.
7.-Analizar el diagrama de componentes del meta-componente a implementar.	En un nivel superior encontramos a los diagramas de componentes, estos muestran las dependencias entre componentes y dentro de estos incluyen las clases creadas a partir del patrón MAPE-K. Realizar el análisis del diagrama de componentes proporciona al arquitecto un mejor conocimiento de cómo implantar una solución autonómica.
8.-Crear una copia del diagrama de componentes.	Crear una copia del diagrama de componentes proporciona una vista de la solución del problema.

	9.-Analizar el diagrama de clases del meta-componente seleccionado.	Analizar el diagrama de clases ayudará al arquitecto a entender cómo interactúan las clases mediante la aplicación del patrón MAPE-K.
	10.-Crear una copia del diagrama de clases.	El diagrama de clases ayuda al arquitecto a describir la estructura del sistema.
	11.-Analizar los diagramas de secuencia asociados al meta-componente seleccionado.	Los diagramas de secuencia muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Es importante que el arquitecto analice y comprenda los diagramas de secuencia ya que a través de estos implementan una solución autonómica.
	12.-Crear copias de los diagramas de secuencia.	Ayudan al desarrollador a tener un mejor entendimiento de cómo se implementa una solución autonómica.
	13.-Realizar la evaluación de la arquitectura del componente seleccionado.	Garantizar sistemas de mejor calidad.

El resultado de aplicar **STEER** a la meta-arquitectura **ARTLESS** será el diseño de una arquitectura específica (Punto D Figura 47), con las características genéricas de **ARTLESS** más las características específicas del sistema que se desea construir.

La primera vez que se aplique el proceso **STEER** se realizará todo el ciclo sin obviar ningún paso, esto se realizará para los cuatro meta-componentes. La segunda vez que se realice una instancia de un meta-componente se podrán obviar pasos, esto se realiza con el objetivo de reducir el tiempo de instanciar la meta-arquitectura. Por ejemplo, uno de los pasos que se podrá omitir será el análisis de la plantilla SAQAS ya que el arquitecto con anterioridad habrá realizado el análisis de la plantilla. El haber realizado el proceso **STEER** completo ayudará al arquitecto a omitir varios pasos la siguiente vez que instancie el meta-modelo de arquitectura.

5.1.3 Tercer nivel: Mecanismos de instanciación de Specific System

En el Sistema Específico, el arquitecto del sistema utiliza el ciclo de vida (Punto E, Figura 47). Para implementar una arquitectura específica, el equipo de desarrollo se encarga de seguir esta arquitectura e implementar el sistema de software.

El ciclo de vida propuesto para el diseño específico de arquitectura es una combinación del modelo secuencial y el modelo iterativo, como se muestra en la Figura 49. El modelo es secuencial por que se tiene que desarrollar una serie de actividades ordenadas. Por ejemplo, la Actividad 2 tiene que ser precedida de la Actividad 3 y no puede ser de forma contraria ya que la Actividad 2 corresponde a realizar las pre-condiciones del sistema y la Actividad 3 corresponde a instanciar un meta-componente. Es iterativo por qué se puede realizar el diseño

de más de un componente al mismo tiempo como se muestra en la Figura 49, se han realizado las instancias del Meta-componente 1 y 2 al mismo tiempo.

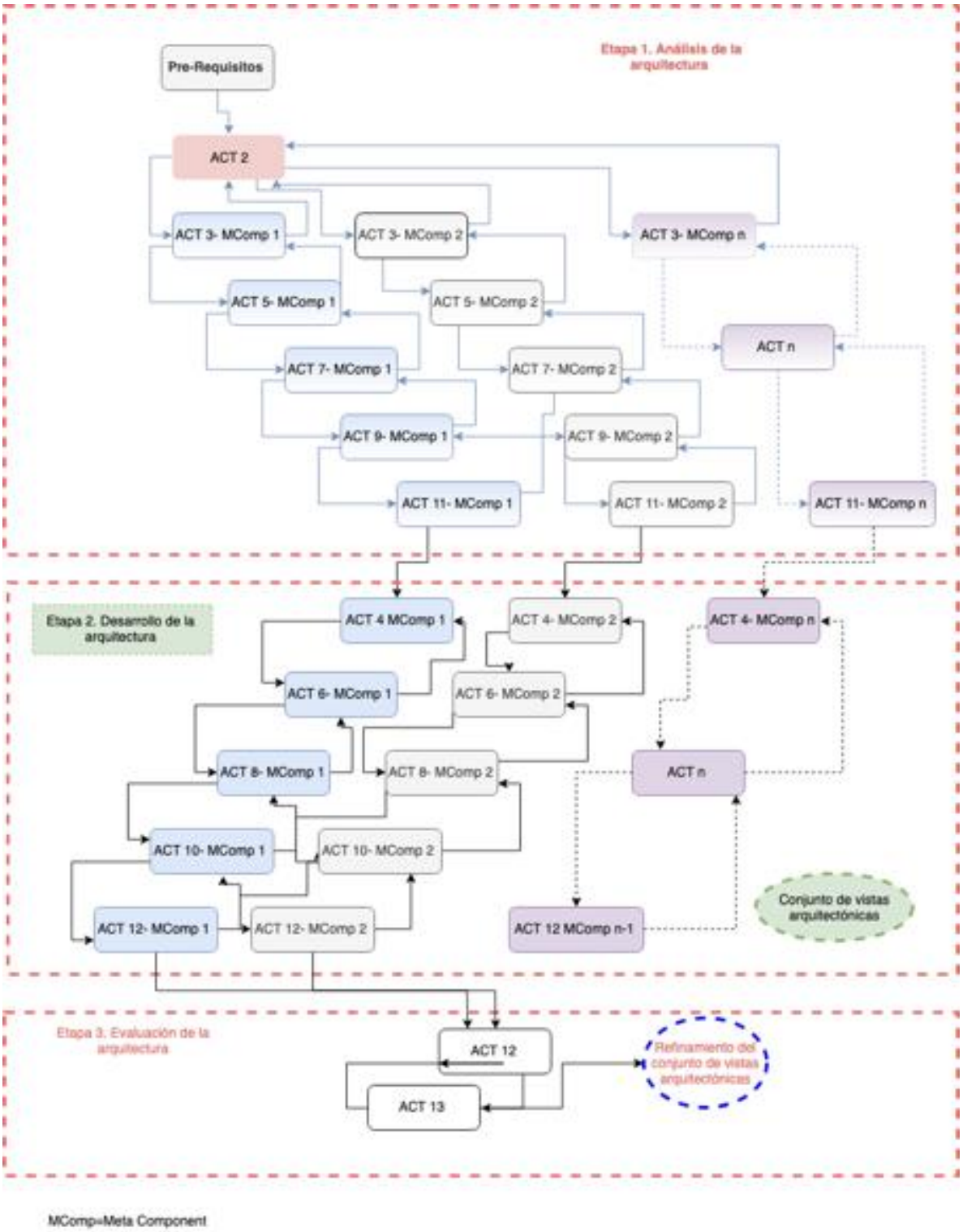

Figura 49. Ciclo de vida.

El resultado de haber ejecutado el ciclo de vida será el diseño de la arquitectura de un componente de un sistema de software que contendrá características comunes a todos los sistemas HCIS, pero también tendrá características específicas al sistema que se construyó (Figura 47, Punto F).

5.2 Integración con el ciclo de vida de desarrollo de software

El proceso **STEER** tiene que ser integrado dentro del marco de algún proceso de desarrollo de software, ya que este proceso es una aportación al diseño de la arquitectura y no al levantamiento de requerimientos, construcción y a la implantación. Por lo que el proceso tiene que ser complementado siguiendo un proceso de más alto nivel como lo es OpenUp, SCRUM o algún otro.

En esta tesis se propone una muestra de cómo integrar el proceso **STEER** en otro proceso de alto nivel como lo es OpenUp, para el desarrollar software. El proceso **STEER** describe los pasos para crear arquitecturas específicas para sistemas HCIS de apoyo al diagnóstico médico, principalmente desarrolla dos actividades: 1) selección de una self-feature, 2) instanciar un meta-componente. Estas actividades pertenecen al diseño de la arquitectura por lo que están dentro del marco de OpenUp y se encuentran en la fase de Elaboración en la iteración de Desarrollo de la Arquitectura (Figura 50).

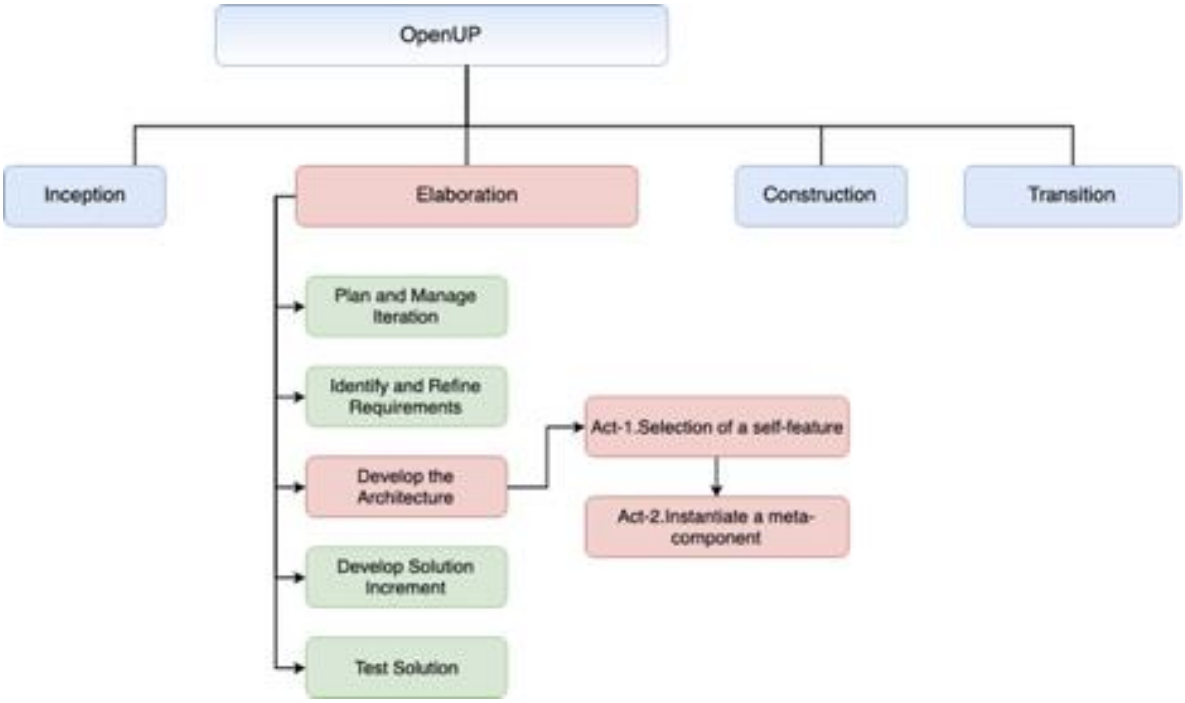

Figura 50. Integración de Actividades 1 y 2 de STEER a OpenUp.

5.3 Instanciación de un meta-componente

Para ejemplificar el proceso de instanciación a un nivel inferior se ha desarrollado la Figura 51, la cual muestra la ejecución del proceso **STEER** y el ciclo de vida del meta-componente Meta-Self-MedicalSecurity (Meta-SMeS), a través de sus diferentes niveles. El componente está expresado en dos dimensiones. La primera dimensión horizontal muestra las capas del componente Meta-SMS como son: Business layer, Back-End layer y Data layer a través de las cuales se implementa el patrón MAPE-K. La segunda dimensión es transversal y está ejemplifica los mecanismos de instanciación del Meta-Componente.

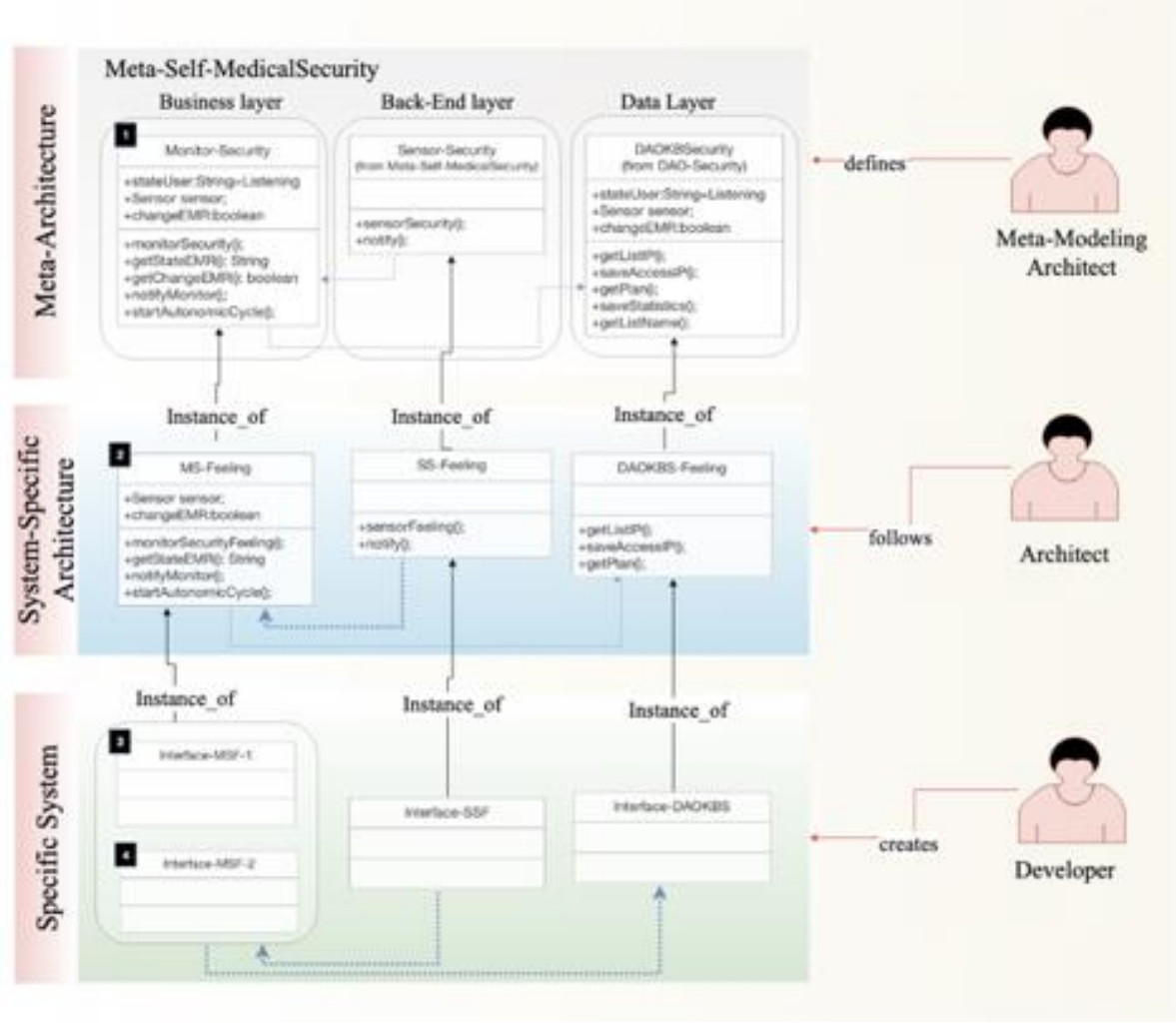

Figura 51. Proceso de instanciación de un meta-componente.

Dentro de la meta-arquitectura **ARTLESS** se encuentra el componente Meta-SMS el cual es definido por el arquitecto de meta-modelado como se indica en la Figura 51. Los pasos que seguir para crear un sistema de software que implementa el componente Meta-S-MS son los siguientes:

1. El arquitecto habrá realizado las pre-condiciones del sistema.

2. Teniendo como entradas las pre-condiciones y el proceso STEER se desarrolla el diseño de la arquitectura específica. Por ejemplo, el arquitecto en la Figura 51, después de haber ejecutado el proceso de instanciación de la clase Monitor-Security (Figura 51, punto 1), ha dado como resultado la clase MS-Feeling (Figura 51, punto 2), la cual contiene algunos métodos y variables de la clase superior.
3. El arquitecto se encarga de seguir el ciclo de vida. Por ejemplo, después de aplicar el ciclo de vida a la clase MS-Feeling (Figura 51, punto 2), se ha producido la interfaz mostrada en los puntos 3 y 4.

Capítulo 6

Evaluación del meta-modelo de arquitectura ARTLESS

6.1 Evaluación de la arquitectura para una instancia de sistema HCIS

La evaluación de la arquitectura se lleva a cabo utilizando una versión del método **ATAM**. El método de evaluación ATAM consta de 4 fases: preparación, familiarización, evaluación y reporte final. Para nuestro caso de estudio solo se verán las fases de familiarización, evaluación y reporte final, que permiten identificar: sin los involucrados (el equipo ATAM y el equipo de arquitectura), riesgos, puntos de sensibilidad y validación del diseño (meta-modelo) de arquitectura.

La *Tabla 34* muestra los 8 pasos de la versión ATAM considerados en esta tesis

ID	Paso	Considerado en el caso de estudio	No considerado en el caso de estudio
1	Presentación de la versión del ATAM		En este caso no se realizará la presentación de la versión del ATAM, ya que no se cuenta con un equipo de arquitectura y lo que se presenta en este caso es una validación de la instanciación de la arquitectura.
2	Presentación de los drivers de negocio	Solo para el diseño autonómico del sistema se consideran los drivers de la *Tabla* 34. Para el diseño de todo el sistema se tienen que considerar los drivers	

		funcionales, restricciones y escenarios (del taller de atributos de calidad QAW).	
3	Presentación de la arquitectura	Se presenta la arquitectura mediante el diseño de la arquitectura resultante del proceso de instanciación del meta-componentes: Meta-S-MDA.	
4	Identificación de las decisiones arquitectónicas	Hasta este paso se ha realizado la presentación de los drivers arquitectónicos y el diseño de la arquitectura, ahora se analizará toda la información obtenida en los pasos 2 y 3, esto con el fin de tener una base sólida para el análisis. En el análisis se presentan cuestiones de cómo reacciona el sistema ante escenarios de crecimiento, interconectividad, seguridad, etc.	
5	Generación del árbol de utilidad	Árbol presentado por el equipo de evaluación, el árbol presenta los principales escenarios identificados, así como su importancia y su complejidad.	
6	Análisis de las decisiones arquitectónicas	El equipo expone a los participantes las principales decisiones arquitectónicas identificadas en la fase de familiarización.	
7	Generación y priorización de escenario	El equipo se basa en el árbol de utilidad, y mediante una técnica de lluvia de ideas, guía en este punto a los participantes en la obtención de escenarios adicionales (Repetir paso 6).	
8	Presentación de los resultados		

	Este paso resume las entradas a la evaluación y los hallazgos de esta: puntos de sensibilidad, relaciones de equilibrio entre atributos de calidad y riesgos.

6.2 Identificación de las decisiones arquitectónicas

Las decisiones arquitectónicas más importantes son:

1. Se estructura el sistema en capas, en cada una de estas se establecen elementos correspondientes al dominio del problema.

2. Se hace uso del patrón MAPE-K, el cual está implementado para realizar una solución autonómica.

3. La conexión entre cada uno de los componentes se realiza por medio de interfaces, fomentando la alta cohesión y el bajo acoplamiento.

4. Se proponen diferentes patrones de diseño y tácticas que ayudan a soportar problemas específicos, como: Por ejemplo, la redundancia activa la cual se encarga de duplicar los componentes dentro del sistema, evitando que el sistema se caiga por un tiempo prolongado.

6.3 Generación del árbol de utilidad

La *Tabla 35* muestra los principales escenarios relevantes para la arquitectura. Se utiliza cuando el número de personas participando en la evaluación es reducido como en este caso. El árbol inicia con la utilidad como elemento más importante y de ahí se especializa en los atributos de calidad clave de acuerdo con los drivers identificados. También se analiza el documento de visión y alcance, QAW, y el documento de drivers.

Tabla 35. Árbol de utilidad

Utilidad Capacidad que tiene una cosa de servir o de ser aprovechada para un fin determinado	Funcionalidad	Básica
		Deseable
	Compatibilidad —> Interoperabilidad	Sistemas HCIS
		Dispositivos externos
	Desempeño	Tiempo de respuesta
		Volumen de transacciones
	Disponibilidad	Fallas de software
		Fallas de comunicación
	Usabilidad—> Efectividad	Enfocado a una tarea

Dado que la evaluación busca encontrar escenarios que detecten riesgos en la arquitectura, los posibles escenarios son:

Tabla 36. Árbol de utilidad evaluación

Utilidad	Funcionalidad	Básica	(A, A) Filtrar pacientes de acuerdo con su diagnóstico médico. (A, A) Obtener nuevos pacientes de manera autónoma. (B, A) Recuperar imágenes médicas de pacientes.
		Deseable	(A, M) Generar reportes de pacientes atendidos al día.
	compatibilidad —> Interoperabilidad	Sistemas HCIS	(A, M) Recuperar información de sistemas que no implementen un estándar.
		Dispositivos externos	(A, A) Configurar dispositivos médicos. (A, M) Establecer comunicación con dispositivos que no implementen un estándar médico.
	Desempeño	Tiempo de respuesta	(A, M) Consultar la información de un paciente en una situación particular. (A, A) Consultar la información de un paciente que tenga imágenes médicas.
		Volumen de transacciones	(A, M) 30 usuarios consultan información médica con tiempo de respuesta < 1 minuto.
	Disponibilidad	Fallas de software	(M, M) Falla de la base de datos
		Fallas de comunicación	(M, M) Falla al establecer comunicación con los dispositivos médicos.
	Efectividad	Tiempo de respuesta	(M, M) Registro de un paciente en un tiempo < 5 minutos.

*Se consideran algunos de los escenarios.

Se indica la relevancia de cada escenario para el negocio, así como la complejidad de implementarlo.
A= Alta.
M= Media.
B= Baja.
Por ejemplo.
(A, A) = (Alta importancia para el cliente/negocio, Alta dificultad de implantación)

6.4 Análisis de las decisiones arquitectónicas

Para llevar a cabo el análisis de las decisiones de arquitectura se seleccionan los escenarios más importantes para el negocio y los de mayor complejidad. También se pueden combinar algunos de los escenarios para la simplificación del análisis.

Escenario para utilizar: Filtrar pacientes de acuerdo a su diagnóstico médico.

En las *Tablas 37* y *38* se muestran el Análisis de enfoques arquitectónicos para algunos escenarios.

Tabla 37. Análisis de enfoques arquitectónicos

Análisis de enfoques arquitectónicos				
Escenario núm. 1	Filtrar pacientes de acuerdo a su diagnóstico médico.			
Atributo(s)	Usabilidad, Disponibilidad, Compatibilidad			
Entorno	EMR (Electronic Medical Record)			
Estímulo	Un nuevo paciente ha llegado a la base de datos sin ser asignado a un médico.			
Respuesta	El sistema debe detectar de manera automática que se recibió un nuevo paciente que se le ha asignado un diagnóstico médico y en base a este se le asignará un médico especialista.			
Decisiones arquitectónicas	**Punto de sensibilidad**	**Equilibrio**	**Riesgo**	**No-riesgo**
El sistema está estructurado en capas.	Con el diseño el punto de sensibilidad es de una transacción a la vez. Es decir, solo se puede filtrar un paciente a la vez. No se tiene documentación en el diseño o alguna evidencia de que pueda soportar más de una transacción simultánea. Es decir, que se pueda realizar la filtración de pacientes de forma concurrente.	Hay equilibrio entre mantenibilidad y desempeño pues las capas facilitan la modificación de componentes.	R1- No se tiene ningún registro para el manejo de seguridad de los datos de pacientes. R2 - No se tiene ningún registro que el manejo de la información de los pacientes se realice bajo un estándar médico como el HL7.	NR-1Mantenibilidad por la separación de capas. NR-2 Integración de las capas entre sí.
Capa de negocio	La filtración de pacientes se realiza de forma secuencial, no se tiene registro que la filtración de pacientes se realice de forma concurrente.	Facilidad de uso y desempeño ya que esta capa está dedicada a realizar todas las cuestiones de negocio como la asignación de pacientes y no involucrar cuestiones de persistencia de datos	R3- La capa de negocio no esquematiza de forma clara la relación con la capa del backend. R4- No hay evidencia de que el sistema permite realizar	NR-2 Mantenibilidad por la separación de capas.

			transacciones concurrentes.	
Capa de datos	No hay información para determinar el punto de sensibilidad de cuántos pacientes ser almacenados al mismo tiempo. No hay información para determinar cuántos pacientes pueden pueden ser registrados en la base de datos.		R-5 Falta la documentación de la táctica para ver cómo se implementa dentro de un sistema autonómico.	NR-3 Mantenibilidad por la separación de capa. NR-4 Integración de las capas
Patrón MAPE-K	No hay información para determinar el punto de sensibilidad de qué tan difícil es utilizar este patrón.		R-6 Curva de aprendizaje no sea mayor a dos semanas.	
Razonamiento	Este escenario es fundamental y las decisiones arquitectónicas planteadas pueden implementarlo. Sin embargo, la falta de detalles genera riesgos respecto de los cuales hay que asegurarse de manera más precisa que la arquitectura pueda manejar.			
Diagrama Arquitectónico	Diagrama de componentes. Diagrama de clases. Diagrama de implantación.			

Tabla 38. Análisis de los enfoques arquitectónicos escenarios dos

Análisis de enfoques arquitectónicos				
Escenario núm. 2	30 usuarios concurrentes consulta información de pacientes con tiempo de respuesta inferior a < 1 minuto.			
Atributo(s)	Desempeño.			
Entorno	En un momento de operación hay más de 30 usuarios consultando la información de distintos pacientes.			
Estímulo	Un usuario con acceso al sistema HCIS realiza una consulta.			
Respuesta	El sistema devuelve la información de la consulta en un tiempo < 1 minuto.			
Decisiones arquitectónicas	**Punto de sensibilidad**	**Equilibrio**	**Riesgo**	**No-riesgo**
Sistema estructurado en capas	El sistema permite realizar una transacción. El sistema no permite la concurrencia.		R-7 La arquitectura no puede manejar el escenario de desempeño más específicamente el atributo de calidad de escalabilidad.	
Razonamiento				

	Este escenario es fundamental y las decisiones arquitectónicas planteadas pueden implementarlo. Sin embargo, la falta de detalles genera riesgos respecto de los cuales hay que asegurarse de manera más precisa que la arquitectura pueda manejar.
Diagrama arquitectónico	Diagrama de componentes. Diagrama de clases. Diagrama de implantación.

6.5 Resultados de la evaluación: listado de riesgos obtenidos

En la *Tabla 39* se muestran los riesgos identificados en los escenarios propuestos para la evaluación. Mientras que la *Tabla 40* enlista los requerimientos no funcionales asociados al sistema encontrados después de haber ejecutado la evaluación de la arquitectura especifica con ATAM.

Tabla 39. Lista de riesgos

ID Riego	**Descripción del riesgo**
R-1	No se tiene ningún registro para el manejo de seguridad de los datos de pacientes.
R-2	No se tiene ningún registro que el manejo de la información de los pacientes se realice bajo un estándar médico como el HL7.
R-3	La capa de negocio no esquematiza de forma clara la relación con la capa del backend.
R-4	No hay evidencia de que el sistema permita realizar transacciones concurrentes.
R-5	Falta la documentación de la táctica para ver cómo se implementa dentro de un sistema autonómico.
R-6	Curva de aprendizaje no sea mayor a dos semanas para el patrón MAPE-K.
R-7	La arquitectura no puede manejar el escenario de desempeño más específicamente el atributo de calidad de escalabilidad.

Lista de requerimientos no funcionales asociados al sistema encontrados después de haber ejecutado la evaluación de la arquitectura específica con ATAM.

Tabla 40. Lista de riesgos de requerimientos no funcionales

ID	Descripción
RNF-1	El sistema debe filtrar a los pacientes de forma exitosa
RNF-2	EL sistema soporta el aumento de usuarios en un 30%
RNF-3	El sistema debe tener una fiabilidad del 98%, es decir, debe de funcionar los 365 días del año ininterrumpidamente

RNF-4	El sistema persiste la información y genera reportes bajo el estándar médico HL7

6.6 Validación de la arquitectura en función de los riesgos.

Objetivos:

- Obtener un meta-modelo de arquitectura que tenga riesgos aceptables en función de la aplicación del ATAM.

- Desarrollar arquitecturas de calidad bajo la implementación de los siguientes atributos de calidad: usabilidad, disponibilidad y compatibilidad, tomando en cuenta los siguientes requerimientos no funcionales.

La *Tabla 41* muestra los atributos de calidad evaluados asociados a los requerimientos no funcionales. Mientras que la *Tabla 42* muestra las correcciones realizadas a la arquitectura específica después de haber ajustado la arquitectura en función de los riesgos obtenidos.

Tabla 41. Atributos de calidad evaluados asociados a los requerimientos no funcionales.

Atributo de calidad	RNF
Usabilidad	RNF-1. Es sistema debe filtrar a de los pacientes recibidos de forma exitosa.
Disponibilidad	RNF-2. El sistema soporta el aumento de usuarios en un 30%.
	RNF-3. El sistema debe tener una fiabilidad del 98%, es decir debe funcionar 365 días del año ininterrumpidamente.
Compatibilidad	RNF-4. El sistema genera reportes bajo el estándar médico HL7.

Tabla 42. Descripción de los riegos y solución

ID Riego	Descripción del riesgo	Solución
R-1	No se tiene ningún registro para el manejo de seguridad de los datos de pacientes.	El manejo de la seguridad de los datos de pacientes podrá ser manejada bajo la implementación del componente autonómico Meta-Self-MedicalSecurity.
R-2	No se tiene ningún registro que el manejo de la información de los pacientes se realice bajo un estándar médico como el HL7.	Mediante la solución autonómica se ha decidido que el componente Analyzer, analice los datos obtenidos y los modifique en caso de que sea necesaria para que la persistencia de la información en la base de conocimiento sea bajo el estándar médico HL7.

R-3	La capa de negocio no esquematiza de forma clara la relación con la capa del backend.	Se ha definido un puerto en la capa de negocio y en la capa de backend para que la notificación que envía el Sensor sea por el port 1.
R-4	No hay evidencia de que el sistema permita realizar transacciones concurrentes.	Se ha implementado el patrón mediador para distribuir las transacciones y así evita cuellos de botella.
R-5	Falta la documentación de la táctica *Pause and Resume* para ver cómo se implementa dentro de un sistema autonómico.	Se ha implementado la táctica Pause and Resume dentro del diagrama de despliegue.
R-6	Que el arquitecto no entienda cómo se implanta este patrón de arquitectura.	Se proporciona la documentación necesaria para facilitar el aprendizaje de este nuevo patrón.
R-7	La arquitectura no puede manejar el escenario de desempeño más específicamente el atributo de calidad de escalabilidad.	Se ha decidido implementar el patrón mediador para la distribución de transacciones dentro del componente Meta-S-MDA.

Resultados de la validación

- Se ha obtenido una lista de riesgos que el arquitecto deberá de corregir antes de empezar la implementación del sistema.
- Que el modelo de arquitectura específico puede ser mejorado corrigiendo los riesgos obtenidos a partir de la evaluación.
- El modelo de arquitectura es vulnerable en algunas partes del diseño.
- La arquitectura instanciada a partir de ARTLESS ayudará a validar y a ajustar la arquitectura en función de los riesgos obtenidos, produciendo mejores diseños una vez que se ha ajustado el diseño de la arquitectura en función de los riesgos.

6.7 Discusión y limitaciones

Realizar la evaluación de un meta-modelo de arquitectura es sumamente complicado, no se encontró algún método en la literatura que muestre un proceso para realizar dicha evaluación, y los métodos existentes sirven para evaluar modelos de arquitectura particulares. Desarrollar un método que nos ayude a evaluar meta-modelos de arquitectura sería un trabajo sumamente exhaustivo que no se terminaría en el tiempo estimado para esta tesis, por lo que se tuvieron que buscar opciones para realizar la evaluación del meta-modelo de arquitectura.

Lo primero que se realizó para realizar la evaluación fue crear una instancia del meta-modelo de arquitectura, llevándolo al diseño de un modelo particular, ya que se había realizado el diseño del modelo y se seleccionó un método de evaluación. Se aplicó la evaluación al modelo particular y se obtuvieron una lista de riesgos que se aplicaron de nuevo a la arquitectura, esto se realizó con la finalidad de validar la arquitectura en función de los riesgos encontrados.

También la lista de riesgos se aplicó al meta-modelo de arquitectura, esto es posible dado a que la arquitectura particular de donde se obtuvieron los riesgos contiene un núcleo de características del meta-modelo, validando así el meta-modelo de arquitectura en función de los riesgos.

En particular es sumamente complicado realizar la evaluación si no se cuenta con el procedimiento adecuado.

El método de evaluación ATAM descrito en el Capítulo 3 fue de gran ayuda, cuando se realizó la evaluación de modelo de arquitectura, este proporciona una guía detallada de cada uno de los pasos que se tiene que seguir para realizar la evaluación, el resultado de la evaluación es una lista de riesgos que se tendrán que resolver antes de empezar la construcción del sistema.

Capítulo 7

Trabajo Futuro

Se cuenta actualmente con una primera versión del Meta-modelo de arquitectura **ARTLESS** la cual cumple con los objetivos y requerimientos planteados en la Sección 1.2 y 4.2 de esta tesis; además, durante este ciclo de desarrollo se ha construido una arquitectura lo suficientemente flexible que permite la inclusión de nueva funcionalidad, también permite implementar otros atributos de calidad que propiciaría tener una arquitectura basada en atributos de calidad. Por lo tanto, a continuación, se muestra el trabajo que ha quedado pospuesto para futuros ciclos de desarrollo:

Durante el ciclo de desarrollo de levantamiento de requerimientos del Meta-modelo de arquitectura se tuvieron algunos problemas al pasar de requerimientos y atributos de calidad a escenarios autonómicos, dado ha que no se tiene un método que guíe el proceso de transformar los atributos de calidad a escenarios autonómicos. Por lo tanto, es necesario diseñar un proceso que ayude en la transformación o elaboración de escenarios autonómicos.

Durante la etapa de diseño del Meta-modelo de arquitectura se encontró que no existe metodología o proceso de desarrollo que guíe el diseño de meta-modelos de arquitectura, por lo tanto, como trabajo a futuro puede ser planteado como tesis de maestría o de doctorado al desarrollar una metodología o proceso que guíe el desarrollo de meta-modelos de arquitectura bajo el paradigma de la computación autonómica.

Durante la etapa de evaluación del Meta-modelo de arquitectura no se encontró ningún método que ayudará a la evaluación de Meta-modelos de arquitectura, por lo tanto, queda como trabajo a futuro diseñar un proceso de evaluación de Meta-modelos de arquitectura, más específicamente queda desarrollar un método de evaluación para meta-modelos de

arquitectura bajo el paradigma de computación autonómica de apoyo al diagnóstico médico. Este trabajo puede ser planteado como tesis de maestría.

Como se mencionó al principio del trabajo a futuro, la arquitectura ARTLESS se ha construido lo suficientemente flexible como para agregar nuevos atributos de calidad, pero más específicamente se pueden implementar nuevos meta-componentes que coadyuven a los sistemas de información para el cuidado de la salud a mejorar en los campos que aún son deficientes, como es la seguridad. Aunque en esta tesis se propone un meta-componente enfocado a la seguridad de los sistemas, no se cubren todos los campos de seguridad, siendo así qué se puede complementar este Meta-componente agregando atributos de calidad en dicho campo.

Capítulo 8

Conclusiones

En esta tesis se ha enfatizado la importancia que tiene el diseño arquitectónico para desarrollar sistemas de software complejos, especialmente aquellos que tratan con información médica como los HCIS. Esta importancia radica en diseñar vistas arquitectónicas que den soporte a las necesidades del cliente; en casos de sistemas médicos son: la protección de información de los pacientes, la conexión con otros dispositivos médicos, sistemas siempre activos, sistemas que tengan tolerancia a fallos entre algunas otras necesidades.

En el caso de un meta-modelo de arquitectura la importancia del diseño es todavía más importante, por que a partir del meta-modelo se crea el diseño de varias arquitecturas particulares y si el meta-modelo está mal diseñado, las arquitecturas particulares tendrán defectos.

Para diseñar el meta-modelo de arquitectura **ARTLESS** se planteó una versión adaptada de la metodología; diseño guiado por atributos (ADD). Siendo esta una metodología probada para realizar el diseño de arquitecturas particulares. De esta forma hemos tomado ADD para orientar el diseño del meta-modelo tomando los puntos de interés (Sección 3.2.2.1) para guiar el desarrollo y las actividades que llevaron a diseñar **ARTLESS**.

Los principales objetivos, necesidades, características se seleccionaron basados en una revisión sistematizada de la literatura. El principal aporte del estudio radica en que permite recolectar toda la información relacionada con el tema de interés, mostrando la posibilidad de tomar decisiones justificadas con base en un método, (Estudio de Mapeo Sistemático

(EMS)). En el caso, de proyectos como esta tesis que están basados en nuevas metodologías de diseño de arquitecturas y el uso de patrones de diseño (p.ej. patrón MAPE-K) que están en un proceso de desarrollo fuerte, la ejecución de EMS debe ejecutarse frecuentemente debido a la evolución de los sistemas que se desarrollan bajo el paradigma de la computación autonómica.

Como se mostró en la motivación en el EMS (Capítulo 2) se buscaron trabajos relacionados de HCIS con CA debido al interés que existe en esta tesis de apoyar a los hospitales de proporcionar sistemas con la mínima intervención humana. El presente estudio del arte, realizado bajo la metodología EMS, mostró las principales necesidades de los sistemas de información para el cuidado de la salud. Así también se analizaron trabajos de HCIS con CA y en especial el diseño de arquitectura. Se encontró que existen muy pocos trabajos que involucran a los sistemas HCIS y a la computación autonómica y más específicamente no se encontraron meta-modelos ni modelos de arquitectura que implementen un conjunto de vistas arquitectónicas que ayuden a diseñar arquitecturas para sistemas médicos bajo el paradigma de la computación autonómica. También se encontraron muy pocos trabajos que aborden la seguridad en los sistemas médicos.

A partir de los resultados del estado del arte y de la motivación de esta tesis, se presenta el diseño de un meta-modelo de arquitectura en el cual fue necesario considerar las principales necesidades que tienen los sistemas HCIS como son: seguridad, configuración, reparación e identificación de diagnósticos médicos (Sección 2.4). La implementación de otras necesidades recabadas en el estado del arte queda enunciada como líneas de investigación abiertas para trabajo futuro, por lo tanto, quedarán fuera del alcance de esta tesis.

Los requerimientos obtenidos a partir del estado del arte fueron utilizados para guiar el taller de atributos de calidad (QAW), dado que el estado del arte mostró un escenario general de los sistemas HCIS. Con los objetivos planteados en esta tesis, se enfocó el taller QAW encontrando así los atributos de calidad enfocados a las principales necesidades y escenarios asociados a cada atributo. Recordando que el taller QAW que plantea el SEI, es necesario un equipo, el cual involucra diferentes usuarios relacionados con el diseño del sistema que se va a construir.

Para esta tesis no se contaba con un equipo, por lo que se tuvo que realizar una adaptación del QAW (Sección 3.2.1.4) simulando diferentes usuarios relacionados al diseño de la arquitectura y al sistema. Para dar soporte a este QAW se basó en las principales necesidades de los sistemas HCIS encontrados en el estado del arte (Capítulo 2). Para justificar más precisamente los escenarios que se seleccionaron para el desarrollo de la arquitectura, se desarrolló un diagrama de casos de uso basados en el estado del arte y en la definición de cada uno de los meta-componentes desarrollados para el meta-modelo de arquitectura. Por lo que teniendo el taller de atributos de calidad y el diagrama de casos de uso, se realizó una

asociación (*Tabla 23,25,27 y 29*), entre los escenarios y los casos de uso que tuvieran una relación, obteniendo así los escenarios a ser seleccionados.

Los escenarios seleccionados fueron transformados en escenarios autonómicos utilizando la plantilla SAQAS (Sección 3.2.1.5), los cuales sirvieron como entradas al diseño guiado por atributos que se adaptó al igual que el QAW, ya que ADD recibe como entradas escenarios de atributos de calidad y no escenarios autonómicos. Como hemos planteado en esta tesis, además de que está planteado para realizar el diseño de un modelo de arquitectura particular y no un meta-modelo de arquitectura como el que se presenta en esta tesis, por lo que se ha tenido que adaptar ADD abstrayendo los puntos de interés dando como resultado la metodología planteada en la Sección 3.1.2.3 para el diseño del meta-modelo de arquitectura.

El resultado de ejecutar la versión adaptada de ADD es el diseño del meta-modelo de arquitectura llamado **ARTLESS** el cual está basado en un conjunto de escenarios autonómicos que dan soporte al diseño de los cuatro meta-componentes que conforman la meta-arquitectura. Estos meta-componentes contienen los siguientes atributos de calidad: usabilidad, disponibilidad, compatibilidad, mantenibilidad y seguridad (Ver Apéndice B *Tabla 45* Justificación de la selección de atributos de calidad asociados a un meta-componente), diseñando así una meta-arquitectura bajo atributos de calidad.

El diseño de la meta-arquitectura **ARTLESS** presenta un conjunto de vistas arquitectónicas documentadas bajo el estándar que presenta UML en su versión 3. El conjunto de vistas de alto nivel permite diseñar diagramas específicos utilizados para diseñar arquitecturas particulares. El desarrollo de cada una de las vistas implementa el patrón MAPE-K, el cual es el encargado de producir un diseño autonómico.

El meta-modelo de arquitectura está conformado por cuatro componentes descritos con detalle en el Capítulo 4. Cada uno de estos componentes cumplen un objetivo específico dentro de los sistemas HCIS de apoyo al diagnóstico médico:

- El primer componente, el Meta-S-MDA está dedicado a filtrar el diagnóstico médico de pacientes.
- El segundo componente, el Meta-S-MS, encargado de la seguridad de datos de pacientes y de ataques internos como externos al sistema HCIS, específicamente cubre los campos de confidencialidad, autenticación, e integridad.
- El tercer componente, el Meta-S-MH está dedicado a la curación de la base de datos que contiene el registro médico electrónico y al servidor que contiene el sistema HCIS.
- El cuarto componente, dedicado a la configuración de la base de datos que contiene el registro médico de paciente, a la configuración del servidor que contiene al sistema HCIS y a la configuración de nuevos dispositivos y usuarios dentro de los sistemas HCIS de apoyo al diagnóstico médico.

El Meta-modelo ARTLESS ofrece un conjunto de componentes dedicados a los sistemas HCIS de apoyo al diagnóstico médico.

De acuerdo a la definición de un meta-modelo de arquitectura, el objetivo principal es definir un lenguaje para crear un modelo, es decir, el meta-modelo **ARTLESS** es el lenguaje para crear modelos de arquitectura específicos. Hablando más estrictamente existe una brecha entre un meta-modelo y un modelo específico. Entonces para pasar de un meta-modelo a un modelo específico es necesario utilizar un proceso el cual guíe paso a paso como crear la arquitectura de un sistema específico a partir de **ARTLESS**. Por lo tanto, en el Capítulo 5 de esta tesis se presenta el proceso llamado **STEER**, el cual nos muestra una guía detallada de todos los pasos que se tiene que realizar para crear modelos de arquitectura particulares.

Por lo tanto, la nueva arquitectura generada contiene el núcleo de características del meta-modelo más las características propias del proyecto en cuestión. Para garantizar el correcto funcionamiento de la arquitectura se realizar un proceso de evaluación. A partir de la evaluación obtiene una lista de riesgos, estos los hemos utilizado para dos cosas: 1) validar la arquitectura específica en función de los riesgos. 2) validar el meta-modelo de arquitectura en función de los riesgos. Para el primer caso: se valida el modelo de arquitectura específica en función de los riesgos encontrados. Es decir, se corrige la arquitectura específica, por lo tanto, cuando se realice de nuevo la evaluación de la arquitectura, el resultado sería riesgos aceptados en función de la última evaluación realizada. Para el segundo caso: se valida el meta-modelo en función de los riesgos obtenidos a partir de la evaluación, ya que la arquitectura específica contiene un núcleo de características del meta-modelo. Por tanto, los problemas o riesgos que tiene la arquitectura específica los tiene el meta-modelo debido a que se ha instanciado a partir de el; por lo que al ajustar el meta-modelo de arquitectura en función de los riesgos, se estaría validando también las arquitecturas específicas que fueran realizadas después de ajustar el meta-modelo.

Utilizar meta-modelo **ARTLESS** para desarrollar arquitecturas partículas HCIS de apoyo al diagnóstico médico fomenta el desarrollo de sistemas inteligentes, debido a que en él se implementan las funciones inteligentes propuestas por IBM enfocadas a los sistemas médicos, además de que en esta tesis se a desarrollo un nuevo componente inteligente enfocado a filtrar el diagnóstico médico de pacientes. Este trabajo también presenta algunas limitaciones: Primero para utilizar el meta-modelo en proyectos reales de software el arquitecto deberá tener un alto grado de conocimiento en meta-modelado, tecnologías del SEI como QAW, ADD y conocimientos en CA. Otra limitación será el esfuerzo en horas que dedique el arquitecto para aprender a utilizar el meta-modelo pero este se verá compensado, debido a que podrá generar varias arquitecturas que incorporen características autonomías.

Apéndice A: Sensor, Base de Datos y Conexión del sensor con la BD

Sensor

El sensor que presenta la meta-arquitectura debe de realizar las siguientes funciones dentro de la base de datos:

Detectar que ha habido cambio en la base de datos, principalmente: 1) se ha realizado una inserción de un nuevo paciente a la BD; 2) se ha realizado la actualización de un paciente en la BD; 3) se ha realizado la solicitud de información de pacientes; 4) se ha eliminado la información de un de paciente. Si bien, es sabido que un trigger o disparador puede detectar estas funciones, debido a que este se ejecuta cuando sucede algún evento sobre las tablas a las que se encuentra asociado. Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla. Todos estos eventos son guardados en una tabla dentro de la base de datos. Los triggers no cumplen, con el objetivo de: 1) procedimiento se realice en tiempo real; 2) enviar notificaciones para lanzar el ciclo autonómico. Por lo que es necesario implementar un sensor en la base de datos.

Base de Datos

Si bien, las bases de datos presentan varias características que ayudan a realizar las transacciones de forma segura como son: 1) atomicidad, es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias. Se dice que una operación es atómica cuando es imposible para otra parte de un sistema encontrar pasos intermedios. Si esta operación consiste en una serie de pasos, todos ellos ocurren o ninguno; 2) consistencia, es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto, se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos; 3) durabilidad, es la propiedad que asegura que, una vez realizada la operación, ésta persistirá y no se podrá deshacer, aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera; 4) aislamiento, es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sea independiente y no generen ningún tipo de error. También es necesario definir las características de conexión.

Requerimientos básicos de una base de datos para establecer una conexión:

- Tipo de base de datos: EL tipo de base de datos que contiene los metadatos sobre la configuración del dominio.
- Host de la base de datos: El nombre del equipo donde se aloja la base de datos.
- Puerto de base de datos: El número de puerto empleado por la base de datos.

- Nombre de la base de datos: El nombre de la base de datos.
- Usuario de la base de datos: La cuenta de usuario para la base de datos que contiene la información sobre la configuración del dominio.
- Seguridad de la capa de transporte: Este describe cómo habilitar conexiones cifradas para una instancia de Motor de base de datos mediante la especificación de un certificado para el Motor de base de datos utilizando el administrador de configuración.

Requerimientos específicos de conexión del sensor con la base de datos.

Tener las credenciales de la base de datos.

- Una credencial es un registro que contiene la información de autenticación (credenciales) necesaria para conectarse a un recurso situado en una base de datos. Las credenciales incluyen un nombre de usuario y una contraseña.

- Para proporcionar mayor seguridad será necesario crea una credencial específica para realizar la conexión del sensor con la base de datos. Esto con el objetivo de tener acceso a los eventos que sucedan en la base de datos en tiempo real.

- Conocimiento sobre las consultas que se van a monitorear. Por ejemplo, eventos de INSERT o DELETE.

Apéndice B: Caso de estudio

Para realizar la evaluación del meta-modelo de arquitectura **ARTLESS** se ha realizado la siguiente propuesta.

En base a que no se tiene lineamientos o un estándar para la evaluación de meta-modelos y desarrollar y proponer dichos lineamientos sería un trabajo sumamente exhaustivo, se ha propuesto crear una instancia particular de **ARTLESS** y a partir de esta, aplicar la evaluación de la arquitectura a la instancia resultante de la instanciación.

Por lo anterior mencionado se ha decidido realizar la simulación de un caso de estudio de un hospital en México. Con base al análisis del estado del arte, se recolectaron algunos de los principales problemas que afrontan los hospitales, como son el manejo de grandes cantidades de información, la heterogeneidad de la información, la seguridad de la información que se maneja, entre otros. Basándonos en esta información se decidió crear una arquitectura que dé soporte para crear un sistema que afronte este tipo de problemas.

Para la construcción de la arquitectura se siguió la siguiente metodología.

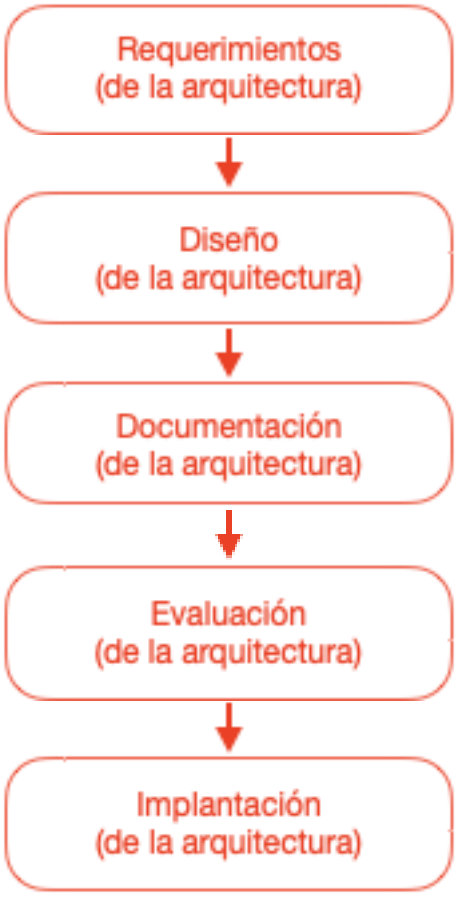

Figura 52. Ciclo de vida del diseño de arquitectura.

Requerimientos de la arquitectura.

Esta etapa se enfoca en la captura, documentación y priorización de requerimientos que influyen sobre la arquitectura y que, por lo habitual, se conocen en inglés como drivers arquitectónicos. Los atributos de calidad juegan un rol preponderante respecto de los requerimientos, así que esta etapa hace énfasis en ellos. Otros requerimientos, como los casos de uso y las restricciones, son también relevantes para la arquitectura.

Diseño de la arquitectura.

El diseño es la especificación de un objeto, creado por algún agente, que busca alcanzar ciertos objetivos en un entorno particular, usando un conjunto de componentes básicos, satisfaciendo un conjunto de requerimientos y sujetándose a ciertas restricciones.

Documentación de la arquitectura.

El diseño necesita ser comunicado de forma adecuada. Así abordamos aspectos de comunicación del diseño y más particularmente en la actividad de documentación de la arquitectura.

La documentación del diseño de la arquitectura de software cumple varios propósitos importantes. Entre ellos están: • Comunicar a los distintos involucrados el diseño y las decisiones que permitieron llegar a éste. • Permitir realizar análisis y evaluación del diseño. • Soportar las actividades de mantenimiento.

Evaluación de la arquitectura

La arquitectura es un aspecto tan importante dentro del desarrollo que es conveniente realizar actividades de verificación de forma temprana, con el fin de identificar problemas que podría resultar muy costoso eliminar posteriormente. La evaluación de la arquitectura de software permite justamente realizar la verificación del diseño.

Requerimientos de la arquitectura para un sistema HCIS

Para definir el alcance del proyecto se ha realizado mediante el documento de ***Visión y Alcance***, el cual define el alcance y los objetivos de alto nivel de un proyecto de software. Una declaración clara del problema, una propuesta de solución y las características de alto nivel de un producto que ayudan a establecer expectativas y reducir riesgos. Dentro del documento de Visión también se definen las necesidades y requerimientos del sistema.

Posteriormente se ha realizado una versión del *QAW*, en este caso, no se cuenta con el personal suficiente para realizar una versión completa del QAW, por lo que se ha realizado una simulación suponiendo que se tienen los siguientes stakeholders: desarrollador, técnico, administrador, operaciones, product owner.

Por lo tanto, se han definido los alcances del proyecto y se han cumplido las pre-condiciones necesarias para realizar una instancia de la meta-modelo **ARTLESS**.

Tabla 43. Pre-condiciones del sistema HCIS.

Pre-condiciones del sistema	Satisfecho	No Satisfecho
Definir el alcance del proyecto	X	-
Definir los requerimientos del sistema	X	-
Definir los atributos de calidad	X	-

Diseño de la arquitectura para un sistema HCIS

Como siguiente punto se realiza un análisis de las self-features propuesta, con la finalidad de ver si las necesidades del negocio se pueden resolver implementando alguno de los meta-componentes de **ARTLESS**.

Para esta etapa del diseño se ha presentado el proceso **STEER** el cual está compuesto por dos actividades (Figura 54), las cuales nos ayudaran a seguir los pasos necesarios para crear instancias de la arquitectura ARTLESS.

Como primer paso tenemos la Act-1, la cual nos ayuda a realizar la selección de la self-feature más adecuada para nuestro problema.

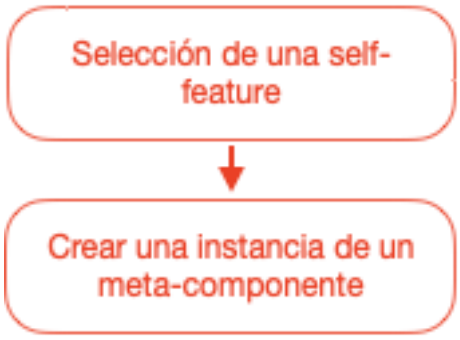

Figura 53. Actividades del proceso STEER.

Parte 1: Selección de una self-feature

Tarea 1.- Analizar los escenarios en los que se puede realizar una instanciación de las self-features.
La *Tabla 44* muestra todos los escenarios que cubre la arquitectura **ARTLESS** y los cuales se han analizado para realizar la selección de los meta-componentes a implementar.

Tabla 44. Escenarios que cubre la arquitectura ARTLESS

Self-feature	Escenarios
Meta-Self-MedicalSecurity	Un usuario intenta acceder a la base de datos del hospital, pero ha olvidado su contraseña de acceso. El sistema solo permite realizar tres intentos para acceder, después de realizar los tres intentos el sistema verifica que sea un usuario con acceso, en caso de que se verifique correcto el sistema envía un formulario de recuperación de contraseña, en caso contrario bloquea al usuario.

	Un usuario malintencionado rompe la seguridad del sistema HCIS y se conecta a la red del hospital para obtener información de los pacientes. El sistema detecta que ha ingresado un usuario no identificado y procede a bloquearlo de inmediato, el sistema generando un reporte de suceso ocurrido.
	Un usuario que es nuevo dentro del sistema HCIS desea obtener la información de más de tres pacientes al mismo tiempo. El sistema en respuesta envía una notificación de que no se puede visualizar la información de más de tres pacientes. Al mismo tiempo, el usuario hace caso omiso de la notificación y desea ver la información de más de tres pacientes, por lo que el sistema bloquea 1 minuto al usuario, y en caso de que desee acceder de nuevo durante los siguientes 5 minutos, el usuario es bloqueado por un periodo de 10 minutos.
	Un usuario malintencionado rompe la seguridad del sistema HCIS y realiza copias de la información de los pacientes. El sistema detecta que se ha roto la seguridad y bloquea el acceso a la base de datos, mientras detecta las direcciones IP que han accedido a la base de datos e identifica cuales tienen acceso a esta.
	Un usuario malintencionado se autentica al sistema y ha ingresado a la base de datos en funcionamiento. El usuario modifica datos de pacientes para fines convenientes, el sistema ha detectado cambios en la base de datos, el sistema bloquea al usuario y revisa la información modificada con la base de datos espejo.
Meta-Self-MedicalConfiguration	En un momento de operación se han realizado varias consultas a la base de datos, esta no ha respondido en el tiempo esperado y se ha caído, el sistema ha detectado que la base de datos ha fallado y ha notificado a la base de datos espejo para que se configure y se ponga en funcionamiento.
Meta-Self-MedicalHealing	Un usuario con acceso al sistema ha ingresado a la base de datos y la ha dañado. El sistema detecta que la base de datos no responde a las peticiones y la ponen fuera de servicio. El sistema configura la base de datos espejo para que el sistema funcione con normalidad mientras se repara la base de datos dañada. Una vez que se ha reparado se configura y se ponen en funcionamiento.
	Un usuario que se ha autenticado al sistema ha ingresado a la base de datos y ha borrado una parte intencionalmente, el sistema detecta esta situación y bloquea al usuario inmediatamente. El sistema verifica la información borrada con la base de datos espejo, copia la información eliminada y se ponen en funcionamiento nuevamente.

Hasta este momento hemos analizado cada uno de los escenarios propuestos por **ARTLESS**, observando la problemática que se presenta y cómo se pretende desarrollar la solución.

Tarea 2.- Analizar los atributos de calidad que cubre cada self-feature. La *Tabla 45* muestra qué atributos de calidad están asociados a cada meta-componente.

Tabla 45. Mapeo de Meta-componentes a atributos de calidad.

Meta-Component e	**Atributo de calidad**	**Sub-categoría (s)**	**Justificación**	
Meta-Self-MedicalDiagnosisAnalysis	Usabilidad	Efectividad	Se ha seleccionado el atributo de calidad de efectividad ya que el meta Meta-Self-MDA está enfocado en realizar una tarea que es: filtrar pacientes respecto a su diagnóstico médico. Siendo así que este meta-componente está dedicado a realizar una tarea en específico.	
	Disponibilidad	Confiabilidad	Funcionalidad	Es el componente principal para identificar nuevos pacientes y filtrarlos respecto a su diagnóstico médico. Este componente tiene que estar disponible en todo momento, ya que los pacientes llegan a los hospitales las 24 horas del día. Por otra parte, el Meta-Self-MDA debe ser confiable y evitar posibles fallas, ya que si llegase a fallar, no se filtraría ningún paciente y habría una acumulación de pacientes en espera de ser filtrados, generando así pérdidas de tiempo y dinero para los pacientes.
	Compatibilidad	Interoperabilidad	El Meta-Self-MDA debe implementar interoperabilidad ya que se debe de comunicar con otros sistemas, por ejemplo cuando un nuevo paciente llega a un hospital se realiza un mini proceso para realizar la persistencia del paciente, primero se ingresan sus datos personales (nombre, edad, peso, NSS, etc), después se ingresa con el médico general, el cual es el encargado de otorgar un diagnóstico y persistir la información del paciente, esta información que se ha guardado en la base de datos es la que ocupará el Meta-Self-MDA y si la información se ha persistido bajo un estándar médico y el Meta-Self-MDA no implementa ese estándar no se podrá realizar el objetivo para el cual fue diseñado.	

Meta-Self-MedicalConfiguration	Usabilidad	Eficiencia	El Meta-Self-MC implementa una sub-categoría de usabilidad la cual es eficiencia. La razón de implementar este atributo de calidad es brindar al usuario final un mejor sistema. Por ejemplo, supongamos que la base de datos donde se encuentran los registros médicos de los pacientes ha sufrido un error, por lo que debe de configurarse la base de datos que se encuentra en standby. Esta deberá configurarse en el menor número de transacciones y el menor tiempo posible para que no se vea afectado el sistema.
	Mantenibilidad	Modificabilidad	El objetivo del Meta-Self-MC es hacer cambios en el ambiente, por lo que si se desea realizar un cambio tanto en arquitectura como en el sistema tiene que ser fácil de realizar y en el menor tiempo posible, por lo tanto, es importante que el Meta-Self-MC implemente el atributo de calidad de mantenibilidad y más específicamente que pueda implementar modificabilidad para que a la hora de realizar cambios no se introduzcan errores ni degradación del sistema o arquitectura.

Meta-Self-MedicalSecurity	Seguridad	Confidencialidad	En un hospital la protección de la información de los pacientes es muy importante ya que la filtración de la información se podría utilizar para fines perjudiciales a los pacientes. Por lo que es importante que la información médica de los pacientes está protegida contra la divulgación no autorizada.
		Autenticación	El Meta-Self-MS debe implementar autenticación ya que en un hospital se manejan grandes cantidades de información y no todos los usuarios deben de tener acceso a esta. Supongamos que un usuario malintencionado intenta ingresar al sistema, este usuario primero deberá autenticarse y en caso de que el acceso sea denegado el usuario será bloqueado de inmediato.
		Integridad	El Meta-Self-MS debe implementar integridad ya que los datos de los pacientes no deben ser modificados al menos que sea por el personal autorizado.
	Disponibilidad	Confiabilidad	El Meta-Self-MS es el encargado de proteger el sistema de posibles ataques internos como externos, por lo que este componente no debe fallar, ya que en el momento que este componente falle todo el sistema queda vulnerable a posibles ataques es por esta razón por la que el Meta-Self-MS debe implementar el atributo de seguridad de confiabilidad.
	Usabilidad	Manejo de errores	El Meta-Self-MS debe realizar el manejo de errores debido a que, si en un momento de operación el

			sistema HCIS de apoyo al diagnóstico médico sufre un error, el sistema quedará vulnerable a ataques.	

Meta-Self-MedicalHealing	Disponibilidad	Confiabilidad	Tolerancia a fallos	El objetivo del Meta-Self-MH es detectar, diagnosticar y reparar automáticamente el mal funcionamiento por lo que si un componente falla lo tiene que reparar, pero si el Meta-Self-MH falla no se puede auto-reparar solo, por lo que necesita implementar el atributo de calidad de tolerancia a fallas mediante la implementación de un patrón o táctica.
			Recuperabilidad	El Meta-Self-MH debe implementar recuperabilidad ya que la información de los pacientes es muy importante, por ejemplo, supongamos que la base de datos donde se encuentran los registros médicos de los pacientes se ha dañado y se ha perdido gran cantidad de información, esto sería un gran problema ya que el hospital podría enfrentarse a demandas por parte de los pacientes, por lo que es importante que tenga un método de recuperación de información.
	Mantenibilidad	Modificabilidad	Riesgo de daño	El Meta-Self-MH que es un componente enfocado a reparar fallas en el sistema es importante que implemente mantenibilidad más específicamente en el contexto de riesgo de daño ya que sin componente llegase a fallar ya se tendría contemplada esa falla y se podría reparar más fácilmente o dar mantenimiento antes de que fallase.
	Usabilidad	Manejo de errores	Es importante considerar que el Meta-Self-MH implemente el atributo de calidad de usabilidad más específicamente la sub-categoría de manejo de errores, ya que los usuarios podrían generar errores bajo ciertas circunstancias de uso del sistema	

Basándonos en las necesidades del negocio y al análisis de escenarios y atributos de calidad se han seleccionado los meta-componentes a implementar. La *Tabla 46* muestra la relación de necesidades del negocio con los meta-componentes.

Tabla 46. Necesidades del negocio a meta-componentes.

ID	**Necesidades del negocio**		**Meta-Componente**
Caso de uso	**Nombre de caso de uso**		**-**
CU-4	Filtrar a los pacientes de acuerdo al diagnóstico médico.		Meta-Self-MedicalDiagnosisAnalysis
CU-9	Conexión con otros dispositivos médicos.		Meta-Self-MedicalConfiguration
CU-14	Detectar una falla		Meta-Self-MedicalHealing
Restricciones	**Tipo de restricción**	**Restricción**	-
RT-3	Técnica	Desarrollo con el patrón MAPE-K	-
Atributos de calidad	**Categoría de atributo de calidad**	**Escenario**	
QA06	Desempeño	En un momento determinado se produce una falla en la base de datos, por lo que las peticiones realizadas no son atendidas. El sistema detecta esta situación y configura la base de datos espejo (standby) en a lo más de 8 pasos, el sistema retoma sus actividades normales en menos de 5 minutos.	Meta-Self-MedicalConfiguration
QA11	Confiabilidad	En un momento de operación normal ha fallado la base de datos y ha entrado la base de datos espejo que ha fallado también. El sistema detecta, diagnostica y repara la falla en a lo más 1 hora, con pérdida de información de a lo más 1 hora en la base de datos.	Meta-Self-MedicalHealing
QA08	Desempeño	En un momento de operación normal ha tenido una falla el servidor que contiene la información del sistema HCIS, el sistema retoma sus operaciones normales en un tiempo de a lo más 10 minutos.	Meta-Self-MedicalHealing

QA07	Seguridad	No se puede obtener o intercambiar información sensible de pacientes, toda esta almacenada y se transfiere de forma cifrada.	Meta-Self-MedicalSecurity

Parte 2: Crear una instancia de un meta-componente

En este sistema que se está construyendo se tiene previsto que se implementen dos de los cuatro meta-componentes propuestos en la arquitectura **ARTLESS**.

Siguiendo los pasos recomendados por el arquitecto de meta-modelado, se ha decidido instanciar primero el Meta-S-MDA.

Siguiendo el proceso STEER para instanciar el Meta-S-MDA

1. Realizar las pre-condiciones:

Hasta este punto se han realizado las pre-condiciones necesarias para realizar una instancia de **ARTLESS**.

2. Revisar los escenarios que cubre el meta-componente a implementar.

Escenario
Filtrar pacientes respecto a su diagnóstico médico

3. Analizar el diagrama de casos de uso del meta-componente asociado.

4. Se ha realizado el análisis de los casos de uso del Meta-S-MDA y se ha decidido implementar cada uno de los usos ya que son fundamentales para el funcionamiento del sistema.

5. Realizar una copia del diagrama de casos de uso, incluyendo los casos de uso a implementar

En este caso el Meta-S-MDA es una propuesta del arquitecto de meta-modelado, se ha decidido dejar el componente original sin agregar ninguna modificación ya que es una aportación a los sistemas médico.

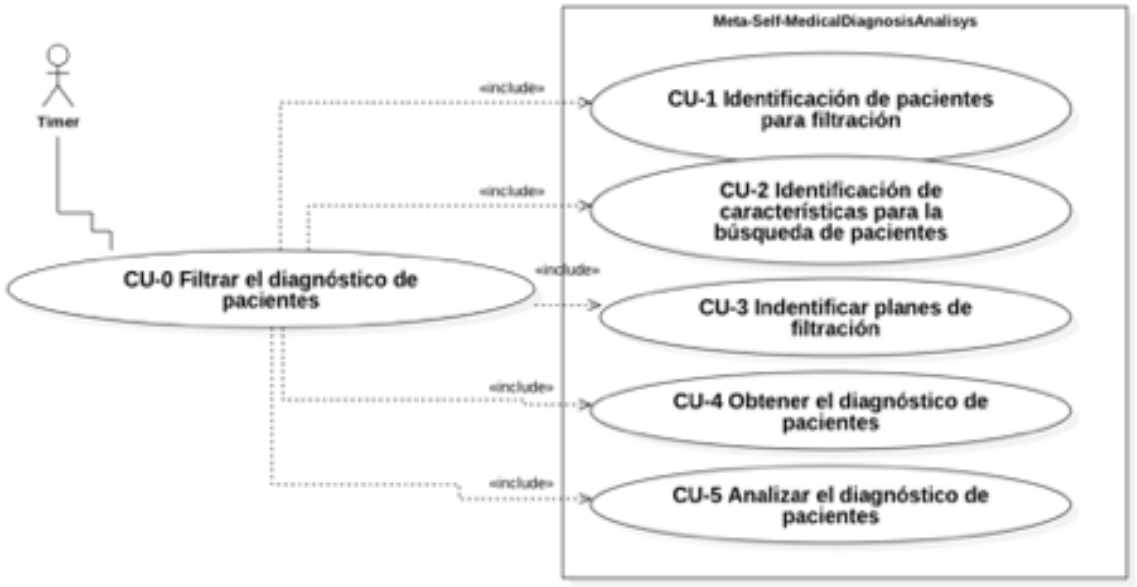

Figura 54. Diagrama de casos de uso del Meta-S-MDA.

6. Analizar la plantilla SAQAS del meta-componente

Mediante el análisis realizado a la plantilla SAQAS, se obtuvo que esta plantilla específica cómo implementar una solución autonómica mediante la interacción de sus distintos elementos. La *Tabla* 47 representa la plantilla del Meta-S-MDA analizada.

7. Realizar una copia de la plantilla SAQAS del meta-componente seleccionado.

Tabla 47. Plantilla SAQAS del Meta-S-MDA.

Source	Médico	
Stimulus	Filtrar el diagnóstico de pacientes	
Artifact	EMR (Electronic Medical Record)	
Environment	Tiempo de ejecución	
Response	Sensor	Detecta que ha habido un cambio en EMR-Data-Base y envía una notificación al Monitor.
	Monitor	Cuando recibe la notificación del Sensor lanza el ciclo autonómico. Verifica todos los accesos al EMR-Data-Base desde su última fecha de acceso obteniendo los nuevos pacientes por su idPatient. Envía una lista de pacientes al componente analizar.
	Analyzer	Recibe la lista. Recuperar toda la información de los pacientes y sintetizarla generando un reporte de cada paciente. Envía la información al componente Planner
	Planner	

		Selecciona el plan adecuado para realizar la filtración de pacientes de acuerdo al análisis recibido. Envía el plan al Executor.
	Executor	Ejecuta el plan
	Knowledge	Guarda los planes del Planner y los reportes generados por el Analyzer

8. Analizar el diagrama de componentes del meta-componente a implementar.
En este paso se ha realizado el análisis del Meta-S-MDA, que está construido mediante una arquitectura de capas e implementa una solución autonómica mediante la integración del patrón MAPE-K.

La Figura 4 representa el Meta-S-MDA.

9. Crear una copia del diagrama de componentes.
El diagrama de componentes representa una vista de la solución del problema (Ver Figura 55).

10. Analizar el diagrama de clases del meta-componente seleccionado.
En este paso se ha realizado un análisis del diagrama de clases del Meta-S-MDA, primero se han visto las clases y los métodos que contienen estas y la relación que se tiene con las diferentes clases.

11. Crear una copia del diagrama de clases.
El diagrama de clases representa una vista lógica de la solución del problema (Ver Figura 56).

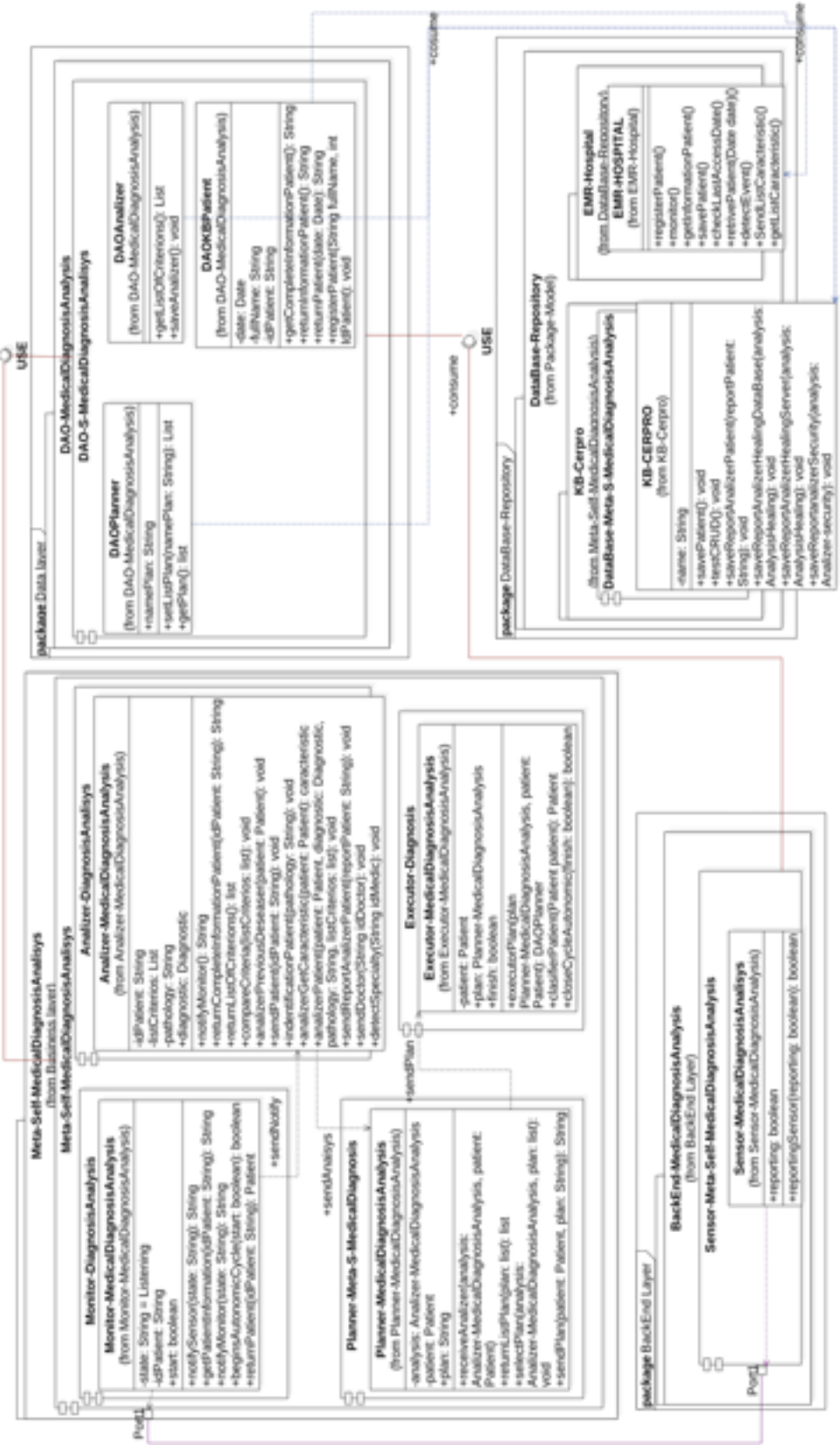

Figura 55. Diagrama de componentes del Meta-S-MDA.

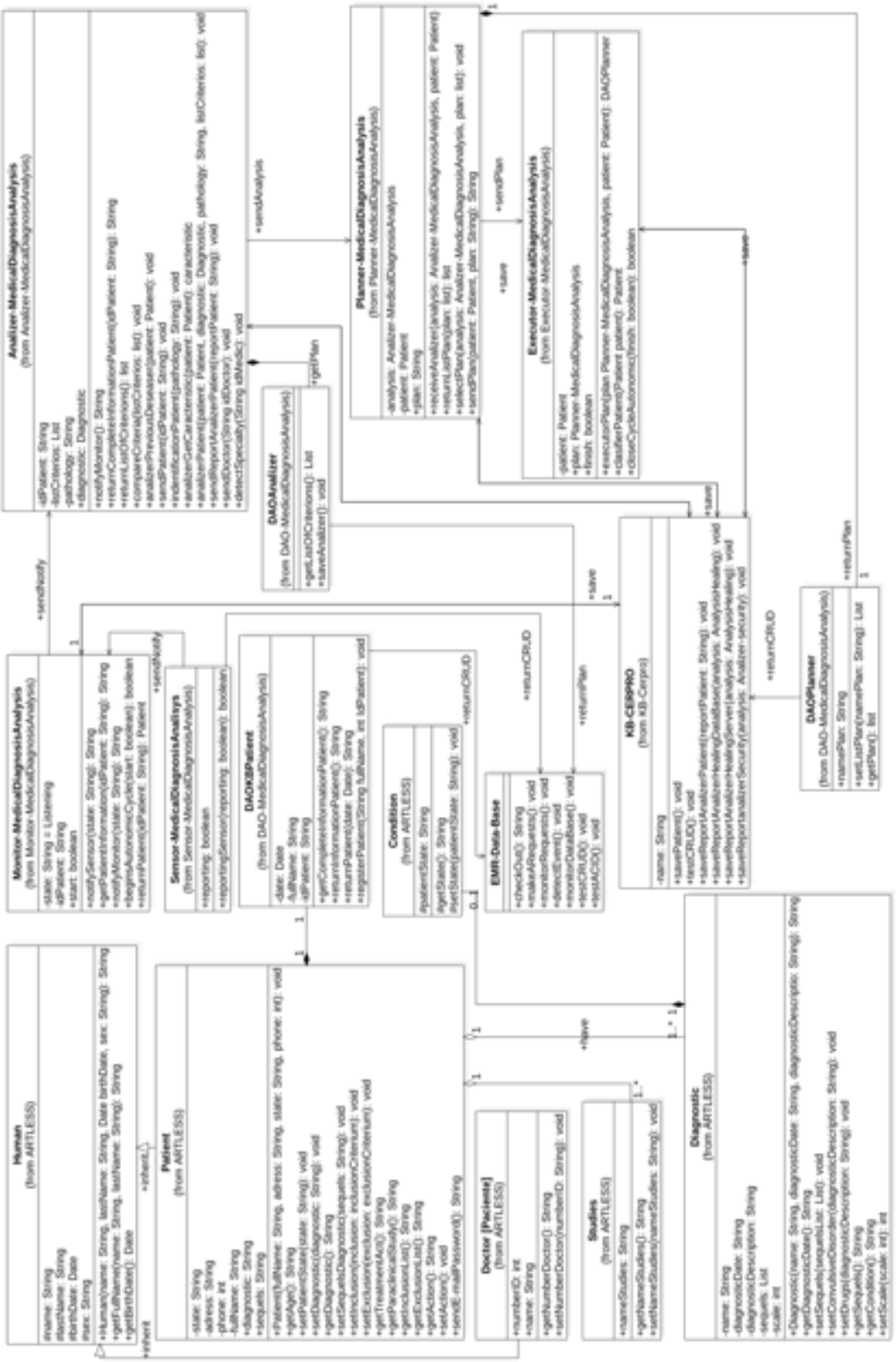

Figura 56. Diagrama de clases del Meta-S-MDA.

12. Analizar los diagramas de secuencia asociados al meta-componente seleccionado

Se han analizado los diagramas de casos de uso, ya que es importante que el desarrollador comprenda cómo se está implementando la solución autonómica al problema de filtrar el diagnóstico de pacientes.

13. Crear copias de los diagramas de secuencia.
Para ejemplificar el diagrama de secuencia implementando computación autonómica solo se mostrará un diagrama (Ver Figura 57).

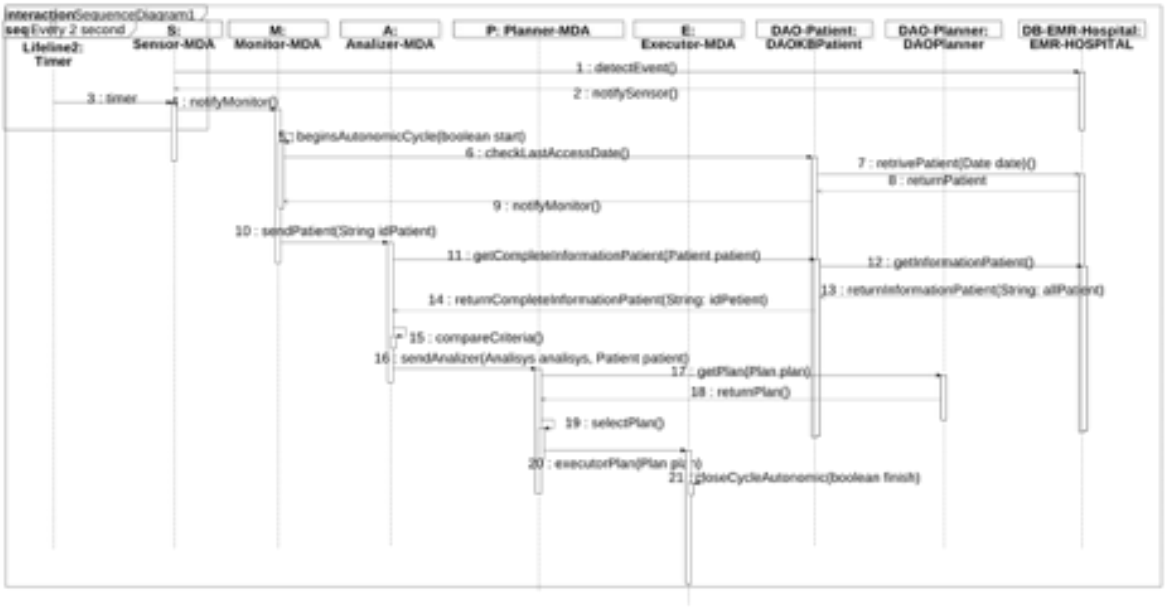

Figura 57. Diagrama de secuencia del Meta-S-MDA

Proceso STEER para instanciar el Meta-S-MC

Siguiendo los pasos recomendados por el arquitecto de meta-modelado, ahora se instancia el Meta-S-MC, el cual da solución a la problemática de configurar diferentes dispositivos de manera autónoma.

1. El primer paso ya se ha realizado cuando se realizó la instancia del Meta-S-MDA.

2. Revisar los escenarios que cubre el meta-componente a implementar.

Escenario
En un momento de operación se han realizado varias consultas a la base de datos, esta no ha respondido en el tiempo esperado y se ha caído, el sistema ha detectado que la base de datos ha fallado y ha notificado a la base de datos espejo para que se configure y se ponga en funcionamiento.

3. Analizar el diagrama de casos de uso del meta-componente asociado

Se ha realizado el análisis de los casos de uso del Meta-S-MC y se ha decidido implementar cada uno de los usos ya que son fundamentales para el funcionamiento de configurar la base de datos y el sistema HCIS, y se ha decidido agregar un nuevo caso de uso.

4. Realizar una copia del diagrama de casos de uso, incluyendo los casos de uso a implementar

Se ha realizado la copia del diagrama de casos de uso y se ha agregado el caso de uso 2, el cual servirá para configurar nuevos dispositivos dentro de un sistema HCIS. Como se muestra en la Figura 58.

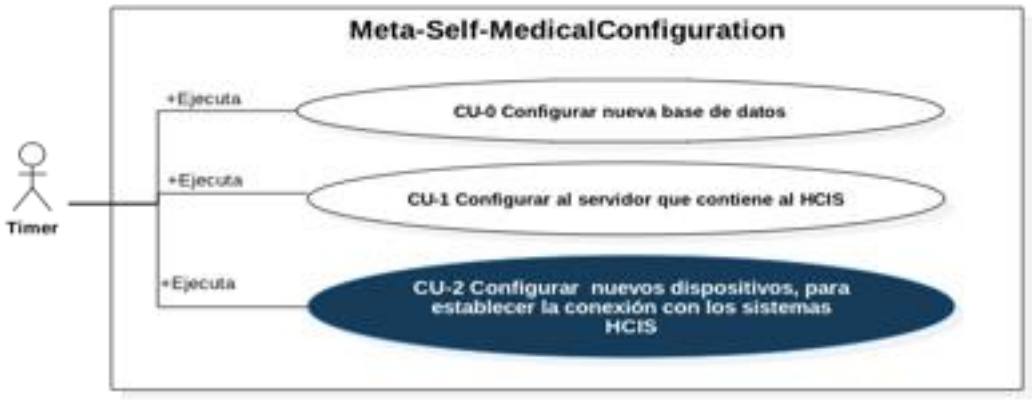

Figura 58- Diagrama de casos de uso.

5. Analizar la plantilla SAQAS del meta-componente.

Mediante el análisis realizado a la plantilla SAQAS, se observa que esta plantilla específica cómo configurar una base de datos mediante la implementación de computación autonómica.

6. Realizar una copia de la plantilla SAQAS del meta-componente seleccionado

Tabla 48. Escenario autonómico del M-S-MC.

Source	Usuario	
Stimulus	En un momento de operación se han realizado varias consultas a la base de datos, esta no ha respondido en el tiempo esperado y se ha caído, el sistema ha detectado que la base de datos ha fallado y ha notificado a la base de datos espejo para que se configure y se ponga en funcionamiento.	
	Después de más de dos años de uso ininterrumpido, el servidor que contiene el software del HCIS ha fallado.	
	En un momento de operación normal se ha detectado un nuevo sistema (sistema PACKS), que se desea conectar con el sistema HCIS.	
Artifact	EMR, Servidor	
Environment	Tiempo de ejecución	

Response	Sensor	Detecta cambios en el ambiente y envía notificaciones al Monitor cuando ha sucedido un cambio.
	Monitor	Cuando recibe notificación del Sensor, se lanza el ciclo autonómico. Monitorea a la base de datos del EMR y al servidor que contiene el software del HCIS. Monitor detecta que un nuevo sistema desea tener acceso al sistema HCIS.
	Analyzer	**Caso 1:** Recibe notificaciones del componente Monitor y analiza la situación detectada por el Monitor. En el caso de fallo de la base de datos, el Analyzer realiza peticiones para cerciorarse del fallo de la base de datos. Con base en el análisis realizado genera un reporte que envía al Planner. **Caso 2**: Analiza las peticiones realizadas al servidor y detecta que el servidor no responde Analiza y detecta las fallas por las cuales el servidor no responde (genera estadísticas). **Caso 3:** Analyzer analiza el tipo de sistema que se desea ser configurado y desea conectarse con el sistema HCIS de apoyo al diagnóstico médico. Por ejemplo: el analyzer a detectado que el sistema que desea tener conexión con el sistema HCIS dé apoyo al diagnóstico médico es el sistema de manejo de imágenes médicas.
	Planner	**Caso 1:** Recibe el reporte del componente Analyzer y basándonos en este se selecciona el plan más adecuado. Para el Caso 1, se configura la base de datos que se encuentra en suspensión y se reactivan los servicios. **Caso 2:** Se configura el servidor que se encuentra en suspensión temporalmente, para que se reactiven los servicios. **Caso 3:** Habilitar un puerto y una interfaz para configurar el nuevo sistema.
	Executor	Ejecuta el plan.
	Knowledge	Guarda los planes del Planner y las estadísticas generadas por el Analyzer para su posterior aprendizaje.
Self-configuration	Los sistemas autónomos deben ser capaces de instalar, configurar y fusionar los componentes automáticamente y de manera uniforme de los sistemas regidos por políticas.	

7. Analizar el diagrama de componentes del meta-componente a implementar.

En este paso se ha realizado el análisis del Meta-S-C, que está construido mediante una arquitectura de capas e implementa una solución autonómica mediante la integración del patrón MAPE-K. La Figura 59 representa el Meta-S-MC.

8. Crear una copia del diagrama de componentes.
El diagrama de componentes representa una vista de la solución del problema.

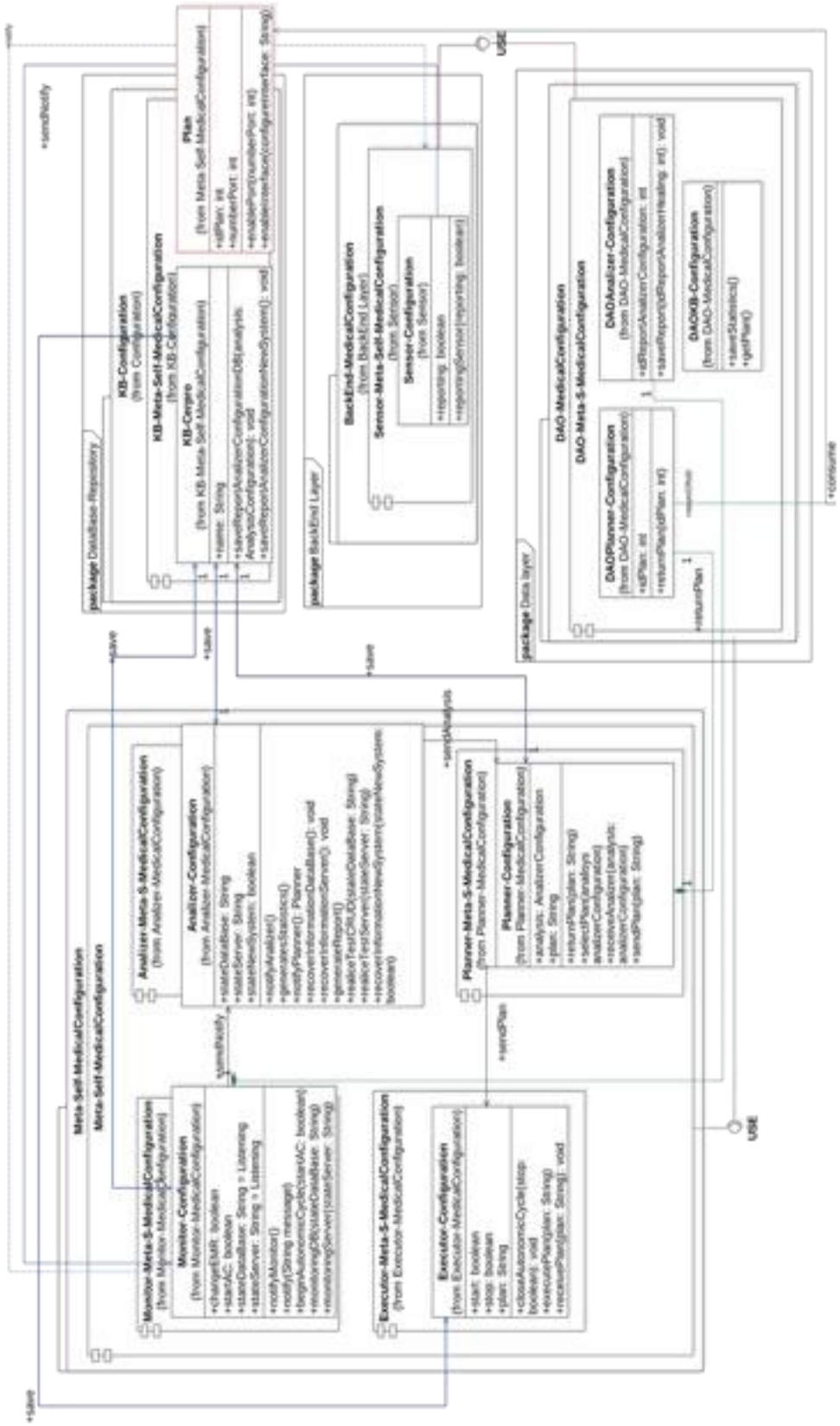

Figura 59- Diagrama de componentes del Meta-Self-MedicalConfiguration.

9. Analizar el diagrama de clases del meta-componente seleccionado.

En este paso se ha realizado un análisis del diagrama de clases del Meta-S-MC, primero se han visto las clases y los métodos que contienen estas y la relación que se tiene con las diferentes clases.

10. Crear una copia del diagrama de clases.
El diagrama de clases representa una vista lógica de la solución del problema, en el siguiente diagrama de clases: se han implementado métodos y clases que dan soporte al CU seleccionado de la Figura 60.

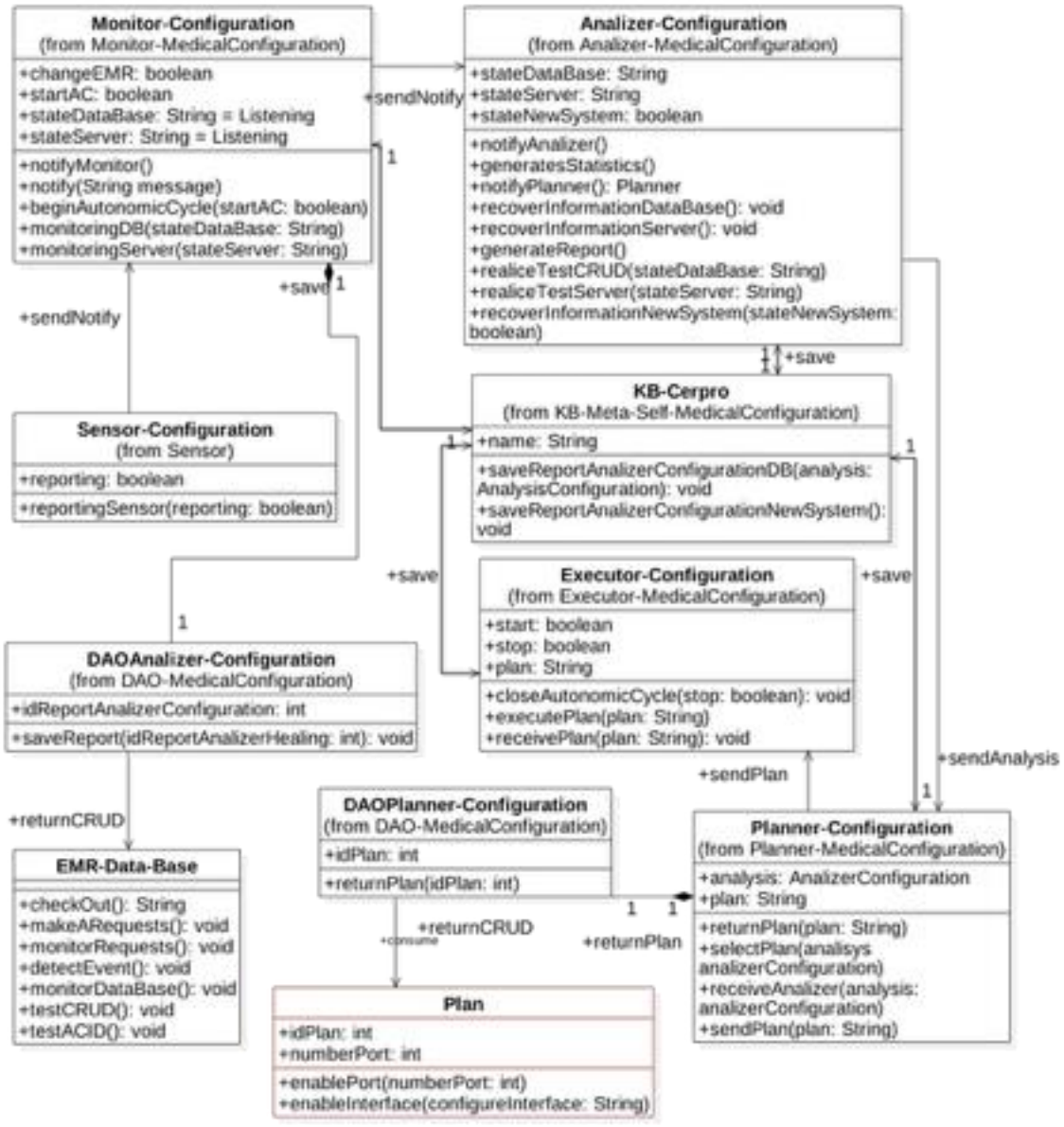

Figura 60- Diagrama de clases del Meta-S-MC.

11. Analizar los diagramas de secuencia asociados al meta-componente seleccionado

Se han analizado los diagramas de casos de uso, ya que es importante que el desarrollador comprenda cómo se está implementando la solución autonómica al problema de filtrar el diagnóstico de pacientes.

12. Crear copias de los diagramas de secuencia.
Para ejemplificar el diagrama de secuencia implementando computación autonómica soló se mostrará un diagrama (ver Figura 61).

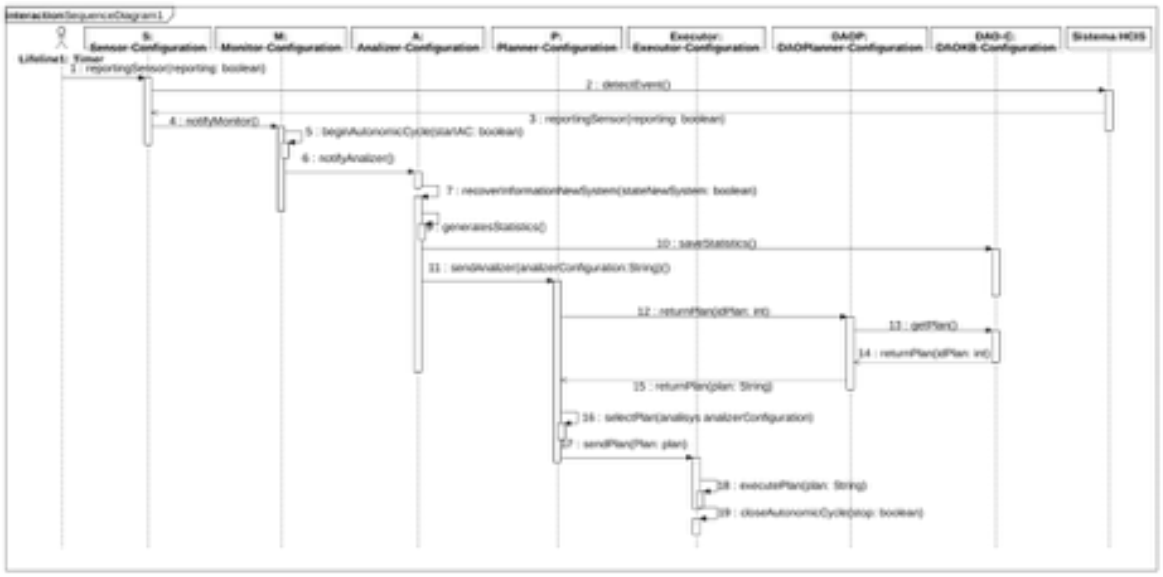

Figura 61. Diagrama de secuencia del Meta-SMC.

Referencias:

[0] Horn, P. (2001). Autonomic Computing: IBM's Perspective on the State of Information Technology. http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf.

[1] Article, Mezghani, E.; Exposito, E. & Drira, K. A Model-Driven Methodology for the Design of Autonomic and Cognitive IoT-Based Systems: Application to Healthcare IEEE Transactions on Emerging Topics in Computational Intelligence, 2017, 1, 224-234.

[2] Article, Lasierra, N.; Alesanco, A.; O'Dullivan, D. & GarcÃ-a, J.An autonomic ontology-based approach to manage information in home-based scenarios: From theory to practice.Data and Knowledge Engineering, 2013, 87, 185-205.

[3] InProceedings, Chen, Q.; Lambright, J. & Abdelwahed, S.Towards Autonomic Security Management of Healthcare Information Systems, 2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016, 113-118.

[4] Conference, Salih, N. & Zang, T. Autonomic and Cloud computing: Management services for healthcare. ISIEA 2012 - 2012 IEEE Symposium on Industrial Electronics and Applications, 2012, 23-28.

[5] Conference, Omar, W.; Amin, S. & Taleb-Bendiab, A. Autonomic model for managing complex healthcare applications. Proceedings - Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems, EASe 2007, 2007, 94-98.

[6] InProceedings, Chandrasekharan, K.; Rao, A. & Sruthi, B. Autonomic Computing in Medical Informatics: Accessing and Retrieval of EMR. 2009 International Conference on Computer Engineering and Technology, 2009, 2, 412-416.

[7] Conference,Zhu, Y.; Sloman, M.; Lupu, E. & Sye, L. Vesta: A secure and autonomic system for pervasive healthcare. 2009 3rd International Conference on Pervasive Computing Technologies for Healthcare - Pervasive Health 2009, PCTHealth 2009, 2009.

[8] InProceedings, Schaeffer-Filho, A.; Lupu, E. & Sloman, M. Realising management and composition of Self-Managed Cells in pervasive healthcare. 2009 3rd International Conference on Pervasive Computing Technologies for Healthcare, 2009, 1-8.

[9] J. Kephart, D. Chess. The Vision of Autonomic Computing. IEEE Computer 2003;36:41-50.

[10] E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, S. L. Keoh, A. Schaeffer-Filho, and K. Twidle, "AMUSE: Autonomic Management ofUbiquitous e-Health Systems," Concurrency and Computation: Practice and Experience, 2007.

[12] E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, K. Twidle, S.-L. Keoh, and A. Schaeffer-Filho, "AMUSE: autonomic management of ubiquitous systems for e-health," J. Concurrency and Computation: Practice and Experience, John Wiley, May 2007.

[13] B.A. Kitchenham, Procedures for Undertaking Systematic Reviews, Joint Technical Report, Computer Science Department, Keele University (TR/SE-0401) and National ICT Australia Ltd. (0400011T.1), 2004.

[14] InProceedings Petersen, K.; Feldt, R.; Mujtaba, S. & Mattsson, M.Systematic Mapping Studies in Software Engineering Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, BCS Learning & Development Ltd., 2008, 68-77.

[15] Article Abeywickrama, D. B. & Ovaska, E. A survey of autonomic computing methods in digital service ecosystems. Service Oriented Computing and Applications, 2017, 11, 1-31.

[16] Article. Ramirez, A. J.; Knoester, D. B.; Cheng, B. H. C. & McKinley, P. K. Plato: a genetic algorithm approach to run-time reconfiguration in autonomic computing systems. Cluster Computing, 2011, 14, 229-244.

[17] Article. Sterritt, R. Autonomic computing. Innovations in Systems and Software Engineering, 2005, 1, 79-88.

[18] Article Urovi, V.; Olivieri, A. C.; Bromuri, S.; Fornara, N. & Schumacher, M. I. A Semantic Publish-subscribe Coordination Framework for IHE Based Cross-community Health Record Exchange. SIGAPP Appl. Comput. Rev., ACM, 2013, 13, 38-49.

[19] InProceedings. Taweel, A.; Speedie, S.; Tyson, G.; Tawil, A. R.; Peterson, K. & Delaney, B.Service and Model-driven Dynamic Integration of Health Data Proceedings of the First International Workshop on Managing Interoperability and Complexity in Health Systems, ACM, 2011, 11-17.

[20] InCollection Rutten, E.; Marchand, N. & Simon, D. de Lemos, R.; Garlan, D.; Ghezzi, C. & Giese, H. (Eds.) Feedback Control as MAPE-K Loop in Autonomic Computing Software Engineering for Self-Adaptive Systems III. Assurances, Springer International Publishing, 2017, 9640, 349-373.

[21] InCollection Quadri, M. A. R.; Mannava, V. & Ramesh, T. Wyld, D. C.; Zizka, J. & Nagamalai, D. (Eds.) Service Composition Design Pattern for Autonomic Computing Systems Using Association Rule Based Learning Advances in Computer Science, Engineering & Applications, Springer Berlin Heidelberg, 2012, 167, 1017-1025.

[22] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed., ser. SEI Series in Software Engineering. Addison Wesley, 2013.

[23] Rumbaugh, J., Jacobson, I., & Booch, G. (2000). El Lenguaje Unificado de Modelado Manual de Referencia. Mexico: Addison Wesly.

[24] Software engineering-Software product Quality, Requirements and Evaluation (SQuaRE) Quality model, ISO/IEC JTC1/SC7 N4386, 2009-07-16.

[25] Requerimientos, A. y. (s.f.). *Software Guru.* Recuperado el 25 de Junio de 2020, de sg.com.mx: https://sg.com.mx/revista/28/requerimientos-y-arquitectura

[26] Ralph, P. & Wand, Y. (2009). A Proposal for a Formal Definition of the Design Concept, Lecture Notes in Business Information Processing , 14, pp. 103-136.

[27] Cervantes Maceda H, (2016), Arquitectura de Software: Conceptos y Ciclo de Desarrollo, Mexico DF, Cengage Learning.

[28] Vision, D. d. (s.f.). *IBM® IBM Knowledge Center.* Recuperado el 25 de Junio de 2020, de ibm.com: https://www.ibm.com/support/knowledgecenter/es/SSYMRC_6.0.2/com.ibm.rational.rrm.help.doc/topics/r_vision_doc.html

[29] Nour Ali.,Alfonso Martnez.,Lorena Ayuso.,Angelina Espinoza.(2006). Self-Adaptive Quality Requirement Elicitation Process for Legacy Systems: A Case Study in HealthCare.

[30] Inc. Eclipse Foundation. (s.f.). *OpenUP*. Recuperado el 25 de Junio de 2020, de utm.mx: http://www.utm.mx/~caff/doc/OpenUPWeb/

[31] Inc. Eclipse Foundation. (s.f.) The Platform for Open Innovation and Collaboration. Recuperado el 25 de Junio de 2020. de The Eclipse Foundation. eclipse.org: https://www.eclipse.org.

[32] Kruchten, P. (Noviembre 1995). Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura del Software. *IEEE Software*.

[33] Object Management Group. (s.f.). *Meta-Object Facility*. Recuperado el 25 de Junio de 2020, de omg.org: https://www.omg.org/mof/

[34] Open Galen. (s.f.). *OpenGalen*. Recuperado el 25 de Junio de 2020, de opengalen.org: http://www.opengalen.org

[35] Carnegie Mellon Unirvesity. (n.d.). *Software Engineering Institute*. Recuperado 25 de Junio 2020, de sei.cmu.edu: https://sei.cmu.edu

[36]. Ballot, H. D. (Enero de 2015). *Health Level Seven International.* Recuperado el 25 de Junio de 2020, de hl7.org: http://www.hl7.org/documentcenter/public/standards/mu_draft/CDAR2_IG_CDP_R1_D2_2015JANdraft.pdf

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00087
Matrícula: 2173802305

Arquitectura autonómica para sistemas de software de apoyo al diagnóstico médico.

Con base en la Legislación de la Universidad Autónoma Metropolitana, en la Ciudad de México se presentaron a las 10:00 horas del dia 13 del mes de julio del año 2020 POR VÍA REMOTA ELECTRÓNICA, los suscritos miembros del jurado designado por la Comisión del Posgrado:

DRA. REYNA CAROLINA MEDINA RAMIREZ
M. EN C. MARCO ANTONIO NUÑEZ GAONA
LIC: LUIS FERNANDO CASTRO CAREAGA

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo
DIRECCION DE SISTEMAS ESCOLARES

Bajo la Presidencia de la primera y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS (CIENCIAS Y TECNOLOGIAS DE LA INFORMACION)

DE: NARMO DIONEY REYES CARDOSO

NARMO DIONEY REYES CARDOSO
ALUMNO

y de acuerdo con el articulo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

APROBAR

Acto continuo, la presidenta del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

REVISÓ

MTRA. ROSALIA SERRANO DE LA PAZ
DIRECTORA DE SISTEMAS ESCOLARES

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JESUS ALBERTO OCHOA TAPIA

PRESIDENTA

DRA. REYNA CAROLINA MEDINA RAMIREZ

VOCAL

M. EN C. MARCO ANTONIO NUÑEZ GAONA

SECRETARIO

LIC. LUIS FERNANDO CASTRO CAREAGA

El presente documento cuenta con la firma –autógrafa, escaneada o digital, según corresponda- del funcionario universitario competente, que certifica que las firmas que aparecen en esta acta – Temporal, digital o dictamen- son auténticas y las mismas que usan los c.c. profesores mencionados en ella