UNIVERSIDAD GASTÓN DACHARY
DEPARTAMENTO DE INGENIERÍA Y CIENCIAS DE LA PRODUCCIÓN
INGENIERÍA EN INFORMÁTICA

INGENIERÍA DEL SOFTWARE

TRABAJO PRÁCTICO INTEGRADO

DOCENTES:
Amatte, Fernando Javier
Belloni, Edgardo

INTEGRANTES - GRUPO 1:
Arroyos, Agustín
Echeverría, Celeste Oriana
Kaechele, Axel
Terleski, Lorenzo Isaac
Velazquez, Daniel Mauricio
Zschach, Alan Erick

2023

Índice

Índice de Figuras

Índice de Tablas

Introducción

El objetivo de este trabajo es revisar y perfeccionar el estudio de nuestro caso seleccionado, así como identificar cómo los conceptos presentados en los capítulos sobre sistemas confiables, ingeniería para la fiabilidad, seguridad, protección y resiliencia pueden enriquecer nuestra comprensión y enfoque en la ingeniería de software.

En las secciones subsiguientes, analizaremos cada uno de estos conceptos y procesos para aplicarlos de manera efectiva a nuestro proyecto.

Este trabajo no sólo busca enriquecer nuestro enfoque en el desarrollo de software, sino también promover la comprensión de los desafíos y enfoques necesarios para crear sistemas confiables y seguros en un mundo cada vez más dependiente de la tecnología.

Consignas

Parte I

1. **SINTETICEN PUNTOS CLAVE de (modelos de) procesos de desarrollo de software cubiertos u objeto de revisión autónoma en esta asignatura.** Por ej. incluyan: Modelos Dirigidos por un Plan; Modelos Genéricos de Procesos de Desarrollo Componibles e Híbridos (Unified Process o variantes del mismo, por ej. Open UP); Métodos Ágiles; y, V-Model; entre otros.
2. **ELABOREN UN RESUMEN SINTÉTICO, y a modo de CV ACTUALIZADO a noviembre de 2023, de las experiencias y conocimientos de los integrantes del Grupo de TPI aplicando el/los proceso/s revisados en el punto 1.**
3. **SELECCIONEN, junto con su Grupo de TPI, UN (1) CASO DE ESTUDIO (Sistema Socio-Técnico Crítico de Riesgo para la salud, vida, economía y/o educación de personas).** Indiquen motivaciones para la selección.
4. **PARA EL CASO DE ESTUDIO:**
   - 4.1. DESCRIBAN DE MANERA SINTÉTICA, pero comprensible, EL CASO DE ESTUDIO Y SUS ANTECEDENTES, incluyendo además de su contexto, stakeholders, requerimientos (Funcionales y No Funcionales) que consideran esenciales, etc., **la caracterización del caso en cuanto**:
     - Su definición y clasificación como un Sistema Socio-Técnico Crítico.
     - Requisitos (Atributos de Calidad) NO Funcionales principales que remiten directamente a Propiedades de Confiabilidad de Productos Software, cubiertas en la asignatura.
     - Al/los Tipos de producto y de Tipo de Sistemas o Aplicaciones de Software involucrados (genérico o hecho a medida; sistema de sistemas, sistema embebido, etc.) cubiertos en la asignatura.
   - 4.2. **ELABOREN UNA DESCRIPCIÓN ACERCA DE CÓMO ADAPTAR EL CASO A UNA REALIDAD LOCAL, REGIONAL O NACIONAL**. Resaltar cuáles aspectos ya han profundizado y cuáles aspectos debieran profundizar más para poder hacer esto que describen. Por ej.: Incluir legislación/es revisadas y a revisar; organizaciones y stakeholders contactados y a contactar y entrevistar, indagar, etc.
   - 4.3. **Considerando criterios generales y particularmente atributos asociados a la confiabilidad de procesos de desarrollo; ANALICEN VENTAJAS Y DESVENTAJAS DE APLICAR -CADA UNO DE LOS PROCESOS DE DESARROLLO DE SOFTWARE REVISADOS EN EL PUNTO 1- AL CASO DE ESTUDIO ADAPTADO**. Justificar cada análisis realizado. Sinteticen en un cuadro comparativo el análisis de ventajas y desventajas realizado.
   - 4.4. **DETERMINEN EL/LOS PROCESO/S MÁS CONVENIENTE/S DE APLICAR AL CASO DE ESTUDIO ADAPTADO**. Justificar de forma sintética la respuesta. Aporte en un único cuadro resumen el/los proceso/s seleccionado/s como más conveniente/s de aplicar al caso de estudio.
5. **ANALICEN CÓMO DEBIERA SIMPLIFICARSE Y/O ADAPTARSE EL PROCESO DE DESARROLLO SELECCIONADO EN EL PUNTO 4 (d)**; considerando, particularmente, conceptos inherentes a la Ingeniería para la Confiabilidad y

Seguridad de Sistemas cubiertos en la asignatura, y de acuerdo a cuál/es Propiedades de Confiabilidad (y, sub propiedades directamente asociadas) deberá soportar el producto SW a desarrollar. Indicar en cuáles disciplinas y actividades fundamentales qué (métodos, técnicas, heurísticas y/o herramientas, etc.) descartarán o incorporarán justificadamente.

6. **ELABOREN UN META-MODELO DEL PROCESO DE DESARROLLO GLOBAL FINALMENTE SIMPLIFICADO Y/O ADAPTADO**. Es decir, construir un Modelo del Modelo de Proceso (o combinación de Procesos) seleccionado(s) como conveniente(s) de aplicar al producto SW, y que fuera finalmente simplificado y/o adaptado en punto 5.
    - Identificar y utilizar los Tipos de Diagramas§ de UML que considere convenientes para cada perspectiva relevante.
    - Aportar breve descripción de elementos (al menos, actividades fundamentales, pero también podrían incluirse roles, artefactos, etc.).
7. **ELABOREN UN META-MODELO DEL PROCESO DE INGENIERÍA O DISCIPLINA DE REQUERIMIENTOS INCLUIDA EN EL PROCESO DE DESARROLLO GLOBAL FINALMENTE SIMPLIFICADO Y/O ADAPTADO** (el del punto 6, ut supra).
    - Identificar y utilizar los tipos de diagramas de UML que sean convenientes para incluso modelar principales flujos de trabajo en la disciplina.
    - Aportar breve descripción de elementos (al menos, actividades o tareas principales, pero también podrían incluirse roles, artefactos, etc.).
8. **ELABOREN UN META-MODELO DEL PROCESO O DISCIPLINA DE DISEÑO INCLUIDA EN EL PROCESO DE DESARROLLO GLOBAL FINALMENTE SIMPLIFICADO Y/O ADAPTADO** (el del punto 6, ut supra).
    - Identificar y utilizar los tipos de diagramas de UML que sean convenientes para incluso modelar principales flujos de trabajo en la (sub)disciplina.
    - Aportar breve descripción de elementos (al menos, actividades o tareas principales, pero también podrían incluirse roles, artefactos, etc.). Garantizar que se incluye algún método de evaluación de arquitecturas alternativas.
9. **ELABOREN UN META-MODELO DEL PROCESO O DISCIPLINA DE VERIFICACIÓN (PRUEBA DEL SOFTWARE) INCLUIDA EN EL PROCESO DE DESARROLLO GLOBAL FINALMENTE SIMPLIFICADO Y/O ADAPTADO** (el del punto 6, ut supra).
    - Identificar y utilizar los tipos de diagramas de UML que sean convenientes para incluso modelar principales flujos de trabajo en la (sub)disciplina.
    - Aportar breve descripción de elementos (al menos, actividades o tareas principales, pero también podrían incluirse roles, artefactos, etc.).
10. **ANALICEN BREVEMENTE CONSIDERACIONES PRELIMINARES ACERCA DE ESTÁNDARES A TENER EN CUENTA, ENTORNOS Y TECNOLOGÍAS A UTILIZAR Y/O INTEGRAR PARA LA IMPLEMENTACIÓN DEL PRODUCTO SOFTWARE A DESARROLLAR.**

Parte II

11. **EJECUTEN EL MODELO DE PROCESO DE DESARROLLO FINALMENTE SIMPLIFICADO Y/O ADAPTADO, de manera que de forma general se cubra (al menos parcialmente) lo siguiente:**
    - INGENIERÍA DE REQUERIMIENTOS incluyendo:
        - Una versión del documento Visión del Proyecto. El cual podrá incluir Síntesis de Análisis que hayan realizado respecto de: Estudio de Mercado o Negocio, Dominio del Problema, Riesgos, etc.
        - Identificación y definición de (los principales) Requerimientos de Usuario.
        - Identificación y especificación de (los principales) Requerimientos del Sistema correspondientes al punto anterior. Si lo consideraron pueden utilizar especificaciones formales.
        - Requerimientos Funcionales, y No Funcionales indicando subclasificaciones y métricas correspondientes.
        - Un Diagrama de Casos de Uso correspondiente a puntos anteriores.
    - DISEÑO ARQUITECTÓNICO incluyendo:
        - Al proponer y seleccionar Arquitecturas de Sistemas alternativas, considerar más allá de los requisitos no funcionales principales -si correspondiera al caso de estudio- aspectos de Ingeniería de Software Distribuido y/o de Tiempo Real.
        - Selección y evaluación (al menos inicial, con algún/os escenarios importantes para el caso de estudio) de Arquitecturas de Sistemas alternativas con método ATAM del SEI en CMU o método alternativo.
    - DISEÑO DETALLADO incluyendo:
        - Consideraciones de utilización de Patrones de Diseño de Gamma et al.
    - PROYECCIÓN DEL DESARROLLO (en ciclos de desarrollo iterativo e incremental, si correspondiera) incluyendo: Consideraciones concretas de Implementación (Entorno, Tecnologías a utilizar y/o integrar, etc.) y Estrategias de Prueba del Software a utilizar.
12. **DISCUTAN acerca de si se han cumplido (y en qué grado) los objetivos del trabajo.**
13. **DISCUTAN limitaciones generales y particulares (de diseño, técnicas u operativas y de herramientas o tecnologías que proponen) para construir el producto software a desarrollar.**
14. **DISCUTAN la evolución que consideran podría tener el producto software a desarrollar y modelo de negocio previsible. Por ejemplo, si consideran e imaginan que puedan corresponder, consideren temáticas cubiertas en Ingeniería de Software Avanzada -Reutilización de Software (Framework, Línea de Producto, etc.), Ingeniería Basada en Componentes, Software como Servicio, Ingeniería Basada en Servicios.**
15. **INCLUYAN UN ANEXO, donde co evalúen su producción respecto de los criterios consignados para el trabajo.**

Desarrollo

Parte I

1. Modelos de Proceso de Desarrollo de Software

A continuación se sintetizan puntos clave de procesos de desarrollo de software a revisar en la asignatura. Se incluirán únicamente aquellos procesos dirigidos por un plan *(Plan-driven processes)*, debido a que son más adecuadas para el desarrollo de sistemas socio-críticos, gracias a su definición y planificación exhaustiva.

1.1. Modelos Genéricos

Los Modelos Genéricos de Desarrollo de Software son enfoques conceptuales que describen las etapas y secuencias de actividades involucradas en el desarrollo de software. Estos modelos genéricos son paradigmas abstractos, de alto nivel, que pueden ser extendidos y adaptados para crear procesos más específicos según las necesidades de un proyecto o una organización. (Sommerville, 2015, p. 45)

1.1.1. Modelo en Cascada

El Modelo en Cascada toma las actividades básicas del proceso de especificación, desarrollo, validación y evolución, y las representa como etapas independientes del proceso, como la especificación de requerimientos, el diseño de software, la implementación, las pruebas, etc.

Es un enfoque secuencial y lineal para el desarrollo de software. En este modelo, todo el proceso es planificado y organizado antes de iniciar el proceso de desarrollo. El desarrollo de software sigue una secuencia fija de etapas, donde cada etapa se completa antes de pasar a la siguiente. (Sommerville, 2015, p. 45)

Además. cada etapa debe ser documentada antes de avanzar a la siguiente, y la documentación debe ser actualizada para reflejar los cambios en el sistema. Las etapas presentadas por Sommerville (Sommerville, 2015, p. 48) son:

1. Análisis y definición de requerimientos.
2. Diseño del sistema.
3. Implementación y pruebas de unidad.
4. Integración y pruebas del sistema.
5. Operación y mantenimiento.

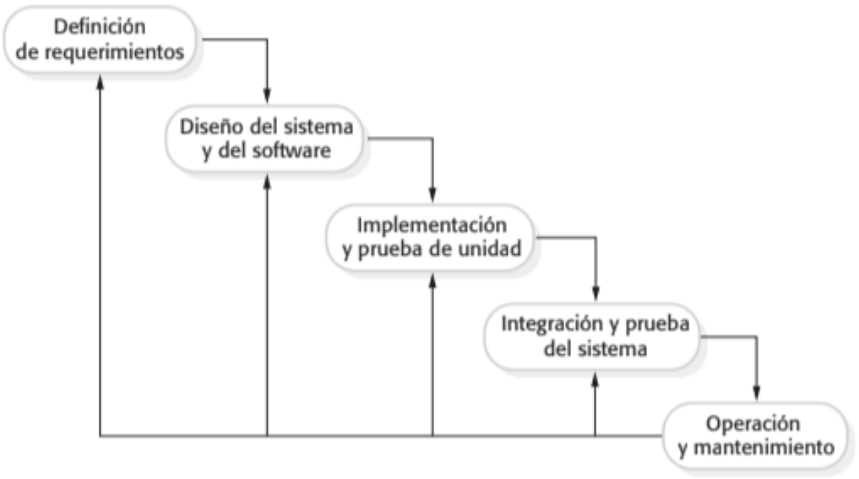


Figura 1.1: El modelo en cascada (Sommerville, 2016, p. 47, Figure 2.1: The waterfall model)

1.1.2. Modelo V

El Modelo V es un enfoque abstracto, aplicable tanto a software como a hardware, que puede ser fácilmente adaptado a una gran diversidad de proyectos y organizaciones según sus necesidades particulares. Se basa en cuatro submodelos o áreas de funcionalidad, conteniendo actividades (y subactividades) que transforman los productos hasta la versión final:

1. Gestión de Proyecto (PM - Project Management): planifica, monitorea, controla el proyecto y transmite información a otros submodelos.
2. Desarrollo del Sistema (SD - System Development): desarrolla el sistema.
3. Control de Calidad (QA - Quality Assurance): especifica los requisitos y criterios de calidad, realiza casos de prueba para los productos.
4. Gestión de Configuración (CM - Configuration Management): administra los productos generados.

Los productos transitan diferentes estados durante su generación y procesamiento, cuyo cambio sólo puede ser producido por una actividad. Cada producto pasa por cuatro estados: Planificado, En proceso, Enviado y Aceptado.

Además, todas las actividades se interrelacionan con tres niveles:

1. Procedimientos: es el modelo de procesos del ciclo de vida, especifica actividades, sus entradas y salidas responde a la pregunta “¿Qué hay que hacer?”.
2. Métodos: para cada actividad se selecciona un método, responde a la pregunta “¿Cómo se hace?”.
3. Herramientas requeridas: establece qué características funcionales deben tener las herramientas que se utilicen para realizar las actividades, responde a la pregunta “¿Qué se usa para realizarlo?”.

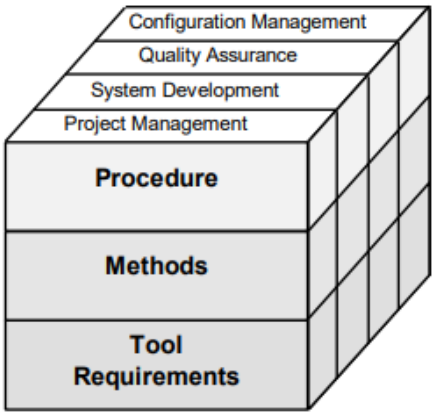


Figura 1.2: La arquitectura del Modelo V. (Bucanac, 1999, p. 3)

1.1.3. Modelo Incremental

Este enfoque vincula las actividades de especificación, desarrollo y validación. El sistema se desarrolla como una serie de versiones (incrementos), y cada versión añade funcionalidad a la versión anterior.

El desarrollo incremental se basa en la idea de diseñar una implementación inicial, exponer ésta al comentario del usuario, y luego desarrollarla en sus diversas versiones hasta producir un sistema adecuado. Las actividades de especificación, desarrollo y validación están entrelazadas en vez de separadas, con rápida retroalimentación a través de las actividades. (Sommerville, 2015, p. 49)

Por lo general, los primeros incrementos del sistema incluyen la función más importante o la más urgente. Esto significa que el cliente puede evaluar el desarrollo del sistema en una etapa relativamente temprana, para constatar si se entrega lo que se requiere. En caso contrario, sólo el incremento actual debe cambiarse y, posiblemente, definir una nueva función para incrementos posteriores. (Sommerville, 2015, p. 50)

La incorporación de más cambios de software se vuelve cada vez más difícil y costosa. Los problemas del desarrollo incremental se tornan particularmente agudos para sistemas grandes, complejos y de larga duración, donde diversos equipos desarrollan diferentes partes del sistema. Esto debe planearse por adelantado en vez de desarrollarse de manera incremental. Se puede desarrollar un sistema incremental y exponerlo a los clientes para su comentario, sin realmente entregarlo e implementarlo en el entorno del cliente. La entrega y las implementaciones incrementales significan que el software se usa en procesos operacionales reales. (Sommerville, 2015, p. 51)

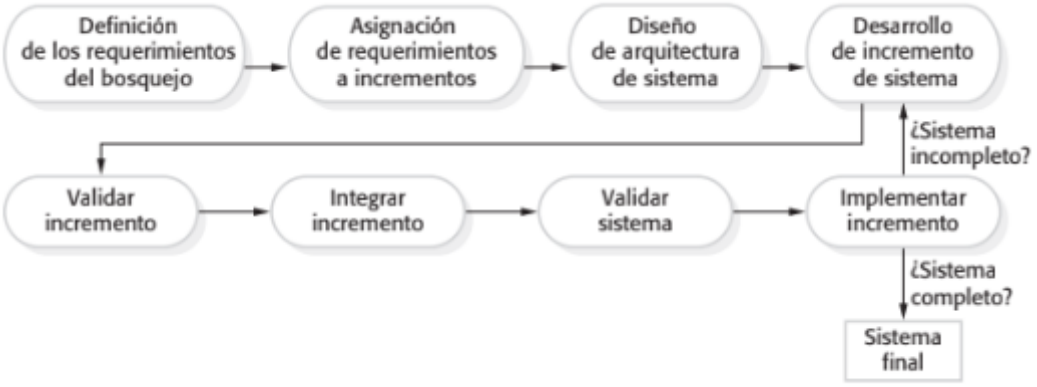


Figura 1.3: Entrega incremental (Sommerville, 2016, p. 64 Figure 2.10: Incremental delivery)

1.1.4. Modelo en Espiral

El modelo en espiral sigue un enfoque incremental, pero se enfatiza más en el análisis de riesgos. El modelo en espiral tiene cuatro fases:

1. Planificación
2. Análisis de Riesgos
3. Ingeniería
4. Evaluación

Un proyecto de software pasa repetidamente por estas fases en iteraciones (llamadas Espirales en este modelo). La Espiral base, comenzando en la fase de planificación, se reúnen los requisitos y se evalúa el riesgo. Cada espiral posterior se basa en la espiral base. Se recopilan requisitos durante la fase de planificación. En la fase de análisis de riesgos, se lleva a cabo un proceso para identificar riesgos y soluciones alternativas. Al final de la fase de análisis de riesgos se produce un prototipo. El software se produce en la fase de ingeniería, junto con pruebas al final de la fase. La fase de evaluación permite al cliente evaluar la salida del proyecto hasta la fecha antes de que el proyecto continúe a la siguiente espiral. (Tumbas & Matkovic, 2010, p.167)

1.1.5. Modelo Orientado a la Reutilización

Desde el 2000, los procesos de desarrollo de software que se enfocan en la reutilización de software existente se han vuelto ampliamente utilizados. En la mayoría de proyectos se reutiliza software, modificándolo según las necesidades e integrándose con el código desarrollado. Este enfoque se basa en componentes de software reutilizables y un framework de integración para la composición de los mismos.

Hay tres tipos de componentes de software que se reutilizan con frecuencia: sistemas de aplicación independientes, colecciones de objetos y servicios web. En este proceso basado en la reutilización, las etapas son:

1. Especificación de requerimientos
2. Descubrimiento y evaluación de Software

3. Refinamiento de requerimientos
4. Configuración de aplicaciones del sistema
5. Adaptación e integración de componentes.

La reutilización de software reduce los costos y los riesgos y también acelera la entrega del software. Sin embargo, también hay desventajas a considerar, como compromisos en los requisitos y la pérdida de control sobre la evolución del sistema. (Sommerville, 2015, p. 52)

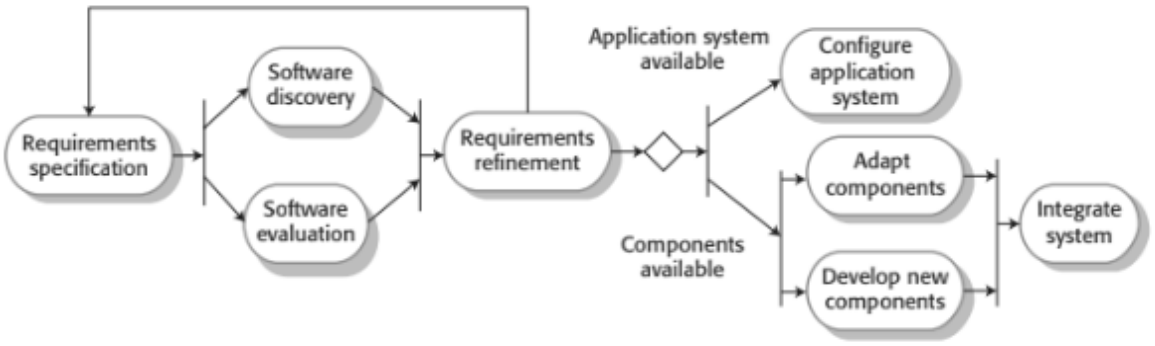


Figura 1.4: Ingeniería del Software basada en la reutilización. (Sommerville, 2016, p.52, Figure 2.3: Reuse-oriented software engineering)

1.1.6. Modelo Basado en Prototipos

Un prototipo es una versión temprana de un sistema de software que es utilizado para demostrar conceptos, probar opciones de diseño y analizar más a fondo los problemas y posibles soluciones del sistema. Un prototipo de software puede ser utilizado en el desarrollo de software para anticipar posibles cambios requeridos:

1. En el proceso de ingeniería de requisitos, un prototipo puede ayudar a la elicitación y validación de los requisitos del sistema.
2. En el proceso de diseño, un prototipo puede ser utilizado para explorar soluciones y desarrollar interfaces de usuario.

Los objetivos del desarrollo basado en prototipos deben ser especificados desde el comienzo del proceso. Luego, se debe decidir qué se incluye y qué se deja fuera del prototipo. La etapa final consiste en evaluar el prototipo, con la colaboración de los usuarios. (Sommerville, 2015, p.63)

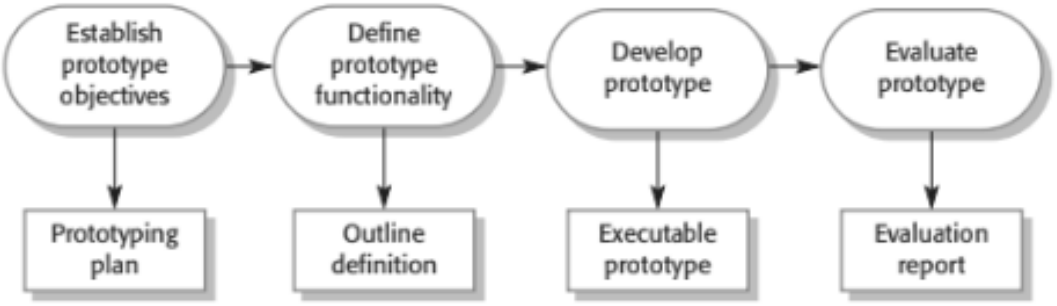


Figura 1.5: Desarrollo de Prototipos. (Sommerville, 2016, p. 63, Figure 2.9: Prototype development)

1.2. Modelos Híbridos

Los modelos de proceso híbridos son enfoques de desarrollo de software que combinan elementos y prácticas de diferentes modelos de proceso tradicionales o ágiles. Estos modelos se crean con el objetivo de aprovechar las fortalezas de múltiples enfoques y adaptarse a las necesidades específicas de un proyecto o una organización.

1.2.1. Modelo de Proceso Unificado

El Proceso Unificado de Desarrollo de Software es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado es el Proceso Unificado Racional (RUP) que es un modelo de proceso adaptable que se ha derivado del trabajo en UML. (IBM, 2021) Sus características son:

1. Dirigido por Casos de Uso: Un caso de uso es un fragmento de funcionalidad del sistema que proporciona un resultado de valor a un usuario. Los casos de uso modelan los requerimientos funcionales del sistema.
2. Centrado en la arquitectura. El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La arquitectura se representa mediante vistas del modelo.
3. Iterativo e incremental. Es iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases está a su vez dividida en una serie de iteraciones. (IBM, 2021)

1.2.2. Modelo Open UP

El modelo OpenUP es una extensión del Proceso Unificado que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado. Algunos de los principios básicos son: equilibrar las prioridades contrapuestas para maximizar el valor de las partes interesadas, evolucionar para obtener retroalimentación y mejorar continuamente, y centrarse en la arquitectura desde el principio para minimizar los riesgos y organizar el desarrollo.

Este modelo divide el proyecto en iteraciones: intervalos planificados en un tiempo determinado, enfocadas en el desarrollo de entregas incrementales. El ciclo de vida del proyecto se estructura en cuatro fases: Inicio, Elaboración, Construcción y Transición. Además, le otorga visibilidad a los interesados y a los miembros del equipo, lo que ayuda a la toma de decisiones.

2. Experiencias del Grupo (CV)

2.1. Formación Académica

- **Educación Secundaria Completa** (Todos los integrantes)
- **Nivel Universitario en curso:** Ingeniería en Informática (Todos los integrantes)

2.1.1. Asignaturas relacionadas

En Ingeniería en Informática:
- Algoritmos y Estructuras I - II
- Análisis de Sistemas
- Diseño de Sistemas
- Metodologías Avanzadas
- Programación Estructurada
- Programación Avanzada I - II
- Taller de Aplicaciones Web

2.1.2. Proyectos realizados

- **Programación Estructurada y Algoritmos y Estructuras II:** Se realizó la implementación de un sistema de Estacionamiento Medido (SEM) y de un sistema de Conteo de votos en Elección de Diputados y Senadores.
  Ambos fueron implementados en lenguaje de programación C, en equipos de 2 o 3 personas, incluyendo acceso a archivos y estructuras de datos dinámicas.

- **Análisis y Diseño Estructurado de Sistemas:** Se realizó el análisis y diseño de sistemas de diversos escenarios aplicando el Modelo en Cascada.
  Entre estos casos de estudio podemos mencionar: tienda de juegos de ocio “Arcadia”, Dietética “Consciencia”, una Librería, Venta de domos, Veterinaria “Patán” y cabañas "Domos misiones".

- **Programación Avanzada:** Se desarrolló de forma incremental un sistema de gestión de turnos bancario, un sistema de gestión de campos de cultivo, un sistema de gestión de obras en construcción, donde se mostraban avances parciales del trabajo por semana a los docentes de la cátedra hasta llegar a una entrega final.

- **Taller de Aplicaciones Web:** Se desarrolló de forma incremental un sistema de consultas y administración de películas en una biblioteca personal, integrando: encriptación de activos críticos (password), conexión y consulta a servicios de terceros (API trakt), autenticación de usuario, validación de email y otros datos de entrada, y arquitectura REST.

2.2. Experiencia Laboral

Los integrantes del equipo no contamos con experiencia laboral.

2.3. Información Adicional

- **Idiomas:** Todos los integrantes del equipo poseemos conocimientos sólidos del Inglés, y somos capaces de interpretar y comunicarnos efectivamente en el idioma.

- **Proyectos independientes:** Mauricio Velázquez trabajó de manera independiente en un proyecto de desarrollo de un videojuego para móviles, utilizando Solar 2D y el lenguaje de programación Lua.

3. Selección del Caso de Estudio

Innopump es una bomba de insulina personal.

El problema del tratamiento convencional de la diabetes es que implica medir los niveles de azúcar del paciente y calcular la dosis de insulina correspondiente para inyectarla. Sin embargo, la cantidad de insulina que necesita el paciente no depende sólo del nivel de glucosa en sangre, sino también de otros factores, como el tiempo de la última inyección de insulina, la velocidad del sistema digestivo del paciente, entre otros.

Con el objetivo de reemplazar el sistema convencional de tratamiento, el sistema de la bomba de insulina realiza varias mediciones, evalúa la tasa de cambio del azúcar en sangre, y, basándose en el nivel actual y en la tasa y dirección del cambio, calcula y administra la cantidad incremental de insulina que debe ser inyectada.

El software que lo controla es un sistema embebido, que recopila información de un sensor o conjunto de sensores y controla una bomba que suministra una dosis controlada de insulina cuando es necesaria.

Además de la seguridad o protección contra daños físicos, este sistema debe mantener altos niveles de confiabilidad: principalmente disponibilidad y confiabilidad, incluyendo seguridad contra daños físicos y fiabilidad.

Estudiar este caso nos brinda una rica matriz de aprendizaje donde convergen la ingeniería de software, el diseño de hardware, la interacción humano-computadora, y las políticas de salud pública.

Su análisis ofrece una oportunidad única para comprender la complejidad y la importancia de los sistemas socio-técnicos críticos como también los desafíos inherentes al diseño de un sistema que debe operar con una precisión y fiabilidad casi perfectas, dada su importancia en la preservación de la vida humana. Un error de cálculo o una falla mecánica no son meras inconveniencias, sino que tienen el potencial de traducirse en consecuencias fatales.

4. Acerca del Caso de Estudio

4.1. Descripción

Una bomba de insulina es un sistema **socio-técnico crítico médico** que simula el funcionamiento del páncreas, un órgano interno. Este dispositivo lleva un control preciso de la glucosa (azúcar) en sangre, el cual es esencial para personas que padecen diabetes.

Es considerado un sistema **crítico** debido a que una falla en su funcionamiento puede provocar lesiones a las personas. En este caso, un funcionamiento incorrecto o impreciso del sistema puede dañar la salud de los pacientes de manera **directa**. (Sommerville, 2015, p. 287). Por lo tanto, el diseño del sistema debe enfocarse en mantener la seguridad o protección contra daños físicos *(safety)*.

Es considerado también un sistema **sociotécnico**, debido a que incluye tanto elementos técnicos (software, procesadores, sensores) como no técnicos (personas, regulaciones, organizaciones): involucra a los pacientes de diabetes, profesionales de la salud quienes monitorean periódicamente, configuran y asesoran la dosificación; las regulaciones y los estándares que controlan el lanzamiento de estos productos. (Sommerville, 2015, p. 292)

Software como producto

- **Software genérico:** el sistema será desarrollado para el mercado en general, promocionado y comercializado a cualquier cliente que lo quiera comprar.
  La especificación de software y su desarrollo serán controlados por el equipo de desarrollo.

Tipo de aplicación

- **Sistema (de control) embebido:** El software de la bomba de insulina es un sistema que opera y controla dispositivos de hardware.

4.1.1. Requisitos funcionales iniciales

Los requerimientos funcionales esenciales de la bomba de insulina:

1. El controlador debe ejecutar un algoritmo de auto testeo cada 1 minuto.
2. El sistema debe medir el nivel de azúcar en sangre y suministrar insulina cada 10 minutos, si fuese necesario.
3. La dosis de insulina a ser suministrada debe ser calculada mediante la medición del nivel actual de azúcar en sangre, comparando éste a un nivel de medición previo.
4. La cantidad de insulina a ser suministrada debe ser calculada según la lectura actual de azúcar, según lo mida el sensor:
   4.1. Si la lectura está por debajo del mínimo, no se debe suministrar insulina.
   4.2. Si la lectura está entre los parámetros adecuados, la insulina sólo será suministrada si el nivel de azúcar está creciendo y la tasa de incremento del azúcar está aumentando.

4.3. Si la lectura está por sobre el nivel recomendado, la insulina se suministra a menos que el nivel de azúcar en sangre esté decreciendo y la tasa de decremento del mismo esté incrementando.
4.4. La cantidad de insulina realmente suministrada puede ser diferente a la dosis calculada debido a las distintas restricciones de seguridad (safety) que se incluyen en el sistema. Hay un límite de la dosis máxima a ser suministrada en cada inyección, y un límite acumulativo por día.

5. El usuario debe poder reemplazar el reservorio de insulina por uno nuevo en cualquier momento
6. El sistema debe admitir un reservorio de 315 ml de insulina.
7. Las condiciones de error que el sistema debe detectar e indicar son:
   7.1. Batería baja: el voltaje de la batería está por debajo de 0.5 v.
   7.2. Falla de sensor: el auto testeo del sensor de azúcar ha resultado erróneo.
   7.3. Falla de la bomba: el auto testeo de la bomba ha resultado erróneo.
   7.4. Falla del suministro: la aguja podría estar bloqueada o insertada incorrectamente.
   7.5. Aguja removida: el usuario ha removido la aguja.
   7.6. Reservorio de insulina removido: el usuario ha removido el reservorio de insulina.
   7.7. Bajo nivel de insulina: el nivel de insulina está bajo (indicando que el reservorio debe cambiarse).

4.1.2. Requisitos no funcionales:

- **Confiabilidad** *(dependability)*
  - **Fiabilidad** *(reliability)*
  - **Precisión**
  - **Corrección**
- **Disponibilidad** *(availability)*
- **Seguridad o protección contra daños físicos** *(safety)*
- Seguridad o protección contra accesos no autorizados *(security)*

4.2. Adaptación a una Realidad Local, Regional o Nacional

4.2.1. Regulaciones y leyes

- **Administración Nacional de Medicamentos, Alimentos y Tecnología Médica (ANMAT):** es el ente regulador encargado del control y la fiscalización de la calidad y sanidad de los productos, substancias, elementos y materiales que se consumen o utilizan en la medicina, alimentación y cosmética humanas, y del control de las actividades, procesos y tecnologías que mediaren o estuvieren comprendidos en dichas materias. (Portal Oficial del Estado Argentino, n.d.)
  Algunas normativas específicas a tener en cuenta son:

  - **Disposición ANMAT 9281/2022:** Es la disposición que crea el Programa de Asistencia Regulatoria a la Investigación y Desarrollo de Productos para la Salud (AR I+D), que tiene como objetivo brindar asesoramiento técnico y científico a los investigadores y desarrolladores de productos médicos innovadores. (Portal Oficial del Estado Argentino, 2022)
  - **Disposición ANMAT 8504/2021:** Es la disposición que establece los requisitos técnicos y administrativos para la inscripción de los productos médicos basados en software, incluyendo los software como producto médico (SaMD) y los software de apoyo a la decisión clínica (SDCD). (Portal oficial del Estado argentino, 2022)

- **Gobierno de la República Argentina:**
  - **Decreto 1490/92:** Es el decreto que establece el régimen de autorización, registro, fabricación, importación, exportación, comercialización y vigilancia de los productos médicos en Argentina. (Ministerio de Justicia y Derechos Humanos, 1992)
  - **Ley de Protección de Datos Personales (Nº 25.326):** garantiza la privacidad y seguridad de la información de los pacientes. (Ministerio de Justicia y Derechos Humanos, 1992)

- **Normas ISO (Internacional Organization for Standarization):** es un conjunto de estándares con reconocimiento internacional, creados con el objetivo de ayudar a las empresas a establecer niveles de cumplimiento de calidad, eficiencia y seguridad en relación con la gestión, prestación de servicios y desarrollo de productos en la industria.
  - **ISO 14971:2019 Dispositivos médicos/ productos sanitarios (MD) - Aplicación de la gestión del riesgo a los MD:** trata los procesos para gestionar los riesgos asociados con los MD. Los riesgos pueden estar relacionados con lesiones, no solo para el paciente, sino también para el usuario y otras personas. Los riesgos también pueden estar relacionados con daños a la propiedad (por ejemplo, objetos, datos, otros equipos) o al medio ambiente. (ISO, 2019)

- **IMDRF (International Medical Device Regulators Forum):** es un grupo de reguladores de dispositivos médicos alrededor del mundo. Su objetivo es establecer un entendimiento común del software pensado con fines médicos.
  - **Application of Quality Management System:** este documento resalta elementos la calidad de software y prácticas ingenieriles que refuerzan los principios de calidad de dispositivos médicos. Estos principios son la base de las buenas prácticas para mantener y controlar la calidad de los productos. (IMDR SaMD Working Group, 2015)

4.3. Ventajas y Desventajas de Modelos de Proceso de Desarrollo

4.3.1. Análisis cualitativo

Subcriterios planteados para el Análisis Cualitativo y Cuantitativo de los métodos de proceso a seleccionar:

Sub-criterio	Descripción
Tipo de software	Mediante este criterio se evaluará si el proceso de desarrollo es el adecuado para un software **socio-técnico crítico.**
Requisitos no funcionales	Mediante este criterio se evaluará sí conviene su implementación respecto a los requisitos no funcionales. (Requisitos no funcionales)
Experiencia del equipo de Desarrollo	Mediante este criterio se evaluará si el proceso de desarrollo es factible para el equipo de desarrollo, teniendo en cuenta nuestra experiencia y conocimientos.
Requisitos funcionales	Mediante este criterio se evaluará sí conviene su implementación respecto a los requisitos funcionales. (Requisitos funcionales)

Tabla 1.1: Subcriterios para el análisis cualitativo

Criterio	Sub-Criterio	Cascada	V-Model	Incremental	Espiral	Orientado a la Reutilización	Prototipos	RUP	OPEN UP
Preparación del equipo	Experiencia del equipo de Desarrollo	✔	✔	✔	✔	✔	✔	✔	✔
	Interés en aprender la metodología	✘	✔	✔	✔	✔	✔	✔	✔
Viabilidad del proyecto	Tipo de software	✔	✔	✘	✔	✔	✔	✔	✔
	Requisitos funcionales	✔	✔	✘	✘	✔	✘	✔	✔
	Requisitos no funcionales	✔	✔	✘	✔	✔	✔	✔	✔

Tabla 1.2: Cumplimiento de los subcriterios de análisis cualitativo para cada modelo de proceso

- **Cascada:**
  - **Interés en aprender la metodología**
    - El equipo de desarrollo ya llevó a cabo un proyecto aplicando esta metodología, por lo que los integrantes no mostramos interés en aplicar este método ya que consideramos que es un oportunidad para probar y conocer el funcionamiento de modelos todavía no aplicados.

- **Incremental:**
  - **Requisitos funcionales:**
    - Los requisitos del sistema son críticos, definidos e interdependientes, y consideramos que será difícil desarrollar incrementos del sistema, ya que debemos desarrollar gran parte del sistema para testeos antes de poder ser probado por usuarios.
  - **Requisitos no funcionales:**
    - El modelo de proceso de desarrollo incremental tiende a no enfocarse en el desarrollo de requerimientos no funcionales en las iteraciones, sino que se enfoca en requerimientos funcionales. Para la bomba de insulina son vitales los requerimientos no funcionales ya que aseguran el funcionamiento correcto del sistema y de la vida de las personas.
  - **Tipo de software:**
    - Los sistemas socio-técnicos crítico requieren de documentación detallada del sistema y de su funcionamiento debido a que está en riesgo directo la vida de las personas, los sistemas incrementales no se enfocan en la documentación y suele requerir de feedback constante del uso del sistema, lo cual resulta complejo en la bomba de insulina ya que esta debe desarrollarse y probarse antes de su prueba en personas.

- **Espiral:**
  - **Requisitos funcionales**
    - Si bien los requisitos y el sistema corresponde a un sistema crítico y el análisis de riesgo es importante para conocer los problemas que podríamos afrontar, dividir el sistema en módulos prototipables es complicado debido a la estrecha interdependencia de los criterios, lo que requiere un desarrollo sustancial antes de las pruebas de usuarios con la debida seguridad.

- **Prototipos:**
  - **Requisitos funcionales:**
    - Para el desarrollo de prototipos es importante poder definir módulos que puedan ser desarrollados y probados a través de prototipos simples, pero la bomba de insulina requiere de desarrollar un conjunto de módulos y la prueba de los mismos previo al desarrollo del prototipado.

4.3.2. Análisis cuantitativo

Definición de prioridad, grado de importancia y GIE (Grado de Importancia dado por el Equipo) para cada subcriterio:

Criterio	Subcriterio	Prioridad	Grado de Importancia	GIE
Viabilidad del proyecto	Tipo de Software	1	50	**0.5**
	Requisitos funcionales	2	40	**0.4**
	Requisitos no funcionales	3	30	**0.3**
Preparación del equipo	Interés en aprender la metodología	4	20	**0.2**
	Experiencia del equipo	5	10	**0.1**

Tabla 1.3: Subcriterios para el análisis cuantitativo

Escala de puntuación

Puntaje	Descripción
1	Regular
2	Bueno
3	Muy bueno
4	Excelente

Tabla 1.4: Escala de puntuación para el análisis cuantitativo

V-MODEL			
Criterio	**Subcriterio**	**Evaluación**	**Puntaje**
Viabilidad del proyecto	Tipo de Software	Debido a su enfoque en la reducción de errores, el control de calidad y la documentación detallada, el V-Model es adecuado para un sistema de software crítico para la salud.	4
	Requisitos funcionales	Uno de los cuatro submodelos del V-Model se encarga específicamente de los requisitos y criterios de calidad; y dado que es muy abstracto y personalizable, es adaptable a nuestro caso de estudio.	4
	Requisitos no funcionales	Uno de los cuatro submodelos del V-Model se encarga específicamente de los requisitos y criterios de calidad; y dado que es muy abstracto y personalizable, es adaptable a nuestro caso de estudio.	4
Preparación del equipo	Interés en aprender la metodología	Desarrollar el sistema siguiendo el V-Model nos ayudará a profundizar conocimientos sobre pruebas, verificación y validación, los cuales son conceptos que aún no pusimos en práctica.	4
	Experiencia del equipo	El equipo cuenta con experiencia en la implementación de modelos similares como el modelo en Cascada y ha realizado pequeñas iteraciones de V-Model en el desarrollo de los tps (Tp2 y Tp3).	3

MODELO ORIENTADO A LA REUTILIZACIÓN			
Criterio	**Subcriterio**	**Evaluación**	**Puntaje**
Viabilidad del proyecto	Tipo de Software	Las industrias que desarrollan sistemas socio técnicos críticos para la salud tienden a utilizar componentes que ya fueron probados y certificados. Sin embargo, existe mucha complejidad en la integración de componentes preexistentes al sistema y se debe contar con el código fuente de los componentes reutilizados.	3
	Requisitos funcionales	El modelo de reutilización de software puede ser limitante al momento de buscar soluciones creativas a requisitos específicos. Además, se debe evaluar exhaustivamente que los componentes a utilizar sean adecuados para los requisitos no funcionales del sistema en desarrollo.	1
	Requisitos no funcionales	La integración de componentes es recomendable cuando se realizan sistemas críticos ya que puede resultar difícil alcanzar altos niveles de confiabilidad y seguridad contra daños físicos.	2
Preparación del equipo	Interés en aprender la metodología	Debido a que este modelo es muy utilizado en la actualidad, el equipo de desarrollo tiene gran interés en aprender a implementarla.	4
	Experiencia del equipo	El equipo de desarrollo no tiene experiencia alguna en la implementación de esta metodología.	1

MODELO DE PROCESO UNIFICADO (RUP)			
Criterio	**Subcriterio**	**Evaluación**	**Puntaje**
Viabilidad del proyecto	Tipo de Software	El modelo RUP es una buena opción para sistemas que requieren una documentación detallada.Sin embargo, no tiene un enfoque particular en la disciplina de requerimientos ni pruebas del sistema, las cuales son esenciales para un sistema socio técnico crítico.	3
	Requisitos funcionales	Si bien contempla la disciplina de requerimientos, el modelo de proceso unificado no define precisamente cómo elicitar, identificar y documentar los requisitos, lo cual es fundamental en nuestro sistema.	2
	Requisitos no funcionales	Existen extensiones aplicables a Modelo que se enfocan en requerimientos no funcionales.	4
Preparación del equipo	Interés en aprender la metodología	RUP es un modelo más complejo y que exige una mayor cantidad de documentación y planificación que otros, lo que promueve buenas prácticas pero no es de gran interés para el equipo.	2
	Experiencia del equipo	El equipo no cuenta con experiencia en implementar RUP, pero conocemos a grandes rasgos el desarrollo de sistemas con OpenUP, que es basado en el proceso unificado.	2

MODELO OPEN UP			
Criterio	**Subcriterio**	**Evaluación**	**Puntaje**
Viabilidad del proyecto	Tipo de Software	El modelo OpenUP prioriza la eficiencia y la comunicación del equipo, pero no la confiabilidad del sistema ni la validación, que son esenciales en un sistema socio técnico crítico. Para adaptarse a la bomba de insulina, requeriría múltiples ajustes.	2
	Requisitos funcionales	OpenUP fomenta la documentación, lo que favorece la gestión y el seguimiento de los requisitos funcionales. Pero, al igual que RUP, no especifica precisamente las actividades de elicitación y especificación de requisitos.	2
	Requisitos no funcionales	Existen extensiones aplicables a Modelo que se enfocan en requerimientos no funcionales.	4
Preparación del equipo	Interés en aprender la metodología	El equipo ya conoce a grandes rasgos los principios y disciplinas de OpenUP, por lo que nos interesa enfocarnos en metodologías nuevas.	2
	Experiencia del equipo	Dado que en el cuatrimestre anterior aprendimos a aplicar el modelo caso de estudio, tenemos conocimientos básicos.	3

Tabla 1.5: Cumplimiento de los subcriterios del análisis cuantitativo para cada modelo de proceso

Puntaje dado por el equipo a los Modelos de Desarrollo y resultados:

Criterio	Subcriterio	GIE	Modelo V		Orientado a Reutilización		RUP		OpenUP	
			Puntaje	Valor	Puntaje	Valor	Puntaje	Valor	Puntaje	Valor
Viabilidad del proyecto	Tipo de Software	0.5	4	2.0	3	1.5	3	1.5	2	1.0
	Requisitos funcionales	0.4	4	1.6	1	0.4	2	0.8	2	0.8
	Requisitos no funcionales	0.3	4	1.2	2	0.6	4	1.2	4	1.2
Preparación del equipo	Interés en aprender la metodología	0.2	4	0.8	4	0.8	2	0.4	2	0.4
	Experiencia del equipo	0.1	3	0.3	1	0.1	2	0.2	3	0.6
Puntaje Total				**5.9**		**3.4**		**4.1**		**4**

Tabla 1.6: Puntuación para cada modelo de proceso

4.4. Selección del Modelo de Proceso de Desarrollo

Tras analizar diversos modelos de procesos de desarrollo de software y evaluar sus pros y contras a través de un cuadro comparativo en la página anterior, hemos determinado que el modelo V es la opción más adecuada para nuestro proyecto.

- **Tipo de Software:** El Modelo V ha obtenido la calificación más alta en este criterio, lo que indica que se ajusta mejor al tipo de software que se está desarrollando. Esto podría ser debido a su enfoque estructurado, que es ideal para sistemas donde la claridad en las fases de diseño y prueba es crítica.

- **Requisitos funcionales y no funcionales**: El Modelo V también ha recibido altas puntuaciones en cuanto a los requisitos funcionales y no funcionales, lo que demuestra su capacidad para manejar de manera efectiva tanto las características específicas del software como los criterios de operación general.

- **Preparación del equipo:**
  - **Interés en aprender la metodología:** El equipo ha mostrado un alto interés en aprender y aplicar la metodología del Modelo V, lo cual es crucial para su implementación exitosa.
  - **Experiencia del equipo:** Aunque la puntuación en experiencia es menor en comparación con otros criterios, el alto interés en aprender puede compensar este factor, permitiendo un desarrollo efectivo con el Modelo V.

- **Puntaje Total:** El Modelo V tiene el puntaje total más alto, lo que sugiere que, en general, es el mejor ajuste para las necesidades del proyecto en comparación con las otras metodologías evaluadas.

Considerando que el bomba de insulina es un sistema crítico en términos de seguridad contra daños físicos (safety-critical system), es esencial aplicar una rigurosidad excepcional en la especificación de los requerimientos. El modelo V, conocido por su enfoque sistemático en la verificación y validación en cada etapa del desarrollo, se adapta perfectamente a estas necesidades.

En este marco, proponemos la integración de herramientas de especificación de Modelos de Métodos Formales para una definición precisa y sin ambigüedades de los requerimientos críticos. Utilizaremos la Notación Z para describir el estado del sistema y sus operaciones, reduciendo así los errores en la detección de requerimientos. Esta medida garantizará la seguridad de los usuarios del sistema, buscando minimizar los impactos en los tiempos y costos de desarrollo.

El modelo V nos facilita una documentación estructurada y clara, permitiéndonos verificar el progreso del sistema de manera eficiente. Incorporando técnicas y herramientas de métodos formales, aseguraremos la seguridad que este tipo de sistema exige

Con su enfoque disciplinado y sistemático, el modelo V permite un manejo efectivo de responsabilidades y gestión en la producción de software.

En consecuencia, el equipo de desarrollo seguirá las fases del modelo V, desde el análisis de requerimientos hasta la validación y verificación. Durante la etapa de especificación de requerimientos, utilizaremos la Notación Z y las herramientas y técnicas necesarias para asegurar la claridad y precisión, entregando un software consistente y seguro, que respalde la salud de los usuarios.

5. Adaptación del Proceso de Desarrollo

Al haber sido seleccionado el modelo de desarrollo V, se adaptara y especificarán las disciplinas que serán realizadas, dado que dicho modelo de procesos de desarrollo no especifica el cómo llevarlas a cabo, como tampoco define las herramientas a ser utilizadas en el ciclo de vida del desarrollo.

Por lo tanto, el equipo definirá actividades, técnicas, conceptos y herramientas a ser utilizadas respetando el propósito del modelo V y de forma que pueda ser garantizado un sistema resultante confiable y seguro.

Las adaptaciones (actividades, técnicas, etc) realizadas al modelo de procesos de desarrollo en cada una de sus disciplinas estarán clasificadas por los atributos de calidad que se se busca contemplar en cada disciplina:

- **Confiabilidad *(dependability)*:** el sistema debe funcionar de manera confiable y precisa al entregar la cantidad correcta de insulina para contrarrestar el nivel actual de azúcar en sangre. Incluye tres sub propiedades (Sommerville, 2016, p. 288):
- **Fiabilidad *(reliability)***: el sistema debe proveer los servicios como es esperado.
    - **Corrección**: garantía de que los servicios del sistema son los especificados.
    - **Precisión**: garantía que la información se suministra con el nivel de detalle adecuado.
    - **Puntualidad**: garantizar que la información se suministra cuando se necesita.
- **Seguridad o protección contra daños físicos *(safety)*:** el sistema debe operar sin fallos catastróficos.
- **Seguridad o protección contra accesos no autorizados *(security)*:** el sistema debe protegerse ante intrusiones accidentales o deliberadas.
- **Resiliencia** *(resilience)*: es la capacidad de un sistema de mantener la continuidad de sus servicios críticos ante la presencia de eventos disruptivos, como fallas del sistema o ciberataques.(Sommerville, 2016, p. 409)
    - **Reconocimiento** *(recognition)*: es la capacidad del sistema de reconocer los síntomas de un problema que puede llevar a una caída del sistema. Idealmente, este reconocimiento debería ser posible antes de que ocurra la caída. (Sommerville, 2016, p. 410)

5.1. Disciplina de Requerimientos

5.1.1. Confiabilidad

Estrategias de análisis para formulación de requisitos no funcionales

Durante la especificación de requisitos de fiabilidad y seguridad contra daños físicos, se aplicarán las siguientes estrategias:

Atributo asociado al requisito	Estrategia de análisis	Utilización
Fiabilidad (reliability)	Probabilidad de fallo bajo demanda (POFOD) Probabilidad de que una demanda de servicio provoque una caída de sistema (Sommerville, 2011, p. 323)	POFOD = 0,001 equivale a una probabilidad de 1/1000 de que se produzca una caída cuando se realiza una demanda
	Tasa de ocurrencia de fallos (ROCOF) Número de caídas del sistema observables en un período de tiempo o en una cantidad de ejecuciones del sistema (Sommerville, 2011, p. 323)	ROCOF = 0,002 equivale a una probabilidad de 2 fallos cada 1000 ejecuciones
Seguridad contra daños físicos (safety)	Análisis de Árbol de Fallas *(Fault-Tree Analysis)* (Sommerville, 2011, p. 318)	Análisis de la cantidad de condiciones que pueden llevar a una administración incorrecta de insulina

Tabla 1.7: Estrategias de análisis a utilizar para obtener requisitos no funcionales.

Técnicas de validación

- **Especificación formal:** se incorporará la subactividad Especificación Formal a la actividad Especificación de Requisitos, donde se elaborarán especificaciones expresadas en lenguaje Z en la disciplina de requerimientos, con el objetivo de evitar ambigüedades en las especificaciones, aumentar la confiabilidad del sistema, y facilitar la detección de errores.
- **Gestión de requisitos:** se incorporará la gestión de requisitos a la disciplina de requerimientos dentro del submodelo Gestión de Configuraciones en el Modelo en V. Esta actividad será fundamental durante todo el proceso para garantizar que los cambios en los requisitos sean consistentes, estén reflejados en el desarrollo del sistema y sean informados a todas las partes interesadas.

5.1.2. Fiabilidad

Técnicas para la verificación de requisitos

Evaluación de la formulación de requisitos no funcionales de fiabilidad como métrica para la precisa y correcta determinación de su cumplimiento. En la subactividad de especificación de requisitos. (ver estrategias de análisis para formulación de requisitos no funcionales)

Requisitos de fiabilidad funcional

Se analizará e incluirán según las necesidades, requisitos de fiabilidad funcional en el documento de requisitos realizado en la actividad Especificación de Requisitos, durante la disciplina de requerimientos (Sommerville, 2016, p. 317):

- **Requisitos de comprobación** *(checking requirements):* identifican comprobaciones de las entradas para detectar errores antes de que sean procesadas por el sistema.
- **Requisitos de recuperación** *(recovery requirements):* orientados a ayudar al sistema a recuperarse después de que se haya producido una falla. Estos requisitos generalmente tienen que ver con mantener copias del sistema y sus datos y especificar cómo restaurar los servicios del sistema después de una falla.
- **Requisitos de redundancia** *(redundancy requirements)*: especifican características redundantes del sistema que garantizan que la falla de un solo componente no conduzca a una pérdida completa del servicio.
- **Requisitos de proceso** *(process requirements)*: se plantearán requisitos de buenas prácticas para la implementación del sistema, con el fin de evitar fallos en el proceso.

Enfoques complementarios de evitación, detección, corrección y tolerancia

Al momento de especificar los requisitos, se tendrán en cuenta los tres enfoques complementarios que se utilizan para mejorar la fiabilidad de un sistema (Sommerville, 2016, p. 308):

- **Evasión de fallas**: minimizar el uso de construcciones propensas a errores.

- **Detección y corrección de fallas**: diseñar procesos para descubrir y eliminar fallas en un programa.
- **Tolerancia a fallos**: diseñar el sistema para que los fallos que ocurran durante la ejecución sean gestionados de manera que no produzca caídas en el sistema.

5.1.3. Seguridad o protección contra accidentes físicos(safety)

Requisitos "No debe"

Se especificarán requerimientos del tipo "No debe" en la subactividad de la especificación de requisitos. Estos son requisitos funcionales de alto nivel y determinan los comportamientos que son considerados inaceptables en el sistema, a diferencia de los requerimientos funcionales normales que limitan el funcionamiento del sistema.

Punto de partida para requerimientos funcionales de seguridad que parte del conocimiento del dominio, estándares de seguridad, regulaciones, lo cual conduce a requerimientos de alto nivel que deben/pueden derivar en requerimientos más detallados (Sommerville, 2016, p. 344). Por ejemplo:

- El sistema **no debe** aplicar una dosis de insulina mayor al valor de la dosis máxima establecida.

Especificación de protección contra daños físicos

Se propone incorporar a la disciplina de análisis de requisitos la actividad de **especificación de protección** contra daños físicos.

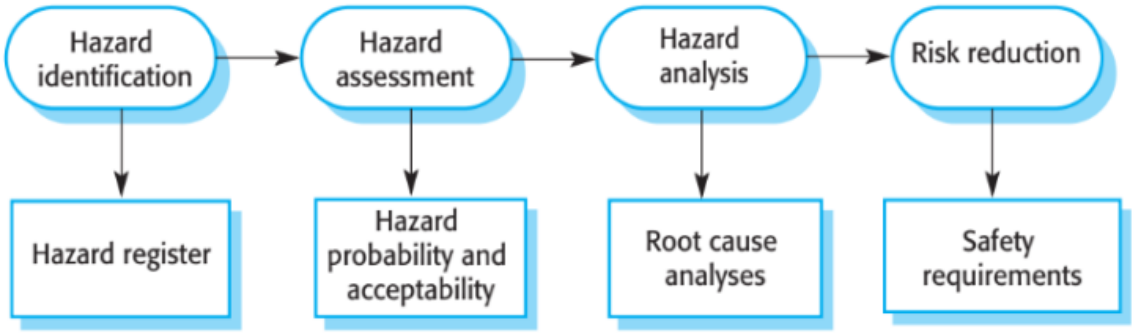


Figura 1.6: Modelo de Especificación Dirigida por Riesgos
(Sommerville, 2016, p. 345 Figure 12.2: Hazard-driven requirement specification)

Si bien el Modelo de proceso en V incluye disciplinas de pruebas de unidad, componentes y sistema, estas solo garantizan la fiabilidad del producto según lo especificado, pero no los posibles fallos de protección no considerados en las especificaciones de los requisitos. Además, define técnicas, herramientas o procedimientos para las especificaciones de requerimientos para lograr un sistema seguro.

Por lo tanto, el sistema puede ser fiable pero a su vez producir lesiones de forma directa, en caso de no aplicar técnicas para la especificación de requisitos que brinden protección. Las subactividades que serán incluidas en la especificación de protección son:

- **Identificación de peligros:** consiste en el reconocimiento de condiciones peligrosas que se puedan originar en el sistema y generar un accidente, considerando los requerimientos “No debe”, conocimientos del sistema, experiencia y utilización de clases de riesgos (Fallas de servicio, Eléctricos, Físico, Biológico, etc). (Sommerville, 2016, p. 345-346)
- **Evaluación del peligro:** consiste en la valoración del peligro, determinando los riesgos que derivan del peligro, especificando la probabilidad de ocurrencia y la severidad del peligro. (Sommerville, 2016, p. 346-347)
- **Análisis de peligro:** consiste en la descomposición del peligro en estados que lo conducen a su ocurrencia, hasta descubrir las causas raíces del mismo, que surgirán al no poder seguir descomponiendo. Esto puede ser realizado construyendo un árbol de fallas. (Sommerville, 2016, p. 349)
- **Reducción de peligro**: consiste en identificar los requerimientos de protección, que aseguran de que no surja un peligro o conduzca a un accidente o que si ocurre, el daño sea mínimo. (Sommerville, 2016, p. 351)

Enfoques complementarios de evitación, detección, eliminación y limitación del daño

Durante la obtención de requerimientos del sistema, se tendrán en cuenta tres estrategias complementarias para cerciorarse que no ocurrirán accidentes o serán mínimas las consecuencias (Sommerville, 2016, p. 342)

- **Evitar el peligro:** diseñar el sistema de modo que se eviten riesgos.
- **Detectar y eliminar el peligro**: diseñar el sistema de modo que los peligros se detecten y eliminen antes de causar un accidente.
- **Limitar el daño**: diseñar el sistema con características de seguridad contra daños físicos que minimicen el daño resultado de un accidente.

Bitácora de peligro

Es el documento central de seguridad ante daños físicos. Proporciona evidencia de cómo se consideraron los peligros identificados durante el desarrollo del software. Se propone agregar la actividad Elaborar Bitácora de Peligros, donde se documentará el proceso de análisis de cada peligro y los requerimientos que se generaron asociados a este, incluyendo: cómo, cuándo y quiénes trataron el peligro identificado durante la disciplina de requerimientos

.

5.1.4. Seguridad o Protección contra la Intrusión Accidental o Deliberada (Security Engineering)

Requerimientos de seguridad ante intrusiones deliberadas o accidentales

Debido a que el sistema Innopump no está conectado a la red, no es vulnerable a la mayoría de ataques maliciosos intencionados a los que están expuestos otros sistemas. Es por ello que se propone agregar una actividad de Especificar Requisitos de identificación, autenticación y autorización, donde incluirán sólo algunos tipos de requisitos de seguridad, asociados a los casos en que se le dé un uso inapropiado al sistema de forma accidental.

Será incluida en la subactividad especificación de requisitos de la disciplina de requerimientos.

- **Requisitos de identificación, autenticación y autorización:** consisten en la posibilidad de comprobar la identidad del usuario, contemplar la caducidad de sesión, de renovación de contraseña, etc. Es fundamental para el sistema Innopump poder validar la identidad del médico de cabecera si se quisiera realizar una configuración manual del dispositivo, de manera que un usuario no capacitado no pueda modificar el comportamiento del sistema. (Sommerville, 2016, p. 383)

Casos de uso inapropiado

Se propone modelar, junto a los casos de uso normal del sistema, otro tipo de casos llamados **casos de uso inapropiado**, durante la actividad Obtención de requisitos. Estos representarán interacciones maliciosas con el sistema, para identificar y representar posibles amenazas. Cada uno tendrá su especificación de caso de uso y serán asociados a la identificación del riesgo, y su análisis posterior.

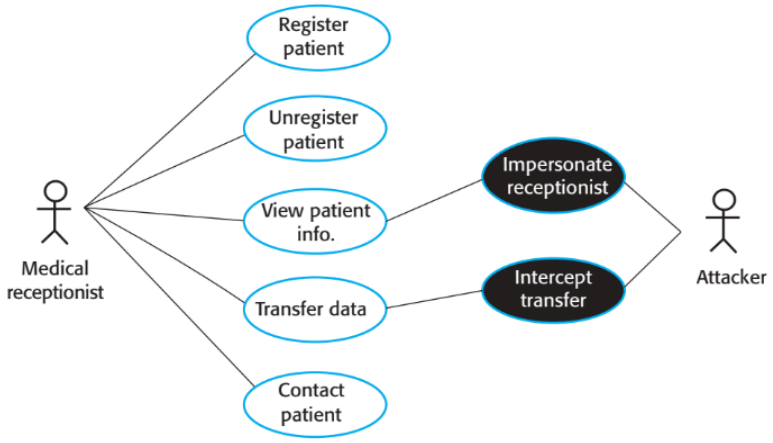


Figura 1.7: Ejemplo de casos de uso inapropiado del sistema Mentcare
(Sommerville, 2016, p. 387, Figure 13.8: Misuse cases)

5.1.5. Resiliencia

Requisitos de resiliencia

Se propone incorporar a la actividad de especificación requisitos de resiliencia del sistema. Estos se enfocarán principalmente en la evasión de la caída del servicio brindada por el sistema. Esta especificación de requisitos se corresponde a la primera de las cuatro “R”: Reconocimiento.

5.1.6. Resumen: Adaptaciones realizadas a la disciplina de requerimiento

CONFIABILIDAD	
Actividades/técnicas a ser añadidas	**Descripción**
Principales estrategias de análisis para formulación de requisitos no funcionales para su verificación	Implementar en la actividad de especificación de protección la reformulación de requisitos no funcionales de calidad, utilizando métricas y/o estrategias de análisis para derivar los requisitos.
Técnicas de validación	- Agregar la actividad Gestión de requisitos durante toda la Disciplina de Requerimientos - Agregar la actividad Especificación Formal
FIABILIDAD	
Actividades/técnicas a ser añadidas	**Descripción**
Requisitos de fiabilidad funcional	Agregar la subactividad de evaluación de requisitos de fiabilidad funcional a la subactividad reducción de riesgos de la actividad Especificación de protección.
Enfoques complementarios de evitación, detección, corrección y tolerancia	Agregar la subactividad estrategias de evitación de fallas, detección y corrección de fallas y tolerancia a fallas en la subactividad de evaluación de requisitos de fiabilidad funcional

SEGURIDAD O PROTECCIÓN CONTRA DAÑOS FÍSICOS (SAFETY)	
Actividades/técnicas a ser añadidas	**Descripción**
Requerimientos “No debe”	Agregar la actividad identificación de requisitos “no debe” en la disciplina de requerimientos, antes y su resultado como entrada de la actividad Especificación de protección
Especificación de protección	Agregar la actividad Especificación de Protección incluyendo las subactividades: Identificación de peligros, Evaluación de peligros, Análisis de peligros y Reducción de peligros.
Bitácoras de peligro	Agregar la actividad Elaborar Bitácora de Peligro en la subactividad Identificación de riesgos
Enfoques complementarios de evasión, detección, eliminación y limitación del daño	Agregar la subactividad estrategias evitación, detección, eliminación y limitación de daño en la actividad especificación de protección en la subactividad reducción de riesgos
SEGURIDAD O PROTECCIÓN CONTRA INTRUSIÓN ACCIDENTAL O DELIBERADA	
Requisitos de autenticacion y autorizacion	Se evaluará la implementación de estos requisitos en la subactividad casos de uso inapropiado.
Casos de uso inapropiado	Se implementará como subactividad en la actividad Elaboración de casos de uso.

RESILIENCIA	
Actividades/técnicas a ser añadidas	**Descripción**
Requisitos de resiliencia	Agregar subactividad evaluación requisitos de resiliencia en la subactividad reducción de riesgos

Tabla 1.8: Mejoras propuestas en la Disciplina de Requerimientos

5.2. Disciplina de Diseño arquitectónico

5.2.1. Confiabilidad

Evaluación de aplicación de redundancia y diversidad

Se agrega la actividad refinamiento de arquitectura en la disciplina de diseño arquitectónico que incluirá la subactividad evaluación de de aplicación de redundancia y diversidad (Sommerville, 2016, p. 295):

- **Redundancia**: consiste en incluir más de una sola versión de ciertos componentes, de manera que ante una falla, otra estará disponible de modo que el servicio seguirá activo.
- **Diversidad**: consiste en aplicar funcionalidades de sistema de diferentes maneras, con diferentes componentes, de manera que no fallen de la misma forma.

Aplicación de técnicas de validación

- **Modelado del sistema:** se realizará el Modelo 4+1 del sistema en la actividad Modelado del Sistema de la disciplina de diseño arquitectónico, de manera que los requisitos sean trazables y vinculables a los casos de uso correspondientes, y su representación en cada una de las vistas del modelo.

5.2.2. Fiabilidad

Patrones arquitectónicos

Se evaluará en la subactividad de identificación de enfoques arquitectónicos los patrones arquitectónicos genéricos para la fiabilidad tolerante a fallos:

- **Programación de N Versiones:** La programación de N versiones consiste en desarrollar software tolerante a fallas donde N versiones diversas de un sistema se ejecutan en paralelo. En este enfoque, varios equipos implementan el mismo sistema en computadoras separadas. (Sommerville, 2016, p. 322)

  Sin embargo, esto implica al menos tres versiones del sistema, preferiblemente elaboradas por distintos equipos de desarrollo. Consideramos que es muy difícil aplicar esto al desarrollo de Innopump debido a que somos un equipo pequeño de desarrollo y con plazos límites de tiempo.

- **Automonitoreo**: los sistemas con una arquitectura de automonitoreo se encargan de controlar su propio funcionamiento y tomar una acción si se detecta un problema. El sistema dispondrá de más de un canal que realizará los cálculos y comparará los resultados: si las salidas difieren, se supone una caída y se lleva al sistema a un estado seguro *(fail-safe state)*.
  Esta arquitectura implica utilizar software y hardware diverso en cada canal, para reducir la probabilidad de que se repita el error en ambos canales. (Sommerville, 2016, p. 320)

5.2.3. Seguridad o protección contra accidentes físicos(safety)

Enfoques complementarios de evitación, detección, eliminación y limitación del daño

Durante la evaluación y refinamiento de la arquitectura del sistema, se tendrá en cuenta los requisitos elaborados en base a las estrategias complementarias de evitación, detección, eliminación y limitación del daño.

5.2.4. Seguridad o protección contra la intrusión Accidental o Deliberada (Security Engineering)

Uso de casos de uso inapropiado

Durante la recopilación de escenarios para la selección de arquitectura basada en ATAM, dentro de la disciplina de diseño arquitectónico, se tendrán en cuenta los casos de uso inapropiados generados durante la disciplina de requerimientos. (Ver Disciplina de Requerimientos para Ingeniería de Seguridad contra la Intrusión Accidental o Deliberada).

Evaluación de riesgos de diseño

La evaluación de riesgos de diseño propuesta por Sommerville (Sommerville, 2016, p. 389) consiste en la toma de decisiones durante el diseño arquitectónico que influyen en la seguridad del sistema.

Los riesgos identificados durante la disciplina de requisitos serán tomados en cuenta durante la actividad Recopilar requisitos y restricciones del entorno, e incluidos en la identificación de riesgos, puntos de sensibilidad y *trade-offs*.

5.2.5. Resiliencia

El error humano

Los errores humanos siempre estarán presentes, a menos que se tenga un sistema completamente automatizado. Este problema de los errores humanos puede ser visto de dos maneras:

- **Enfoque humano**: propone que los errores son responsabilidad directa de la persona que actúa de manera descuidada.
- **Enfoque de sistema**: consiste en que la gente puede fallar y va a cometer errores, por lo que el sistema debe reconocer la posibilidad del error humano.

Al momento del diseño arquitectónico de Innopump, se considerará el **enfoque de sistema**, debido a que culpar y castigar a la persona que comete un error no hará que el sistema sea más seguro. Por lo tanto, la posibilidad del error humano deberá ser tenida en cuenta, y el sistema será diseñado de forma que presente la menor cantidad de vulnerabilidades ante estos errores, para reducir la probabilidad de una caída del sistema.

Identificación de servicios críticos para el diseño de sistemas resilientes

Los servicios críticos son aquellos que permiten al sistema cumplir con su propósito principal. En el caso de Innopump, el objetivo principal es medir el nivel de glucosa en sangre y administrar la dosis de insulina correspondiente lo más precisamente posible. Por lo tanto, los servicios principales podrían ser, por ejemplo, la medición, el cálculo y la administración de insulina, mientras que el registro del historial de mediciones no resulta crítico para su funcionamiento. La identificación de servicios críticos es fundamental para la planificación de recuperación y rehabilitación del sistema ante una caída.

Esta subactividad de Identificación de servicios críticos será realizada durante la actividad Identificación de riesgos, puntos de sensibilidad y *trade-offs* durante la disciplina de requerimientos.

5.2.6. Resumen: Adaptaciones realizadas a la disciplina de diseño arquitectónico

CONFIABILIDAD	
Actividades/técnicas a ser añadidas	**Descripción**
Redundancia y diversidad	Agregar la actividad Refinamiento de Arquitectura Agregar la subactividad Evaluación de Aplicación de Redundancia y Diversidad a la actividad de Refinamiento de Arquitectura
FIABILIDAD	
Actividades/técnicas a ser añadidas	**Descripción**
Arquitecturas tolerantes a fallos	Se considerarán en la subactividad de refinamiento arquitectónicos
SEGURIDAD O PROTECCIÓN CONTRA DAÑOS FÍSICOS (SAFETY)	
Actividades/técnicas a ser añadidas	**Descripción**
Enfoques complementarios de evitación, detección, eliminación y limitación del daño	Se tendrán en cuenta los requisitos de evitación, detección, corrección y limitación del daño durante las actividades de Refinamiento de Arquitectura.
SEGURIDAD O PROTECCIÓN CONTRA LA INTRUSIÓN ACCIDENTAL O DELIBERADA(SECURITY)	
Evaluación de riesgos de diseño	Se tendrán en cuenta los riesgos identificados en la Disciplina de Requerimientos durante la actividad Recopilar requisitos y restricciones del entorno, e incluidos en la identificación de riesgos, puntos de sensibilidad y *trade-offs.*

Casos de uso inapropiado	Se tendrán en cuenta los casos de uso inapropiados elaborados en la Disciplina de Requerimientos durante la actividad Recopilar escenarios.
RESILIENCIA	
Actividades/técnicas a ser añadidas	**Descripcion (donde)**
El error humano	Se tendrá en cuenta el enfoque de sistema ante errores humanos durante la actividad Refinamiento de la Arquitectura
Identificación de servicios críticos	Agregar la subactividad Identificación de servicios críticos a la actividad Identificación de riesgos, puntos de sensibilidad y *trade-offs*.

Tabla 1.9: Mejoras propuestas en la disciplina de diseño arquitectónico

5.3. Disciplina de implementación

5.3.1. Confiabilidad

Aplicación de técnicas de validación

Se propone agregar una actividad de Validación durante la Implementación del sistema, conteniendo las subactividades (utilizada para la seguridad contra daños físicos):

- **Inspecciones de diseño y programación:** se propone incorporar como subactividad la Revisión de Diseño y Programación del sistema durante la implementación.
- **Análisis estático:** se propone incorporar como subactividad el Análisis Estático del código del programa, con el objetivo de hallar y corregir errores u omisiones durante la programación.

5.3.2. Fiabilidad

Políticas de diversidad de software

Para el desarrollo de sistemas de alta fiabilidad, se especificarán cuáles de las políticas de diversidad explícitas recomendadas por Sommerville (Sommerville, 2016, p. 324) se implementarán para maximizar la diversidad en el sistema:

- **Métodos de diseño:** se especificará que el equipo de desarrollo plantee soluciones a un mismo problema utilizando diferentes paradigmas o enfoques de diseño, por ejemplo: orientado a objetos, estructurado, orientado a eventos.
- **Lenguajes de programación:** se estipulará que los programas deberán implementarse en diferentes lenguajes.
- **Herramientas:** se requerirá el uso de diferentes entornos de desarrollo para la implementación de distintos módulos o canales del sistema.
- **Algoritmos:** se requerirá el uso de diferentes algoritmos para algunos procesos en la implementación.

Programación para la confiabilidad

Se aplicarán durante la implementación el conjunto de buenas prácticas de programación para el desarrollo de sistemas, recomendado por Sommerville (Sommerville, 2016, p. 326) para sistemas confiables:

1) Controlar la visibilidad de la información en un programa: al programar, se debe adoptar un principio para controlar el acceso a variables y estructuras de datos que utiliza cada componente del programa.
2) Verificar la validez de todas las entradas: la especificación del sistema debe definir qué acciones debe tomar si la entrada es incorrecta, y siempre debe comprobar su validez tan pronto como sean leídas.
3) Proporcionar un controlador *(handler)* para todas las excepciones: para garantizar que las excepciones del programa no causen fallas en el sistema, se debe definir un controlador de excepciones para todas las que puedan surgir.

4) Minimizar el uso de construcciones propensas a errores: algunas prácticas como el uso de números de punto flotante y la asignación de almacenamiento dinámico pueden provocar errores en el tiempo de ejecución.
5) Proporcionar capacidades de reinicio: el sistema debe proporcionar una capacidad de reinicio basada en mantener copias de los datos recopilados o generados durante el procesamiento.
6) Comprobar los límites de la matriz: si se está utilizando un lenguaje que no incluye la verificación de límites de arreglos, se deben incluir comprobaciones de que el índice de la matriz esté dentro de los límites.
7) Incluir tiempos de espera al llamar a componentes externos: para llamadas entre componentes de un sistema, se debe incluir un tiempo de espera para la respuesta, debido a que existe la posibilidad de que este falle.
8) Nombrar todas las constantes que representan valores del mundo real: cuando se utilicen constantes, se debe hacer referencia a ellas por su nombre en lugar de por su valor.

Dependable programming guidelines

1. Limit the visibility of information in a program.
2. Check all inputs for validity.
3. Provide a handler for all exceptions.
4. Minimize the use of error-prone constructs.
5. Provide restart capabilities.
6. Check array bounds.
7. Include timeouts when calling external components.
8. Name all constants that represent real-world values.

Figura 1.8: Buenas prácticas para la programación de sistemas confiables (Sommerville, 2016, p. 326 Figure 11.11: Good practice guidelines for dependable programming)

5.3.3. Seguridad o protección contra accidentes físicos(safety)

Revisión de protección

La revisión de protección consiste en actualizar o ajustar las especificaciones de protección, al identificar nuevos potenciales peligros durante el proceso de desarrollo del sistema: al ejecutar análisis estáticos, inspecciones de programación y de diseño.

- **Análisis estático:** es la utilización de herramientas de análisis automatizado del código fuente de un programa, con el objetivo de detectar anomalías en el programa, como variables sin inicialización o cuyos valores pueden estar fuera del rango.

Fault class	Static analysis check
Data faults	Variables used before initialization Variables declared but never used Variables assigned twice but never used between assignments Possible array bound violations Undeclared variables
Control faults	Unreachable code Unconditional branches into loops
Input/output faults	Variables output twice with no intervening assignment
Interface faults	Parameter type mismatches Parameter number mismatches Nonusage of the results of functions Uncalled functions and procedures
Storage management faults	Unassigned pointers Pointer arithmetic Memory leaks

Figura 1.9: Comprobaciones de análisis estático automatizadas (Sommerville, 2016, p. 360 Figure 12.9: Automated static analysis checks)

- **Inspecciones de programación y diseño:** consiste en que el equipo de desarrollo comprueba cuidadosamente si se han cometido errores, asunciones u omisiones que pueden afectar la seguridad ante daños físicos de un sistema. Debe ser guiado por los riesgos identificados durante la disciplina de requerimientos.

5.3.4. Resumen: Adaptaciones realizadas a la disciplina de implementación

CONFIABILIDAD	
Actividades/técnicas a ser añadidas	**Descripción**
Técnicas de validación	Agregar actividad Validación Agregar subactividad Inspección de diseño y programación a la actividad de Validación Agregar subactividad Análisis Estático a la actividad de Validación
FIABILIDAD	
Actividades/técnicas a ser añadidas	**Descripción**
Políticas de diversidad de Software	Se tendrán en cuenta las políticas de diversidad de software recomendadas por Sommerville al momento de la programación.
Programación para la confiabilidad	Se aplicarán durante la implementación el conjunto de ocho buenas prácticas *(Guidelines)* recomendado por Sommerville
SEGURIDAD O PROTECCIÓN CONTRA DAÑOS FÍSICOS (SAFETY)	
Actividades/técnicas a ser añadidas	**Descripción**
Revisión de protección	Ver Mejora de Implementación para la Confiabilidad

Tabla 1.10: Mejoras propuestas en la disciplina de implementación

5.4. Disciplinas de pruebas

5.4.1. Confiabilidad

Aplicación de técnicas de validación

- **Planificación y gestión de pruebas:** al modelo de disciplina de pruebas planteado se propone incorporar la gestión de pruebas, incluyendo actividades de diseño y ejecución de las mismas.

5.4.2. Fiabilidad

Métricas de fiabilidad

Se incorporarán en la disciplina de pruebas la **generación de reportes** de **la probabilidad de fallos bajo demanda (POFOD)** como métrica de fiabilidad que tendrá el sistema resultante. (Sommerville, 2016, p. 313-314)

El Modelo de proceso en V cuenta con las disciplinas de pruebas de unidad, de componentes y de sistema, donde se ejecutarán pruebas de las funcionalidades del sistema bajo diferentes entradas y condiciones derivadas de los requerimientos especificados para verificar y validar con los mismos. Se especificarán en las pruebas del sistema las **probabilidades de ocurrencia de diferentes fallos** en el sistema (ejecuciones bajo condiciones de fallo), además de las condiciones (definidas en casos de prueba) derivadas de las especificaciones, que fueron aprobadas por el sistema.

5.4.3. Seguridad o protección contra daños físicos(safety)

Argumentos estructurados

Se incorporará a la disciplina de verificación del sistema la actividad de Elaboración de argumentos estructurados de seguridad (basados en contradicciones). Estos se utilizan para tomar la decisión de si el sistema es seguro o no. Consiste en demostrar que la ejecución de un programa no derivará en un estado inseguro (Sommerville, 2011, p. 414).

Pasos a realizar:

1) Suponer que un estado inseguro (identificado en el análisis de peligro durante la disciplina de requerimientos) puede ser alcanzado.
2) Escribir la expresión lógica que define este estado inseguro.
3) Analizar el modelo de sistema y comprobar que, para todas las rutas que conduzcan al estado inseguro, se contradicen las condiciones.
4) Repetir el análisis para cada peligro identificado, para garantizar que el sistema será seguro.

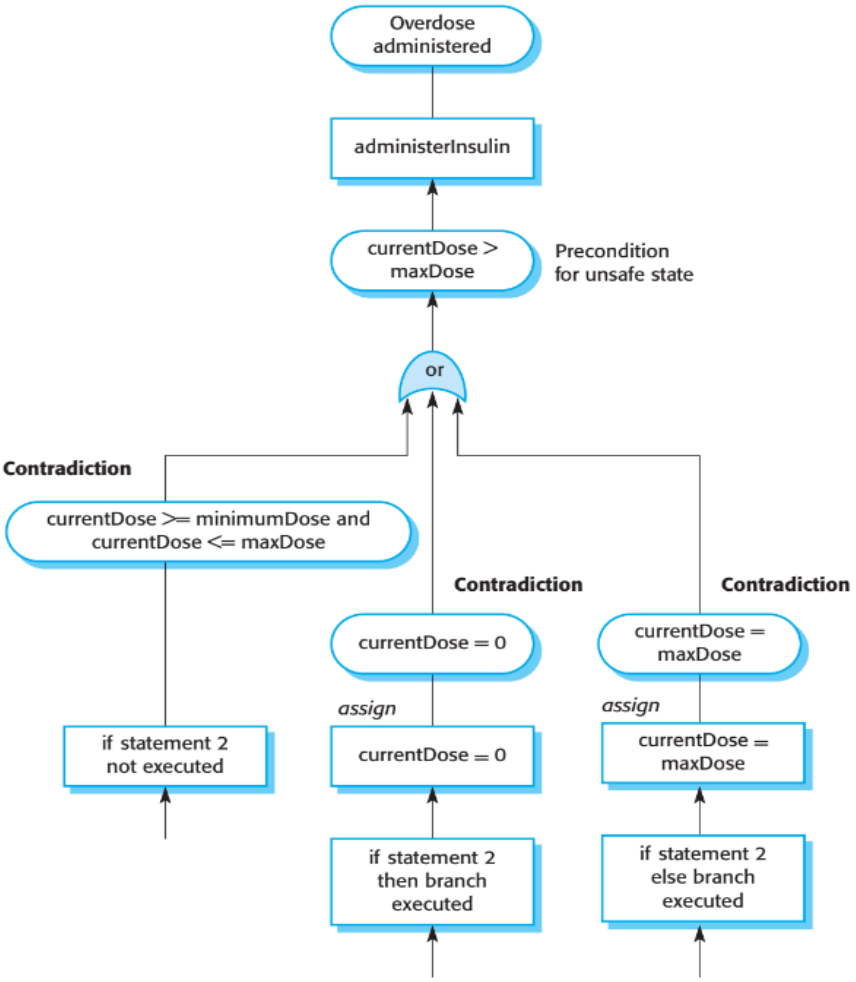


Figura 1.10: Ejemplo de argumentos de protección para la bomba de insulina personal (Sommerville, 2011, p. 367, Figura 12.14: Informal safety argument based on demonstrating contradictions)

Casos de seguridad

Se propone agregar una actividad Elaboración de Casos de Seguridad, donde se diseñarán casos de seguridad *(safety)*, también llamados casos de aseguramiento, luego de ejecutar las pruebas de sistema durante la disciplina de pruebas *(testing)*. Estos casos son recopilaciones de la evidencia documentada que ofrece un argumento convincente y válido de que un sistema está adecuadamente protegido para una aplicación dada en un entorno determinado. (Sommerville, 2011, p. 411)

Cada caso de seguridad hará referencia a la documentación del análisis de peligro, los requisitos asociados, la especificación de protección realizada y los argumentos de protección elaborados para los posibles estados inseguros.

Chapter	Description
System description	An overview of the system and a description of its critical components.
Safety requirements	The safety requirements taken from the system requirements specification. Details of other relevant system requirements may also be included.
Hazard and risk analysis	Documents describing the hazards and risks that have been identified and the measures taken to reduce risk. Hazard analyses and hazard logs.
Design analysis	A set of structured arguments (see Section 12.4.1) that justify why the design is safe.
Verification and validation	A description of the V & V procedures used and, where appropriate, the test plans for the system. Summaries of the test results showing defects that have been detected and corrected. If formal methods have been used, a formal system specification and any analyses of that specification. Records of static analyses of the source code.
Review reports	Records of all design and safety reviews.
Team competences	Evidence of the competence of all of the team involved in safety-related systems development and validation.
Process QA	Records of the quality assurance processes (see Chapter 24) carried out during system development.
Change management processes	Records of all changes proposed, actions taken, and, where appropriate, justification of the safety of these changes. Information about configuration management procedures and configuration management logs.
Associated safety cases	References to other safety cases that may impact the safety case.

Figura 1.11: Contenido de un caso de protección
(Sommerville, 2016, p. 362 Figure 12.10: Possible contents of a software safety case)

5.4.4. Seguridad o protección contra la intrusión accidental o deliberada(security)

Pruebas de seguridad

La revisión de la seguridad del sistema es fundamental para poder afirmar que el sistema será seguro. Los requisitos de seguridad ante la intrusión accidental o deliberada serán requisitos “No debe”, al igual que algunos requisitos de seguridad contra daños físicos (Ver Disciplina de Requisitos para Ingeniería de la Seguridad contra Daños Físicos). Es decir, que los requisitos especificarán lo que no debe suceder en el sistema. Para probar estos requisitos, se diseñarán y ejecutarán pruebas de penetración del sistema *(penetration testing)* durante las actividades Diseñar pruebas de sistema y Ejecutar pruebas de sistema dentro de la disciplina de pruebas:

Las pruebas de penetración del sistema consisten en que el equipo de validación intentará romper las barreras del sistema, simulando ataques y usando su ingenio para comprometer la seguridad del mismo.

5.4.5. Resumen: Adaptaciones realizadas en la disciplinas de pruebas

CONFIABILIDAD	
Actividades/técnicas a ser añadidas	**Descripción**
Técnicas de validación	Agregar actividad de Planificación y Gestión de Pruebas a toda la Disciplina
Fiabilidad	
Actividades/técnicas a ser añadidas	**Descripción**
Métricas de fiabilidad	Se incorporará la subactividad de Generación de Reportes de Fiabilidad en la actividad Ejecutar Pruebas
SEGURIDAD O PROTECCIÓN CONTRA DAÑOS FÍSICOS (SAFETY)	
Actividades/técnicas a ser añadidas	**Descripción**
Argumentos estructurados	Se incorporará la actividad Elaboración de Argumentos Estructurados en la actividad Definir Casos de Prueba
Casos de seguridad	Se incorporará la subactividad Elaboración de Casos de Seguridad en la actividad Definir Casos de Prueba de unidad y la subactividad actualizar casos de prueba de seguridad en las actividades Casos de Prueba de Componente y Casos de Prueba de Sistema

SEGURIDAD O PROTECCIÓN CONTRA LA INTRUSIÓN ACCIDENTAL O DELIBERADA(SECURITY)	
Actividades/técnicas a ser añadidas	**Descripción**
Pruebas de seguridad	Agregar subactividad Diseñar Pruebas de Seguridad a la actividad Definir Casos de Pruebas de Sistema. Agregar subactividad Ejecutar Pruebas de Seguridad a la actividad Ejecutar Pruebas de Sistema.

Tabla 1.11: Mejoras propuestas en la disciplina de pruebas

6. Meta-Modelo del Proceso de Desarrollo Global

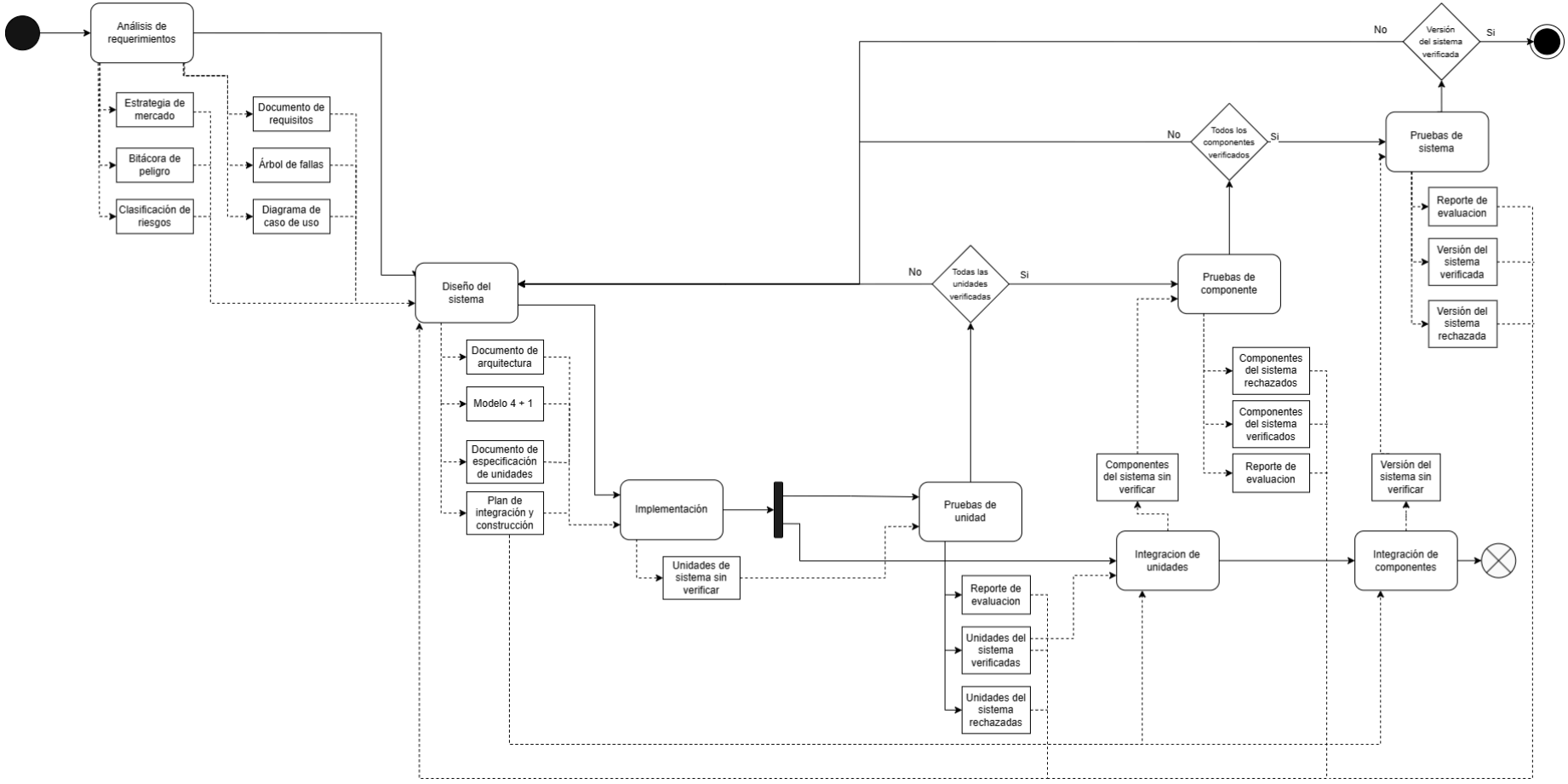


Figura 1.12: Meta-Modelo del Proceso de Desarrollo. Basado en V-Model (Bucanac & University of Karlskrona/Ronneby, 1999) Diagrama de Actividad (UML 2.5.1) link

Actividades

- **Análisis de requerimientos:** se realiza el análisis de mercado del sistema a construir, así como la recolección y definición de requerimientos funcionales y no funcionales del sistema. La validación y verificación de los mismos por medio del análisis de mercado y requisitos de usuario.

- **Diseño de sistema:** recibiendo como entrada el análisis de mercado y el documento de requerimientos, se realiza el análisis y diseño de la arquitectura del sistema obteniendo el documento arquitectónico que describe los modelos y las diferentes vistas de la arquitectura. Describe los diferentes componentes y sus interacciones
- **Implementación:** recibiendo como entrada la especificación de las unidades y el plan de integración, se procede a realizar la construcción del código fuente de los mismos obteniendo las unidades del sistema sin verificar.

- **Integración de unidades:** recibiendo el plan de integración y construcción como las unidades del sistema sin verificar, se realiza en paralelo con las pruebas de unidad, la integración de las unidades de componentes que se hayan verificado correctamente, obteniendo los componentes del sistema sin verificar.

- **Integración de componentes:** recibiendo el plan de integración y construcción como los componentes del sistema sin verificar, se realiza en paralelo con las pruebas de componente, la integración de los mismo, que se hayan verificado correctamente, obteniendo la versión del sistema sin verificar.

- **Pruebas de unidad:** recibiendo la unidad del sistema sin verificar, se construirán los diferentes casos de pruebas para esta unidad, dichos casos de pruebas serán documentados en el plan de evaluaciones y ejecutadas obteniendo el reporte de evaluación como las unidades del sistema aceptadas y rechazadas en tal caso volverá a diseño de módulos donde será redefinida o corregida.

- **Pruebas de componente:** recibiendo los componentes del sistema sin verificar, se construirán los diferentes casos de prueba para este componente, estos casos serán documentados en el plan de evaluación y ejecutadas obteniendo el reporte de evaluación como los componentes del sistema aceptadas y rechazadas en tal caso volverá a diseño de sistema donde será redefinida o corregida.

- **Pruebas de sistema:** Con la versión del sistema sin verificar, se construirán los diferentes casos de prueba, estos casos serán documentados en el plan de evaluación y ejecutadas obteniendo el reporte de evaluación como la versión del sistema aceptada o rechazada en tal caso volverá a diseño de sistema donde será redefinida o corregida.

Artefactos

- **Análisis de requerimientos:**
  - **Documento de estrategias de mercado(negocio):** describe los estudios realizados de otros modelos, las ventajas y desventajas de los modelos más utilizados, un análisis de la competencia, las necesidades y los deseos de los usuarios.
  - **Bitácora de peligro:** es el documento central de seguridad ante daños físicos. Proporciona evidencia de cómo se consideraron los peligros identificados durante el desarrollo del software.
  - **Documento clasificación de riesgos:** documento donde clasificamos los riesgos bajo los criterios de : probabilidad de riesgo,severidad del riesgo, riesgo estimado y aceptabilidad.
  - **Documento de requerimientos del sistema:** resultado del proceso de ingeniería de requisitos, es la descripción de lo que debería hacer el sistema y de los requisitos no funcionales claves que el sistema debería tener.
  - **Documento árbol de fallas:** documento consiste en la descomposición del peligro en estados que lo conducen a su ocurrencia, hasta descubrir las causas raíces del mismo, que surgirán al no poder seguir descomponiendo.
  - **Diagrama de caso de uso:** documento con los principales casos de usos diagramados para el software desarrollado

- **Diseño del sistema:**
  - **Documento arquitectónico:** resultado final del Diseño Arquitectónico, que será fundamental para el mantenimiento y escalabilidad del sistema a largo plazo.
  - **Modelo 4+1:** describe las diferentes vistas o perspectivas : lógica, de procesos, de desarrollo, física y conceptual, de la arquitectura del sistema que se consideren necesarias para llevar a cabo la construcción del sistema de forma correcta ilustrando de forma completa el sistema.
  - **Documento de especificación de unidades:** describe la identificación y descomposición de aquellos componentes que se consideren necesario y la especificación detallada de cada una de las unidades.

- **Implementación:**
  - **Plan de integración y construcción:** describe los procedimientos a llevar a cabo para la correcta construcción e integración de unidades y componentes del sistema. Como las interfaces por las cuales se comunicarán las diferentes unidades y componentes para llevar a cabo la integración.
  - **Unidades del sistema sin verificar:** son aquellos métodos y funciones individuales de las clases, componentes o módulos que utiliza el software.

- **Integración de unidades:**

    - **Componentes del sistema sin verificar:** son los módulos o partes que componen a un sistema y realizan un conjunto de tareas específicas.

- **Integración de componentes:**
    - **Versión del sistema sin verificar:** es una instancia específica del software que ha sido modificada, actualizada o desarrollada para cumplir con ciertos requisitos o mejoras.

- **Pruebas:**
    - **Plan de evaluaciones:** es un documento en el cual están definidos los métodos, criterios, ambiente y casos de prueba de las evaluaciones.
    - **Reporte de evaluaciones:** es un documento en el cual están documentados los resultados de las evaluaciones, junto con las causas de rechazo de los productos los cuales se utilizaran para posteriores re-desarrollos de los productos.
    - **Unidades del sistema (verificadas/rechazadas)**: son aquellos métodos y funciones individuales de las clases, componentes o módulos que utiliza el software.
    - **Componentes del sistema (verificadas/rechazadas):** son los módulos o partes que componen a un sistema y realizan un conjunto de tareas específicas.
    - **Versión del sistema (verificadas/rechazadas):** es una instancia específica del software que ha sido modificada, actualizada o desarrollada para cumplir con ciertos requisitos o mejoras.

7. Meta-Modelo del Proceso de Disciplina de Requerimientos

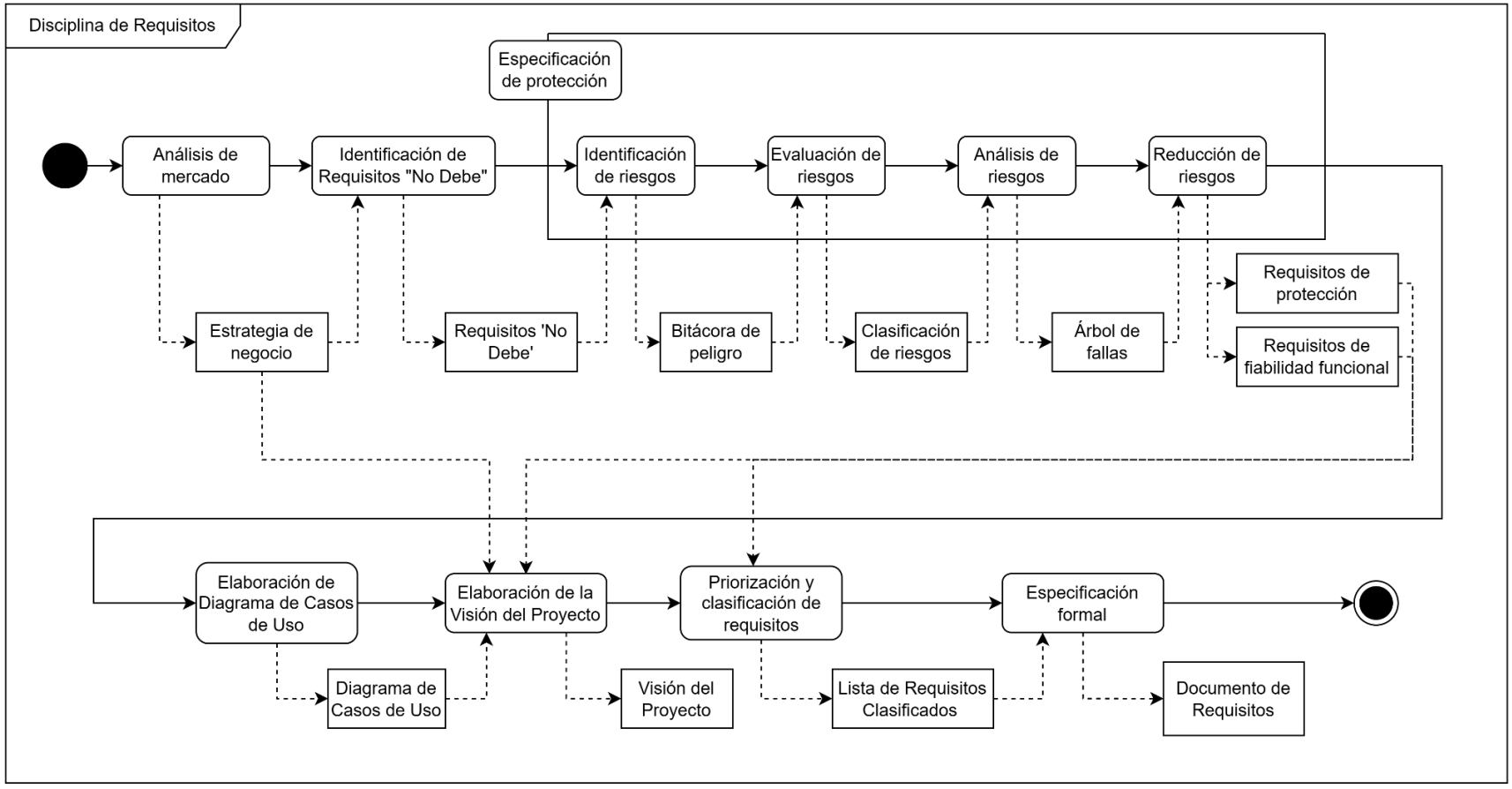


Figura 1.13: Meta-Modelo del Proceso de Ingeniería o Disciplina de Requerimientos
Diagrama de Actividad (UML 2.5.1)

Actividades

- **Análisis de mercado:** tiene como objetivo analizar los productos similares en el mercado, identificar sus ventajas, desventajas, los requisitos funcionales y no funcionales de usuario y sistema asociados a estos y la oportunidades de innovación o mejora aplicables al producto a desarrollar.
- **Identificación de requisitos No Debe**: consiste en la identificación de requisitos funcionales de alto nivel que describen los comportamientos que son considerados inaceptables en el sistema, partiendo del conocimiento del dominio, estándares de seguridad, regulaciones, leyes y políticas de seguridad, y del análisis de mercado realizado.
- **Especificación de protección**
    - **Identificación de peligros:** consiste en el reconocimiento de condiciones peligrosas que se puedan originar en el sistema y generar un accidente, considerando los requerimientos “No debe”, conocimientos del sistema, experiencia y utilización de clases de riesgos (Fallas de servicio, Eléctricos, Físico, Biológico, etc). Estas serán definidas en bitácoras de peligro (ver detalle). (Sommerville, 2016, p. 345-346)
    - **Evaluación del peligro:** consiste en la valoración de los peligros identificados, determinando los riesgos que derivan del peligro, especificando la probabilidad de ocurrencia y la severidad del peligro. Esto impacta en un artefacto Clasificación de riesgos. (Sommerville, 2016, p. 346-347)
    - **Análisis de peligro:** consiste en la descomposición del peligro en estados que lo conducen a su ocurrencia, hasta descubrir las causas raíces del mismo, que surgirán al no poder seguir descomponiendo. Será realizado construyendo un árbol de fallas. (Sommerville, 2016, p. 349)
    - **Reducción de peligro**: consiste en identificar los requerimientos de protección (considera enfoques) y fiabilidad (considera enfoques, tipo y resiliencia), que aseguran de que no surja un peligro o conduzca a un accidente o que, sí ocurre el daño sea mínimo. Utilizando el árbol de fallas para definirlas. (Sommerville, 2016, p. 351).
- **Elaboración de Diagrama de Casos de uso**: con el artefacto estrategias de negocio se realizará una identificación y descripción del comportamiento del sistema con los actores que interactúan con el mismo e identificará nuevos requisitos funcionales y no funcionales que puedan ser descubiertos en el proceso de descripción de casos de uso y sus interacciones. Cada uno tendrá su especificación de caso de uso y serán asociados a los requisitos del sistema. También incluirá al finalizar casos de uso de funcionamiento esperado, la realización casos de uso inapropiado del sistema que derivaran en requisitos de seguridad.
- **Elaboración de la visión del proyecto:** consiste en la definición de manera clara y concisa el propósito, objetivos, alcance, beneficios esperados, partes interesadas, restricciones y criterios para el éxito del proyecto. Utilizando los artefactos de estrategia de negocio, requisitos de protección y fiabilidad como base.
- **Priorización y clasificación de requisitos:** consiste en evaluar y organizar los requisitos previamente obtenidos en función de su importancia e impacto en el

proyecto. Esto mediante la asignación de prioridades y la clasificación en categorías, funcionales y no funcionales como según los atributos de calidad.

- **Especificación formal:** consiste en elaborar especificaciones expresadas en lenguaje Z, con el objetivo de evitar ambigüedades, aumentar la confiabilidad del sistema, y facilitar la detección de errores al obtener mayor precisión y detalle de lo que debe diseñarse e implementarse.
- **Gestión de requisitos**: Consiste en realizar una sesión donde se verifica la trazabilidad y coherencia de los mismos que deberá ser aprobada por cada miembro del equipo de desarrollo al finalizar cada actividad. Así como la notificación de cambios y registro de los mismos

Artefactos

- **Documento de estrategias de mercado(negocio):** describe los estudios realizados de otros modelos, las ventajas y desventajas de los modelos más utilizados, un análisis de la competencia, las necesidades y los deseos de los usuarios.

- **Requisitos no debe:** describe las especificaciones de protección y determinan los comportamientos que son considerados inaceptables en el sistema

- **Bitácora de peligro:** es el documento central de seguridad ante daños físicos. Proporciona evidencia de cómo se consideraron los peligros identificados durante el desarrollo del software.

- **Documento clasificación de riesgos:** documento donde clasificamos los riesgos bajo los criterios de : probabilidad de riesgo,severidad del riesgo, riesgo estimado y aceptabilidad.

- **Documento árbol de fallas:** documento consiste en la descomposición del peligro en estados que lo conducen a su ocurrencia, hasta descubrir las causas raíces del mismo, que surgirán al no poder seguir descomponiendo.

- **Requisitos de protección**: Documento donde se describe,clasifica y evalúa los peligros identificados.

- **Requisitos de fiabilidad:** documento que describe los requisitos de fiabilidad.

- **Diagrama de caso de uso:** documento con los principales casos de usos diagramados.

- **Visión del proyecto:** documento que proporciona una comprensión clara y concisa de los objetivos, metas y principios rectores del proyecto.

- **Lista de requisitos:** Documento que lista los requisitos identificados en el software (no debe, funcionales, no funcionales).

- **Documento de requerimientos del sistema:** resultado del proceso de ingeniería de requisitos, es la descripción de lo que debería hacer el sistema y de los requisitos no funcionales claves que el sistema debería tener.

- **Clasificación de riesgos:** documento que clasifica los riesgos identificados y documentados en la bitácora de peligro.

8. Meta-Modelo del Proceso de Disciplina de Diseño

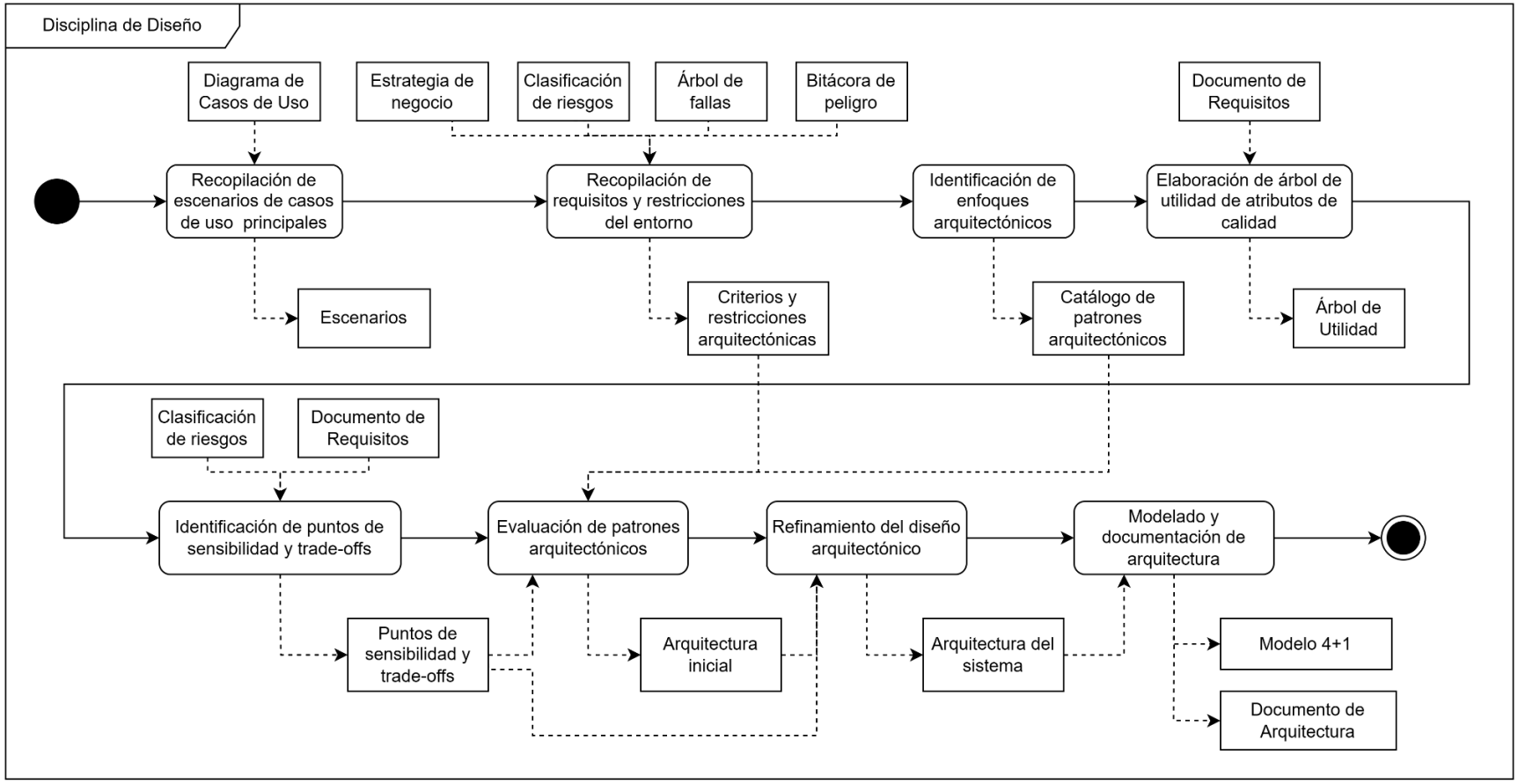


Figura 1.14: Metamodelo del Proceso de Ingeniería o Disciplina de Diseño Arquitectónico
Diagrama de Actividad (UML 2.5.1) Actividades

Las actividades del proceso de diseño arquitectónico están basadas en las fases del proceso ATAM (Architecture Trade-off Analysis Method) presentadas en un reporte técnico del Instituto de Ingeniería de Software (Kazman et al., 2000):

Fases de ATAM	Pasos de ATAM	Actividades en el Metamodelo
Fase 1: Preparación/ Recopilación de información	**Paso 1: Presentar ATAM**	-
	Paso 2: Presentar impulsores de negocio	Recopilación de escenarios de casos de uso principales
		Recopilación de requisitos y restricciones del entorno
	Paso 3: Presentar arquitectura	-
	Paso 4: Identificar enfoques arquitectónicos	Identificación de enfoques arquitectónicos
	Paso 5: Generar Árbol de Utilidad	Elaboración de árbol de utilidad de atributos de calidad
Fase 2: Evaluación de la arquitectura	**Paso 6: Analizar enfoques arquitectónicos** **Paso 7: Tormenta de ideas y análisis de escenarios** **Paso 8: Analizar enfoques arquitectónicos (Repetir paso 6)**	Identificación de puntos de sensibilidad y trade-offs Evaluación de patrones arquitectónicos Refinamiento del diseño arquitectónico
	Paso 9: Presentar resultados	Modelado y documentación de Arquitectura

Tabla 1.12: Pasos de ATAM en la Disciplina de Diseño Arquitectónico

Descripción de Actividades:

- **Paso 1: Presentar ATAM**
  - No se realiza esta actividad de ATAM
- **Paso 2: Presentar impulsores de negocio**
  - Recopilar escenarios de casos de uso principales: consiste en generar escenarios de casos de uso; con el fin de elicitar de manera más precisa objetivos de calidad que serán evaluados próximamente en cada arquitectura.
  - Recopilación de requisitos y restricciones del entorno: utilizando como entradas la estrategia de negocio y el documento de requisitos elaborados en la Disciplina de Requerimientos, esta actividad tiene como objetivo obtener los requerimientos funcionales más importantes, restricciones técnicas, económicas o políticas, contexto del mercado y metas de negocio resultantes del análisis de mercado, y las partes interesadas más importantes. (Kazman et al., 2000)

    También se incluyen los puntos clave de decisiones de diseño arquitectónico: atributos de calidad, enfoques y estilos más adecuados, distribución de los componentes del sistema, entre otros. (Sommerville, 2015, 171)

- **Paso 3: Presentar arquitectura**
- **Paso 4: Identificar enfoques arquitectónicos**
  - Identificación de enfoques arquitectónicos: tiene como objetivo presentar alternativas de arquitectura y comprender los estilos de cada enfoque, pero no analizarlos. (Kazman et al., 2000)

- **Paso 5: Generar Árbol de Utilidad**
  - Elaboración de árbol de utilidad de atributos de calidad: el equipo identifica, prioriza y refina los objetivos de calidad más importantes del sistema, construyendo un árbol de utilidad. (Kazman et al., 2000)

- **Paso 6: Analizar enfoques arquitectónicos**
- **Paso 7: Tormenta de ideas y análisis de escenarios**
- **Paso 8: Analizar enfoques arquitectónicos (Repetir paso 6)**
  - Identificación de puntos de sensibilidad y trade-offs: mediante el análisis de escenarios, la tormenta de ideas junto a especialistas y técnicos, se identificarán riesgos del sistema, puntos de sensibilidad y *trade-offs*. (Kazman et al., 2000)

    Por otro lado, se tomarán decisiones acerca de los potenciales Conflictos Arquitectónicos, principalmente cómo diseñar un sistema que garantice la seguridad ante daños físicos del sistema, mientras mantiene su disponibilidad, puntualidad y otros requisitos no funcionales. (Somerville, 2015, 173)
  - Evaluación de patrones arquitectónicos: consiste en en base a los requisitos funcionales, de calidad, restricciones y metas recopilados en el paso 2.

- Refinamiento del diseño arquitectónico: mejorar el patrón seleccionado como base teniendo en cuenta: aplicación de redundancia, diversidad, enfoques y arquitectura de automonitoreo vista en ingeniería de sistemas de tiempo real.

- **Paso 9: Presentar resultados**
  - Modelado y documentación de Arquitectura**:** la actividad de modelar consiste en generar el Modelo 4+1 de la Arquitectura, propuesto por Krutchen (Krutchen & Rational Software Corporation, 1995). Luego, se elabora el Documento de Arquitectura, definiendo una visión general, el estilo arquitectónico y el Modelo 4+1 elaborado.

Artefactos

- **Diagrama de casos de uso:** documento con los principales casos de usos diagramados.

- **Estrategia de negocio:** elaborado en Disciplina de Requerimientos. Es el resultado de un estudio del mercado actual en el que se encuentran otros modelos, las ventajas y desventajas de los modelos más utilizados, un análisis de la competencia, las necesidades y los deseos de los usuarios.

- **Clasificación de riesgos:** documento que clasifica los riesgos identificados y documentados en la bitácora de peligro.

- **Documento árbol de fallas:** documento consiste en la descomposición del peligro en estados que lo conducen a su ocurrencia, hasta descubrir las causas raíces del mismo, que surgirán al no poder seguir descomponiendo.

- **Bitácora de peligro:** es el documento central de seguridad ante daños físicos. Proporciona evidencia de cómo se consideraron los peligros identificados durante el desarrollo del software.

- **Documento de requisitos:** elaborado en Disciplina de Requerimientos. Es el resultado de la especificación de requisitos e incluye las declaraciones oficiales y detalladas de lo que debe realizar el sistema.

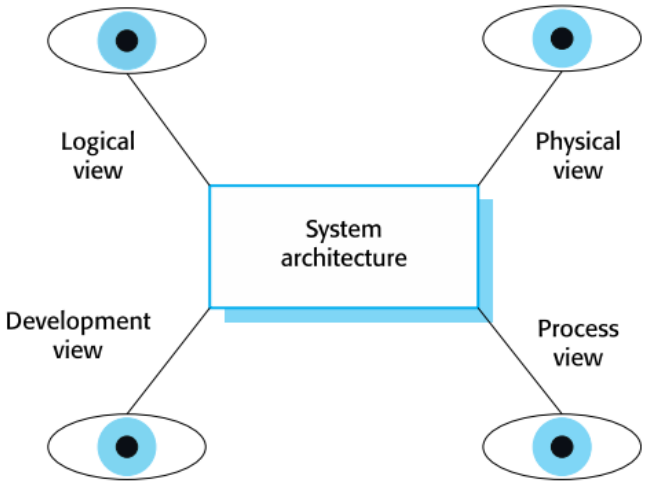
- **Escenarios:** son declaraciones breves que describen una interacción entre un *stakeholder* (interesado del sistema) con el sistema. Un usuario podría describir su uso del sistema para realizar una tarea, mientras que un desarrollador o un técnico de mantenimiento podría describir cómo incorpora un cambio o una actualización al sistema. Se utilizan tres tipos de escenarios en ATAM:
  - **Escenarios de casos de uso:** describen una interacción del usuario con el sistema completo y en ejecución.

- **Criterios y restricciones arquitectónicas:** documento que surge del análisis del estudio de mercado, bitácora de peligro, clasificación de riesgos. Describe las limitaciones y criterios a tomar en cuenta para la selección de patrones arquitectónicos.

- **Catálogo de patrones arquitectónicos:** surge de la identificación de patrones y enfoques arquitectónicos, son los principales enfoques a tener en cuenta para la evaluación y selección.

- **Árbol de utilidad (Priorización de atributos de calidad):** es un esquema en forma de árbol descendente que facilita la priorización de objetivos de calidad más importantes para el sistema en cuestión. El nodo raíz es un atributo de calidad "paraguas" que se subdivide en atributos más específicos, que son las hojas del árbol. Estas hojas serán priorizadas de manera relativa, es decir, en comparación con las demás, utilizando una escala de puntuación o clasificaciones como "Alto", "Medio" y "Bajo".

- **Modelo 4+1:** es el modelo propuesto por Krutchen, que incluye 4 vistas: lógica, de procesos, de desarrollo y física, en adición a una vista general llamada conceptual, para poder ilustrar de manera completa todos los aspectos del sistema. (Krutchen & Rational Software Corporation, 1995).



- Figura 1.15: Vistas Arquitectónicas (Sommerville, 2016, 174)

- **Documento de Arquitectura:** es el resultado final del Diseño Arquitectónico, resumiendo las decisiones tomadas y la arquitectura elegida, que será fundamental para el mantenimiento y escalabilidad del sistema a largo plazo.

9. Meta-Modelo del Proceso de Disciplina de Verificación

9.1. Meta-Modelo de Pruebas de Unidad

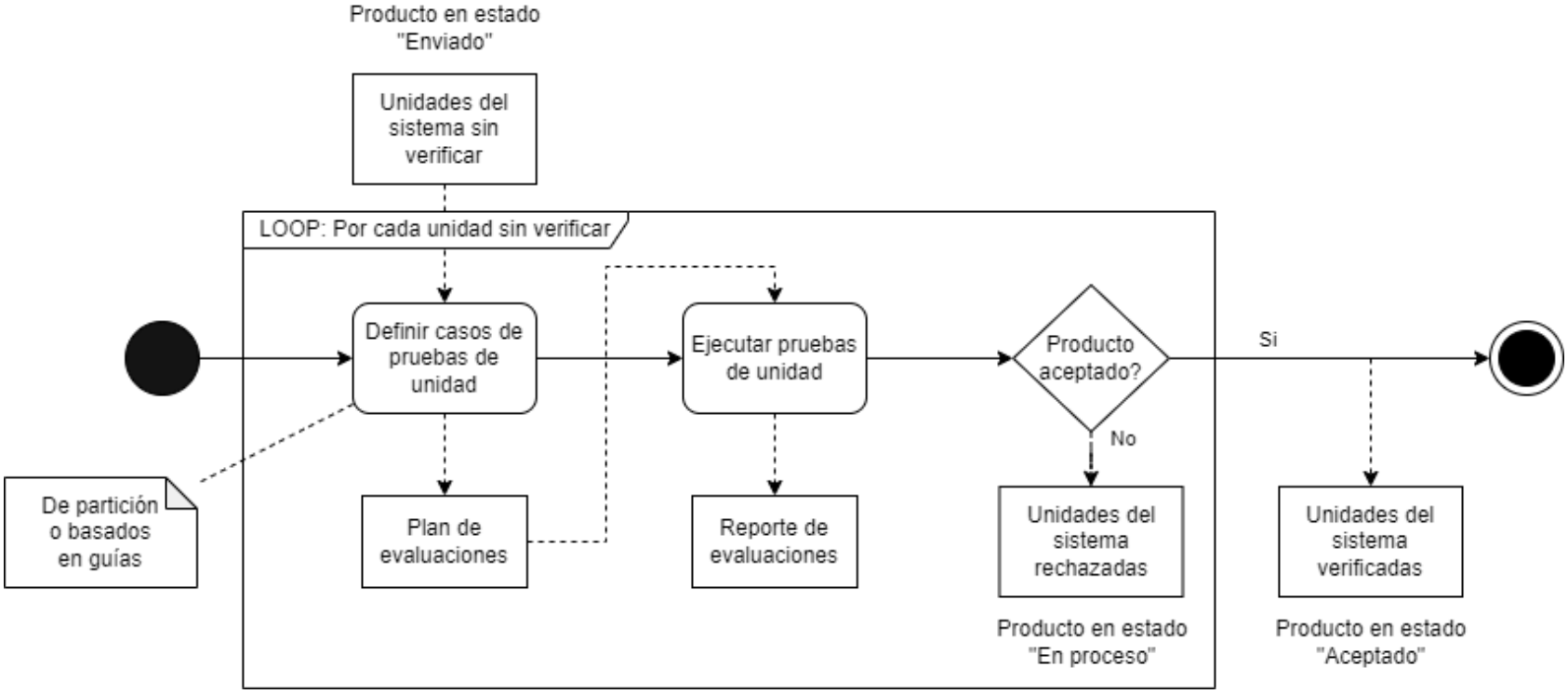


Figura 1.16: Meta-Modelo de Pruebas de Unidad
(Diagrama de Actividad UML 2.5.1)

Actividades

- **Definir casos de prueba de unidad:** En esta actividad se tomará una unidad del sistema sin verificar y se construirán los diferentes casos de pruebas para esta unidad, dichos casos de pruebas serán documentados en el plan de evaluaciones.
  En esta actividad se generan argumentos estructurados de seguridad, que la prueba deberá superar para ser aprobada, cumpliendo con las contradicciones al estado inseguro. Se generará caso de seguridad donde se registrará la evidencia que demuestre que el sistema es seguro.

- **Ejecutar pruebas de unidad:** En esta actividad se ejecutarán los casos de pruebas de unidades que están documentados en el plan de evaluaciones. Los resultados de las pruebas serán documentados en el reporte de evaluaciones. Esta deberá ser registrada en el casos de seguridad construido.
  Si la unidad del sistema pasa las pruebas, se aceptará y pasará a ser una unidad del sistema verificada, la cual se podrá utilizar para el siguiente desarrollo del sistema. Si la unidad del sistema no pasa las pruebas, se rechazará y pasará a ser una unidad del sistema rechazada, la cual se deberá seguir desarrollando para poder verificarla en otro momento. Se registrará el reporte de evaluación de la prueba de ser aceptada en caso de estudio construido. Así como se incluirá información de métricas de fiabilidad resultantes de la prueba en el reporte de evaluación.

- **Planificación y Gestión de Pruebas:** esta actividad consiste en la asignación del responsable/s de realizar la prueba de unidad como las herramientas y entornos requeridos y a ser utilizados para llevar a cabo las pruebas. Así como el registro y seguimiento de reportes de pruebas para alcanzar la aceptación de la misma. Esta iniciará en la actividad Definir casos de prueba de unidad y seguirá con el seguimiento y registro de últimos reportes obtenidos en la actividad ejecutar pruebas de unidad. Esta será ejecutada durante todo el proceso de la prueba.

Artefactos

- **Plan de evaluaciones:** es un documento en el cual están definidos los métodos, criterios, ambiente y casos de prueba de las evaluaciones.

- **Reporte de evaluaciones:** es un documento en el cual están documentados los resultados de las evaluaciones, junto con las causas de rechazo de los productos los cuales se utilizaran para posteriores re-desarrollos de los productos. Incluira información de métricas de fiabilidad alcanzadas, contradicciones a estado inseguro aprobadas e información de test de penetración

- **Unidades del sistema**: son aquellos métodos y funciones individuales de las clases, componentes o módulos que utiliza el software.

- **Enviada**: se encuentra en el proceso de evaluación del submodelo *"Quality Assurance"* (Control de Calidad).
- **En proceso**: se encuentra bajo control del desarrollador. Una unidad del sistema volverá a este estado si es rechazada.
- **Aceptada**: ha sido aceptada por el control de calidad y será publicada.

9.2. Meta-Modelo de Pruebas de Componentes

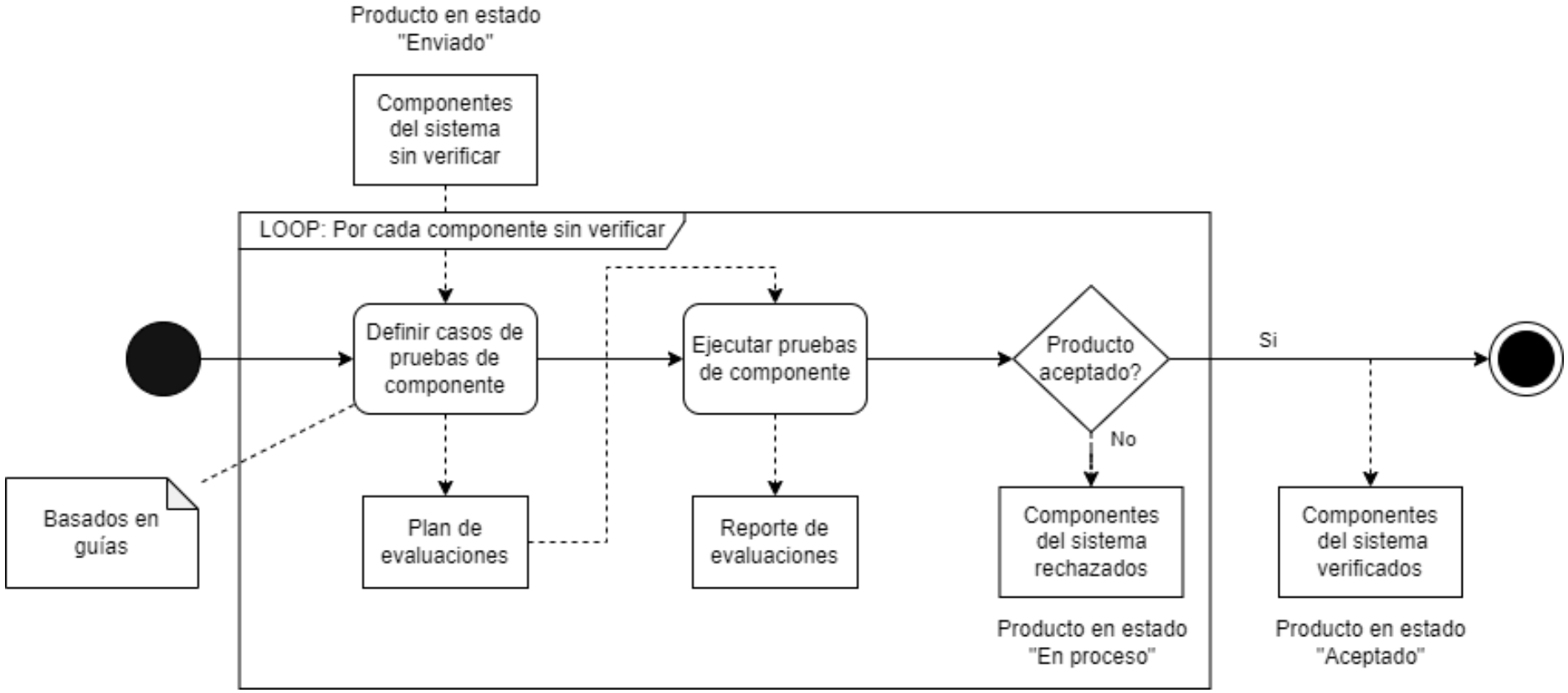


Figura 1.17: Meta-Modelo de Pruebas de Componentes
Diagrama de Actividad (UML 2.5.1)

Actividades

- **Definir casos de prueba de componente:** en esta actividad se tomará un componente del sistema sin verificar y se construirán los diferentes casos de pruebas para este componente, dichos casos de pruebas serán documentados en el plan de evaluaciones. En esta actividad se generan argumentos estructurados de seguridad, que la prueba deberá superar para ser aprobada, cumpliendo con las contradicciones al estado inseguro.

- **Ejecutar pruebas de componente:** en esta actividad se ejecutarán los casos de pruebas de componentes que están documentados en el plan de evaluaciones. Los resultados de las pruebas serán documentados en el reporte de evaluaciones.
  Esta deberá ser registrada en el casos de seguridad construido en la prueba de unidad.
  Si el componente del sistema pasa las pruebas, se aceptará y pasará a ser un componente del sistema verificado, el cual se podrá utilizar para el siguiente desarrollo del sistema. Si el componente del sistema no pasa las pruebas, se rechazará y pasará a ser un componente del sistema rechazado, el cual se deberá seguir desarrollando para poder verificarlo en otro momento. Así como se incluirá información de métricas de fiabilidad resultantes de la prueba en el reporte de evaluación.

- **Planificación y Gestión de Pruebas:** esta actividad consiste en la asignación del responsable/s de realizar la prueba del componente como las herramientas y entornos requeridos y a ser utilizados para llevar a cabo las pruebas. Así como el registro y seguimiento de reportes de pruebas para alcanzar la aceptación de la misma. Esta iniciará en la actividad Definir casos de prueba de componente y seguirá con el seguimiento y registro de últimos reportes obtenidos en la actividad ejecutar pruebas de componente. Esta será ejecutada durante todo el proceso de la prueba.

Artefactos

- **Plan de evaluaciones:** es un documento en el cual están definidos los métodos, criterios, ambiente y casos de prueba de las evaluaciones.

- **Reporte de evaluaciones:** es un documento en el cual están documentados los resultados de las evaluaciones, junto con las causas de rechazo de los productos los cuales se utilizaran para posteriores re-desarrollos de los productos.

- **Componentes del sistema:** son los módulos o partes que componen a un sistema y realizan un conjunto de tareas específicas.

a. **Enviado**: se encuentra en el proceso de evaluación del submodelo *"Quality Assurance"* (Control de Calidad).
b. **En proceso**: se encuentra bajo control del desarrollador. Un componente del sistema volverá a este estado si es rechazado.
c. **Aceptado**: ha sido aceptado por el control de calidad y será publicado.

9.3. Meta-Modelo de Pruebas de Sistema

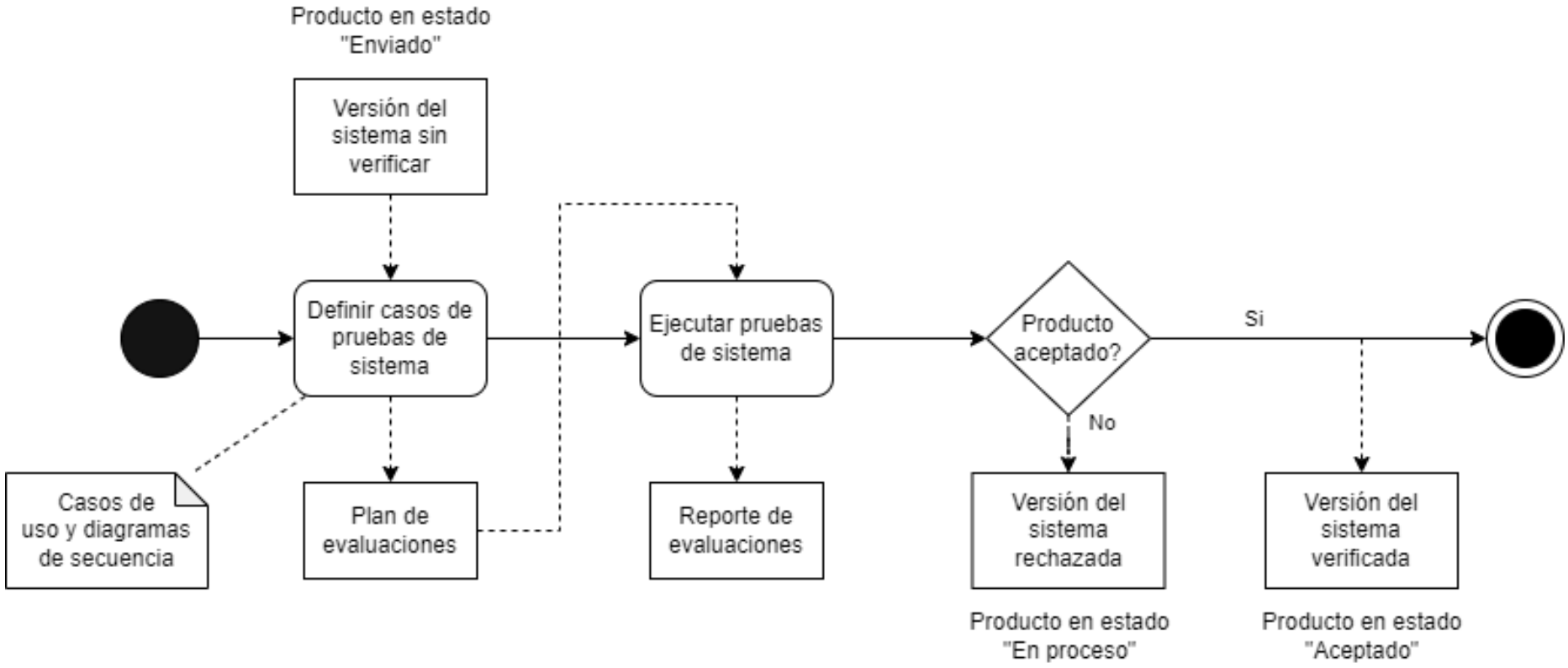


Figura 1.18: Meta-Modelo de Pruebas de Sistema
Diagrama de Actividad (UML 2.5.1)

Actividades

- **Definir casos de prueba de sistema:** En esta actividad se tomará a la versión del sistema sin verificar y se construirán los diferentes casos de pruebas para el sistema, dichos casos de pruebas serán documentados en el plan de evaluaciones.
  En esta actividad se generan argumentos estructurados de seguridad, que la prueba deberá superar para ser aprobada, cumpliendo con las contradicciones al estado inseguro.
  También se diseñarán casos de prueba para la seguridad contra la intrusión accidental o deliberada (pruebas de penetración).

- **Ejecutar pruebas de sistema:** En esta actividad se ejecutarán los casos de pruebas de la versión del sistema que están documentados en el plan de evaluaciones. Los resultados de las pruebas serán documentados en el reporte de evaluaciones. Esta deberá ser registrada en el casos de seguridad construido en la prueba de unidad.
  Si la versión del sistema pasa las pruebas, se aceptará y pasará a ser una versión del sistema verificada, el cual se podrá utilizar para el siguiente desarrollo del sistema. Si la versión del sistema no pasa las pruebas, se rechazará y pasará a ser una versión del sistema rechazada, el cual se deberá seguir desarrollando para poder verificarlo en otro momento. Así como se incluirá información de métricas de fiabilidad resultantes de la prueba en el reporte de evaluación.
  Se ejecutarán pruebas de penetración cuyo resultado estará incluido en el reporte.

- **Planificación y Gestión de Pruebas:** esta actividad consiste en la asignación del responsable/s de realizar la prueba del sistema como las herramientas y entornos requeridos y a ser utilizados para llevar a cabo las pruebas. Así como el registro y seguimiento de reportes de pruebas para alcanzar la aceptación de la misma. Esta iniciará en la actividad Definir casos de prueba de sistema y seguirá con el seguimiento y registro de últimos reportes obtenidos en la actividad ejecutar pruebas del sistema. Esta será ejecutada durante todo el proceso de la prueba.

Artefactos

- **Plan de evaluaciones:** Es un documento en el cual está definido los métodos, criterios, ambiente y casos de prueba de las evaluaciones

- **Reporte de evaluaciones:** Es un documento en el cual están documentados los resultados de las evaluaciones, junto con las causas de rechazo de los productos los cuales se utilizaran para posteriores re-desarrollos de los productos

- **Versión del sistema**: es una instancia específica del software que ha sido modificada, actualizada o desarrollada para cumplir con ciertos requisitos o mejoras.

- **Enviada**: se encuentra en el proceso de evaluación del submodelo *"Quality Assurance"* (Control de Calidad).
- **En proceso**: se encuentra bajo control del desarrollador. Una versión del sistema volverá a este estado si es rechazada.
- **Aceptada**: ha sido aceptada por el control de calidad y será publicada.

Artefactos

- **Plan de evaluaciones:** es un documento en el cual están definidos los métodos, criterios, ambiente y casos de prueba de las evaluaciones.
- **Reporte de evaluaciones:** es un documento en el cual están documentados los resultados de las evaluaciones, junto con las causas de rechazo de los productos los cuales se utilizaran para posteriores re-desarrollos de los productos.
- **Unidades del sistema**: son aquellos métodos y funciones individuales de las clases, componentes o módulos que utiliza el software.
    - **Enviada**: se encuentra en el proceso de evaluación del submodelo *"Quality Assurance"* (Control de Calidad).
    - **En proceso**: se encuentra bajo control del desarrollador. Una unidad del sistema volverá a este estado si es rechazada.
    - **Aceptada**: ha sido aceptada por el control de calidad y será publicada.:

10. Consideraciones

Se decidió incluir la fase de diseño de módulos en la fase de diseño de sistema. En lugar de hacerlas por separado, hemos integrado las actividades en diseño de sistema.

Además, se ha acotado el sistema a únicamente el funcionamiento automático de la bomba de insulina, excluyendo la opción de configuración manual, la cual a pesar de proveer funcionalidades nuevas, aporta también más riesgos al paciente que utiliza el dispositivo médico. Con el sistema Innopump, la medición y administración de insulina será siempre automatizada. Por lo tanto no se ejecutará o aplicaran las siguientes subactividades:

- Disciplina de requerimientos:
  - Casos de uso inapropiado de la actividad de Elaboración de casos de uso.
  - Evaluación de requisitos de autenticación y autorización en la subactividad casos de uso inapropiado.
- Disciplina de diseño arquitectónico:
  - Enfoque de sistema ante errores humanos.
- Disciplina de pruebas de sistema:
  - Pruebas de seguridad (intrusión deliberada o accidental) de la actividad Definir Caso de Prueba de Sistema.

Parte II

11. Ejecución del Modelo de Proceso de Desarrollo

Disciplina de Requisitos

11.1. Análisis de Mercado

En el mercado se cuenta actualmente con dos compañías principales en la venta de bombas de insulina (BI): ROCHE y Medtronic Company, con sus principales modelos, Spirit Combo y Paradigma 715 y 754 (Veo) respectivamente.

A modo de resumen del análisis de mercado realizado previamente, se presentarán sus características en común, puntos a favor y en contra de cada uno, y requisitos derivados de estos para nuestro sistema.

Requisitos identificados previamente (TP 2)

Administración de insulina

1. El sistema debe estar disponible para suministrar insulina cuando sea necesario (*availability*).
2. El sistema debe medir el nivel de azúcar en sangre y suministrar insulina cada 10 minutos, si fuese necesario.
3. El sistema entregará la cantidad correcta de insulina para contrarrestar el nivel actual de azúcar en sangre (*reliability*, *safety*).
4. La dosis de insulina a ser suministrada debe ser calculada mediante la medición del nivel actual de azúcar en sangre, comparando éste a un nivel de medición previo.
5. La cantidad de insulina a ser suministrada debe ser calculada según la lectura actual de azúcar, según lo mida el sensor:
    - 5.1. Si la lectura está por debajo del mínimo, no se debe suministrar insulina.
    - 5.2. Si la lectura está entre los parámetros adecuados, la insulina sólo será suministrada si el nivel de azúcar está creciendo y la tasa de incremento del azúcar está aumentando.
    - 5.3. Si la lectura está por sobre el nivel recomendado, la insulina se suministra a menos que el nivel de azúcar en sangre esté decreciendo y la tasa de decremento del mismo esté incrementando.
    - 5.4. La cantidad de insulina realmente suministrada puede ser diferente a la dosis calculada debido a las distintas restricciones de seguridad (*safety*) que se incluyen en el sistema. Hay un límite de la dosis máxima a ser suministrada en cada inyección, y un límite acumulativo por día.

Manejo de Errores

6. Las condiciones de error que el sistema debe detectar e indicar son:
    - 6.1. Batería baja: el voltaje de la batería está por debajo de 0.5 v.
    - 6.2. Falla de sensor: el auto testeo del sensor de azúcar ha resultado erróneo.
    - 6.3. Falla de la bomba: el auto testeo de la bomba ha resultado erróneo.

6.4. Falla del suministro: la aguja podría estar bloqueada o insertada incorrectamente.
6.5. Aguja removida: el usuario ha removido la aguja.
6.6. Reservorio de insulina removido: el usuario ha removido el reservorio de insulina.
6.7. Bajo nivel de insulina: el nivel de insulina está bajo (indicando que el reservorio debe cambiarse).

7. El sistema deberá tener un tamaño y peso que sea ergonómico para todo tipo de personas.
8. El sistema deberá permitir la carga o cambio de baterías al usuario.
9. El sistema deberá consumir la menor cantidad de energía posible logrando una autonomía de al menos 1 semana efectiva.

Análisis de la línea de productos de software

Producto	Ventajas	Desventajas	Requisitos derivados
Spirit Combo (ROCHE)	Sistema enfocado en la detección de hipoglucemia y la suspensión automática de insulina. Ofrece opciones de conectividad, permite a los pacientes transferir datos de glucosa e insulina a sistemas de gestión de diabetes y a profesionales de la salud para su revisión y análisis. Sistema de infusión confiable y preciso. Interfaz intuitiva, fácil de navegar. Baterías recargables. Tamaño del cartucho 3.5 ml (350 U-100), equivalentes de 7 a 10 días de uso.	Manuales de Usuario no disponibles, especificaciones no claras. El costo asociado es de 5000 a 7000 dólares para su adquisición y un gasto anual promedio de 2500 dólares en insumos (sets de infusión, baterías , reservorios, etc). Uso de conexión bluetooth a muy corto alcance.	El sistema debe contar con documentación incluyendo manual de usuario. El sistema debe contar con una interfaz intuitiva para todo tipo de usuarios. El sistema debe ejecutar un programa de auto testeo cada 30 segundos. Al inicio de cada período de 24 horas (indicado por clock = 00:00:00), el sistema debe reiniciar la dosis acumulativa de insulina suministrada a 0. El sistema debe permitir al usuario reemplazar el reservorio de insulina por uno nuevo. El sistema no debe permitir al usuario reemplazar el reservorio de insulina durante una administración.

Producto	Ventajas	Desventajas	Requisitos derivados
Paradigma 715 y 754 (Veo) (Medtronic)	Sistema enfocado en la detección y prevención de hipoglucemia. Mantiene historiales y registros. Sistema de infusión confiable y preciso. Facilidad de uso, interfaces simples e intuitivas.	El dispositivo ya no es adaptable a las últimas versiones de software desarrolladas por Medtronic (Versión de sistema menor a 2.6). Tamaño de cartucho 1.8 ml (180 U-100), aproximadamente de 4 a 6 días. El costo asociado es de 5000 a 7000 dólares para su adquisición y un gasto anual promedio de 2500 dólares en insumos (sets de infusión, baterías , reservorios, etc)	El sistema debe mantener historiales y registros de mediciones y administraciones de insulina realizadas. El sistema debe admitir un reservorio de 315 ml de insulina.

Tabla 2.1: Ventajas, desventajas y requisitos derivados del análisis de la línea de productos similares en el mercado

11.2. Identificación de requerimientos no debe

Algunos requerimientos “No Debe” identificados inicialmente durante el análisis de mercado son:

No considerados(ver):

- El sistema **no debe** permitir la configuración de dosis/valores excepcionales en parámetros de dosificación.
- El sistema **no debe** permitir el acceso de usuarios no autorizados a la configuración de parámetros de dosificación automatizada.

Considerados

- El sistema **no debe** permitir al usuario reemplazar el reservorio de insulina durante una administración.
- El sistema **no debe** interrumpir el sistema de alarmas ante un fallo en el sistema.
- El sistema **no debe** permitir la administración de una sobredosis o subdosis de insulina
- El sistema **no debe** funcionar con batería por debajo de un umbral aceptable para el correcto funcionamiento.
- El sistema **no debe** pasar por alto la ocurrencia de un fallo en el sistema.

11.3. Especificación de Protección

11.3.1. Identificación de Peligros

Peligros identificados

Algunos de los siguientes peligros identificados en el uso de Innopump son extraídos de la sección *Hazard identification* de Sommerville (Sommerville, 2016, p. 346).

Bitácora de peligro

N°	Peligro	Tipo
1	Cálculo de sobredosis de insulina	Falla del sistema
2	Cálculo de subdosis de insulina	Falla del sistema
3	Fallo del hardware de sensor	Falla del sistema
4	Fallo de energía por falta de batería	Eléctrico
5	Interferencia eléctrica con otros dispositivos médicos	Eléctrico
6	Mal contacto del sensor	Físico
7	Ruptura de partes del dispositivo en el cuerpo del paciente	Físico

N°	Peligro	Tipo
8	Infección causada por el uso del dispositivo	Biológico
9	Reacción alérgica a los materiales del dispositivo o la insulina dosificada	Biológico
10	Pérdida de conexión entre sensores y actuadores	Eléctrico
11	Descalibración de sensores que produce mediciones incorrectas	Falla del sistema
12	Fallos en la estabilidad del dispositivo debido a temperaturas extremas	Físico

Tabla 2.2: Bitácora de peligros

11.3.2. Evaluación de Riesgos

Para clasificar la severidad de los riesgos identificados, se tendrán en cuenta las posibles consecuencias del accidente causado:

N°	Peligro	Posibles consecuencias
1	Cálculo de sobredosis de insulina	Consecuencias severas de salud (coma e incluso muerte).
2	Cálculo de subdosis de insulina	A corto plazo, niveles altos de azúcar. A largo plazo, problemas en los riñones, el corazón o la vista del paciente.
3	Fallo del hardware de sensor	Administraciones muy variables de insulina, que pueden provocar niveles inestables, altos o bajos de azúcar en sangre. La falla puede no ser notada por el paciente.
4	Fallo de energía por falta de batería	No disponibilidad del dispositivo. La falla será notada por el paciente.
5	Interferencia eléctrica con otros dispositivos médicos	Mediciones y administraciones variables de insulina, que pueden provocar niveles inestables, altos o bajos de azúcar en sangre. La falla puede no ser notada por el paciente.
6	Mal contacto del sensor	Mediciones y administraciones variables de insulina, que pueden provocar niveles inestables, altos o bajos de azúcar en sangre. La falla puede no ser notada por el paciente.
7	Ruptura de partes del dispositivo en el cuerpo del paciente	Infección, obstrucción de vasos sanguíneos, reacciones inmunológicas graves a las partes del dispositivo.
8	Infección causada por el uso del dispositivo	Reacciones inmunológicas al uso del dispositivo.
9	Reacción alérgica a los materiales del dispositivo o la insulina dosificada	Reacciones inmunológicas leves al uso del dispositivo.

N°	Peligro	Posibles consecuencias
10	Pérdida de conexión entre sensores y actuadores	No disponibilidad del dispositivo. La falla será notada por el paciente.
11	Descalibración de sensores que produce mediciones incorrectas	Administraciones muy variables de insulina, que pueden provocar niveles inestables, altos o bajos de azúcar en sangre. La falla puede no ser notada por el paciente.
12	Fallos en la estabilidad del dispositivo debido a temperaturas extremas	Administraciones muy variables de insulina, que pueden provocar niveles inestables, altos o bajos de azúcar en sangre. La falla puede no ser notada por el paciente.

Tabla 2.3: Estimación de la severidad de las consecuencias de los peligros identificados

Clasificación de Riesgos

N°	Peligro	Probabilidad del peligro	Severidad del accidente	Riesgo estimado	Aceptabilidad
1	Cálculo de sobredosis de insulina	Media	Alta	Alto	Intolerable
2	Cálculo de subdosis de insulina	Media	Media	Medio	ALARP[1]
3	Fallo del hardware de sensor	Media	Media	Medio	ALARP
4	Fallo de energía por falta de batería	Alta	Baja	Media	ALARP
5	Interferencia eléctrica con otros dispositivos médicos	Baja	Media	Media	ALARP
6	Mal contacto del sensor	Alta	Alta	Alta	Intolerable
7	Ruptura de partes del dispositivo en el cuerpo del paciente	Baja	Alta	Media	ALARP
8	Infección causada por el uso del dispositivo	Media	Media	Media	ALARP
9	Reacción alérgica a los materiales del dispositivo o la insulina dosificada	Baja	Baja	Baja	Tolerable
10	Pérdida de conexión entre sensores y actuadores	Baja	Baja	Baja	Tolerable
11	Descalibración de sensores que produce mediciones incorrectas	Baja	Media	Media	ALARP

[1] ALARP: *As low as reasonably practical*: Riesgos tolerados únicamente si la reducción de los mismos es impráctica o excesivamente costosa.

N°	Peligro	Probabilidad del peligro	Severidad del accidente	Riesgo estimado	Aceptabilidad
12	Fallos en la estabilidad del dispositivo debido a temperaturas extremas	Baja	Media	Media	ALARP

Tabla 2.4: Clasificación de riesgos identificados según probabilidad y severidad

11.3.3. Análisis de Riesgos

Árbol de fallas

Riesgo 1: Cálculo de sobredosis de insulina
Riesgo 2: Cálculo de subdosis de insulina

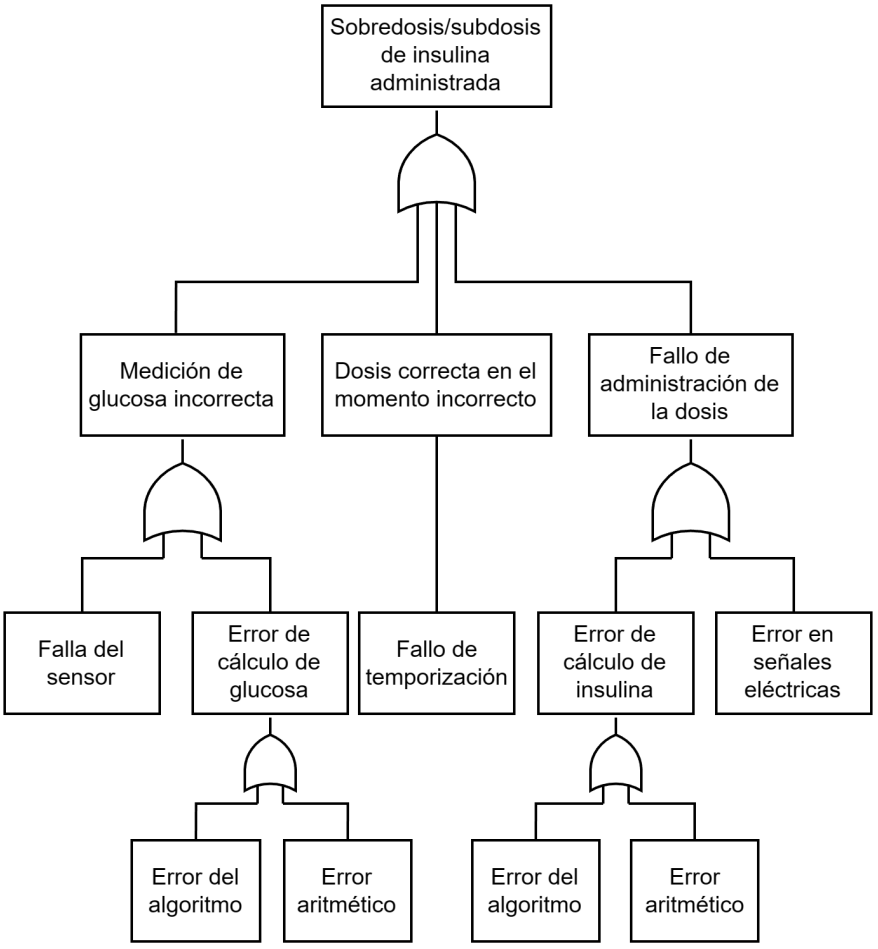


Figura 2.1: Árbol de fallas para los riesgos 1 y 2: Cálculo de sobredosis/subdosis de insulina

Riesgo 3: Fallo de hardware del sensor

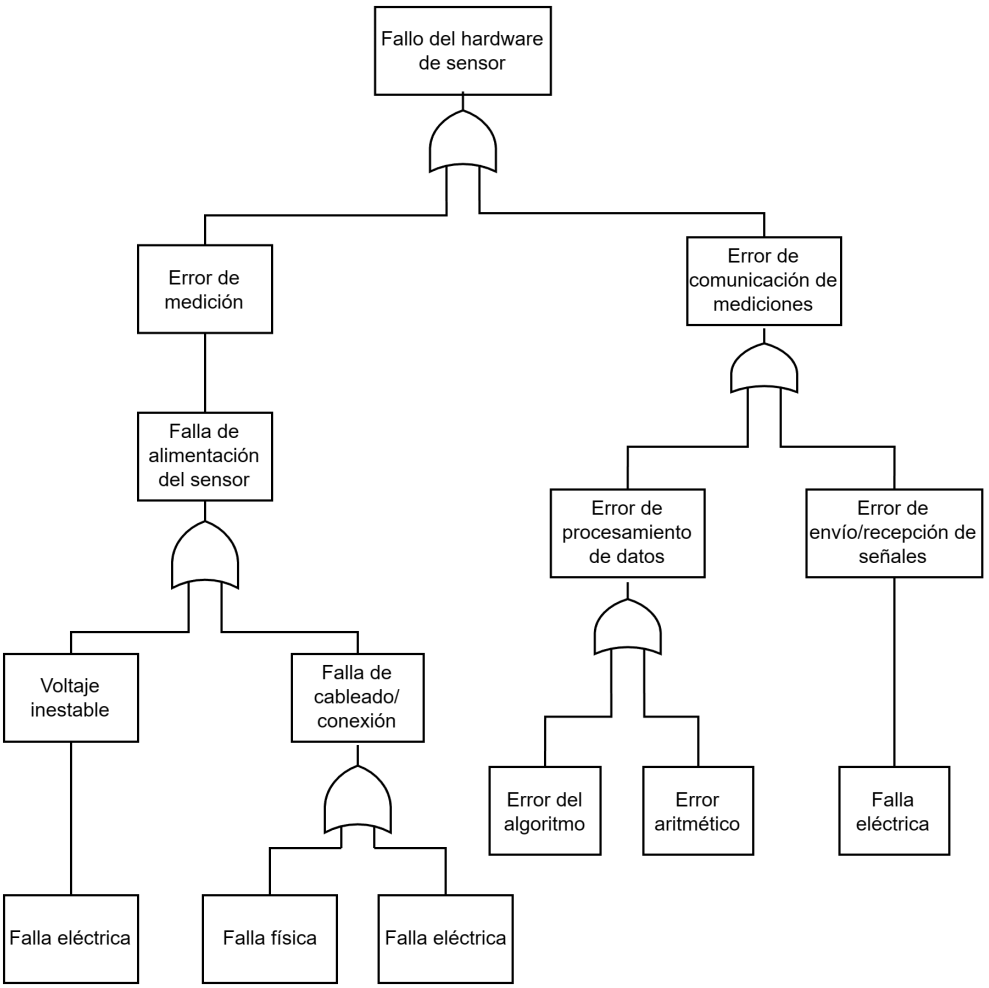


Figura 2.2: Árbol de fallas para el riesgo 3: Fallo de hardware del sensor

Riesgo 10: Pérdida de conexión entre sensores y actuadores

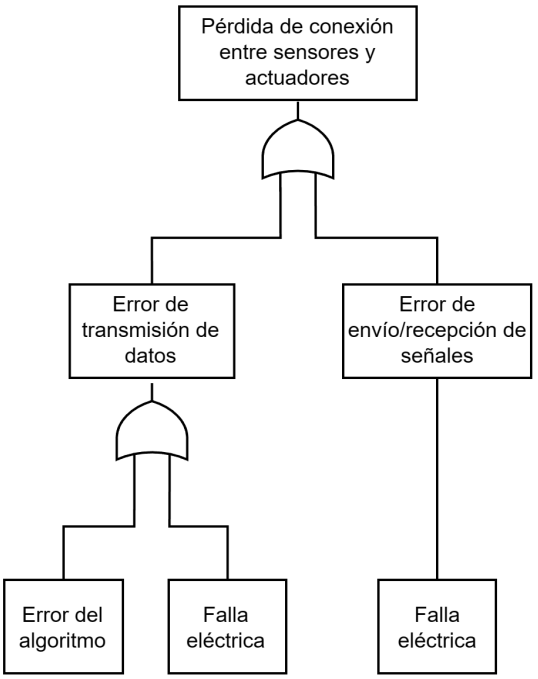


Figura 2.3: Árbol de fallas para el riesgo 10: Pérdida de conexión entre sensores y actuadores

Riesgo 6: Mal contacto del sensor

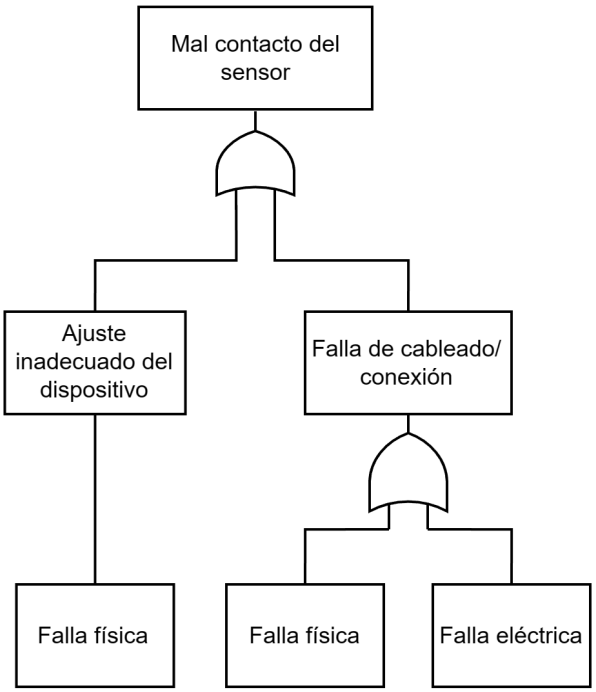


Figura 2.4: Árbol de fallas para el riesgo 6: Mal contacto del sensor

11.3.4. Reducción de riesgos

Requisitos derivados de la especificación de protección

N°	Peligro	Requisitos derivados
1	Cálculo de sobredosis de insulina	El sistema debe calcular y administrar la dosis correcta de insulina de manera fiable y precisa, presentando una Tasa de ocurrencia de fallos (ROCOF) menor a 0,0001. El sistema debe detectar errores en el envío y la recepción de señales eléctricas y detener el funcionamiento.
2	Cálculo de subdosis de insulina	El sistema debe alertar al usuario la detección de errores en el funcionamiento. El sistema no debe permitir el uso del dispositivo tras una repetición de alertas de error.
3	Fallo del hardware de sensor	El sistema debe detectar una falla de hardware del sensor. El sistema debe detectar voltajes inestables y fallos de conexión de cables. El sistema no debe permitir el uso del dispositivo hasta estabilizar la alimentación de energía.
4	Fallo de energía por falta de batería	El sistema debe detectar niveles de batería críticamente bajos y alertar al usuario. El sistema no debe permitir el uso prolongado del dispositivo con niveles críticos de batería que puedan afectar su fiabilidad.
5	Interferencia eléctrica con otros dispositivos médicos	El sistema debe detectar interferencias provenientes de otros dispositivos. El sistema no debe permitir el uso tras detectar interferencias.
6	Mal contacto del sensor	El sistema debe tener un tamaño y peso ergonómicos y adaptables a la diversidad de usuarios.
7	Ruptura de partes del dispositivo en el cuerpo del paciente	El sistema debe detectar y alertar si la aguja está bloqueada o insertada incorrectamente. El sistema no debe permitir el uso tras detectar una ruptura o bloqueo del dispositivo o la aguja.
8	Infección causada por el uso del dispositivo	El sistema debe contar con documentación referida al modo de uso.

N°	Peligro	Requisitos derivados
9	Reacción alérgica a los materiales del dispositivo o la insulina dosificada	El sistema debe contar con documentación incluyendo contraindicaciones y reacciones adversas.
10	Pérdida de conexión entre sensores y actuadores	El sistema debe detectar una falla de comunicación entre sensores y actuadores. El sistema no debe permitir el uso prolongado del dispositivo ante la pérdida de conexión entre sensores y actuadores.
11	Descalibración de sensores que produce mediciones incorrectas	El sistema debe detectar mediciones de glucosa excepcionales.
12	Fallos en la estabilidad del dispositivo debido a temperaturas extremas	El sistema debe detectar temperaturas extremas que afecten el rendimiento del dispositivo. El sistema no debe permitir el uso prolongado del dispositivo bajo condiciones extremas de temperatura.

Tabla 2.5: Requisitos derivados de limitación de riesgos

Requisitos No Debe

En base a la especificación de protección, se reformulan los requisitos “no debe”:

- El sistema **no debe** permitir el uso del dispositivo tras una repetición de alertas de error.
- El sistema **no debe** permitir el uso del dispositivo ante una falla de voltaje hasta estabilizar la alimentación de energía.
- El sistema **no debe** permitir el uso prolongado del dispositivo con niveles críticos de batería que puedan afectar su fiabilidad.
- El sistema **no debe** permitir el uso tras detectar interferencias.
- El sistema **no debe** permitir el uso tras detectar una ruptura o bloqueo del dispositivo o la aguja.
- El sistema **no debe** permitir el uso prolongado del dispositivo ante la pérdida de conexión entre sensores y actuadores.
- El sistema **no debe** permitir el uso prolongado del dispositivo bajo condiciones extremas de temperatura.

11.4. Elaboración de Diagrama de Casos de Uso

Casos de Uso Principales

1. Suministrar dosis de insulina automáticamente
2. Ejecutar algoritmo de auto testeo
3. Remover aguja
4. Remover reservorio de insulina
5. Reiniciar la cantidad de insulina suministrada en el día

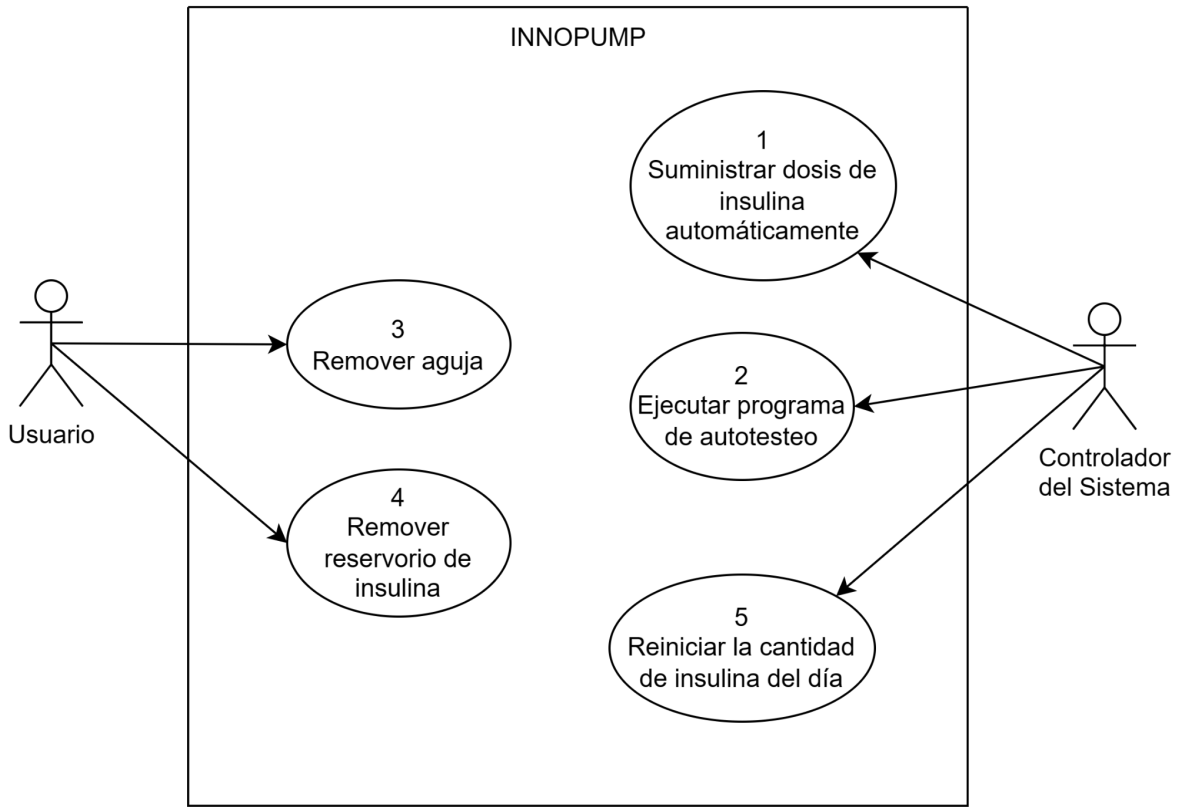


Figura 2.5: Modelo de Casos de Uso para requisitos principales del sistema
Diagrama de Casos de Uso (UML 2.5.1)

11.5. Visión del Proyecto

La visión del proyecto para el desarrollo del sistema de administración de insulina controlado por software es crear una solución alternativa a la forma en que las personas gestionan la diabetes. Este sistema tiene como objetivo principal proporcionar a los usuarios una herramienta segura, confiable y precisa para el control de sus niveles de azúcar en sangre y la administración de insulina.

Para lograr este propósito, el sistema llevará a cabo varias mediciones y evaluará la tasa de cambio del azúcar en sangre en tiempo real. Utilizando esta información, calculará de manera precisa la cantidad incremental de insulina que debe ser inyectada y la administrará de manera eficiente a través de la bomba de insulina.

Uno de los pilares fundamentales de este proyecto es garantizar la seguridad del sistema ante posibles daños a los pacientes. Los usuarios deben tener la confianza de que el sistema funcionará siempre de manera confiable y entregará la cantidad correcta de insulina de manera constante, sin poner en riesgo la salud de los usuarios.

Además de proporcionar una administración precisa de la insulina, el sistema también mantendrá un registro detallado de la periodicidad con la cual se aplican las dosis. Esto no solo ayudará a los usuarios a llevar un control efectivo de su tratamiento, sino que también brindará información valiosa a los profesionales de la salud para realizar un seguimiento y ajuste personalizado de la terapia.

Nuestro enfoque en el desarrollo de este sistema se basa en el compromiso de mejorar la calidad de vida de las personas que dependen de la insulina para controlar su diabetes y en ofrecer una solución que cumpla con los más altos estándares de seguridad y confiabilidad.

Planteamiento del problema

El problema de	La administración manual e imprecisa de insulina que no puede ajustarse a las necesidades fluctuantes del paciente en tiempo real
Afecta a	Personas que sufren de diabetes que requieren monitorización constante y administración precisa de insulina.
El impacto es	Fluctuaciones peligrosas en los niveles de glucosa en sangre, que pueden conducir a complicaciones a corto y largo plazo, incluyendo mal funcionamiento cerebral, inconsciencia, daño ocular, renal y problemas cardíacos.
Una solución exitosa podría ser	Un sistema de bomba de insulina automatizado, confiable y preciso que: - Monitoree constantemente los niveles de glucosa en la sangre. - Administre dosis adecuadas de insulina en tiempo real según las necesidades del paciente. - Reduzca el riesgo de complicaciones relacionadas con la diabetes - Mejora la calidad de vida de los pacientes al ofrecer una gestión más efectiva y menos intrusiva de su condición.

Tabla 2.6: Visión del Proyecto: Planteamiento del Problema

Declaración de la posición del producto en el mercado

Para	Pacientes diabéticos que requieren administración regular de insulina.
Quienes	Necesitan una solución automatizada y precisa para monitorizar y regular sus niveles de glucosa en sangre en tiempo real, reduciendo los riesgos asociados con la administración manual de insulina.
El Innopump	Es una bomba de insulina automática controlada por software
Que	Ofrece monitorización continua, administración precisa de dosis y adaptabilidad en tiempo real para mantener niveles óptimos de glucosa en sangre, mejorando significativamente la calidad de vida de los pacientes.
A diferencia de	Las bombas de insulina convencionales y métodos manuales de administración.
Nuestro producto	Se diferencia por su capacidad de autorregulación, precisión, y su enfoque basado en software que considera múltiples variables para garantizar la seguridad y bienestar del usuario.

Tabla 2.7: Visión del Proyecto: Posición del producto en el mercado

Descripciones de las partes interesadas

Nombre	Descripción	Responsabilidades
Paciente Diabético	El usuario final del sistema de bomba de insulina, depende del dispositivo para el monitoreo y administración precisa de insulina.	Proporcionar feedback sobre la funcionalidad y usabilidad del sistema. Seguir las indicaciones y recomendaciones del sistema y del profesional de salud. Mantener el dispositivo en condiciones operativas y reportar cualquier problema.
Profesionales de Salud (Endocrinólogos, médicos de cabecera, etc.)	Expertos en el cuidado de la diabetes y otros trastornos relacionados. Brindan asesoramiento médico a los pacientes y pueden proporcionar retroalimentación sobre el dispositivo.	Asegurarse de que el dispositivo sea adecuado para las necesidades del paciente. Proporcionar educación y entrenamiento sobre el uso del dispositivo. Monitorear la eficacia del tratamiento y ajustar las recomendaciones según sea necesario.
Desarrolladores del Sistema	El equipo responsable de diseñar, desarrollar y mantener el software de la bomba de insulina.	Asegurar que el sistema funcione de manera confiable y segura. Implementar características basadas en las necesidades y feedback de los usuarios y profesionales de salud. Mantener y actualizar el sistema para garantizar su eficacia a lo largo del tiempo.
Reguladores y Organismos de Salud	Entidades que establecen estándares y regulaciones para dispositivos médicos y garantizan que cumplan con los requisitos de seguridad y eficacia.	Evaluar y certificar la seguridad y funcionalidad del dispositivo. Establecer estándares y regulaciones para dispositivos similares. Monitorizar y reportar cualquier problema o defecto asociado con el dispositivo.

Tabla 2.8: Visión del Proyecto: Descripción de las partes interesadas

Entorno de usuario

- **Número de personas involucradas en completar la tarea:**

Generalmente, es una tarea individual ya que se trata de la gestión personal de salud del paciente diabético. Sin embargo, en casos de pacientes menores de edad o aquellos con dificultades, podría involucrarse un cuidador o un familiar.

- **¿Está cambiando esto?**

Con el avance de la tecnología y las interfaces de usuario más intuitivas, es probable que cada vez más pacientes puedan manejar el dispositivo de forma independiente.

- **Duración de un ciclo de tarea:**

El monitoreo de los niveles de glucosa es continuo, mientras que la administración de insulina puede variar dependiendo de los niveles detectados. Un ciclo completo desde la detección hasta la administración podría ser de minutos a horas, dependiendo de las necesidades del paciente.

- **Cantidad de tiempo gastado en cada actividad:**

Monitoreo: Continuo.

Administración de insulina: Puede variar, pero por lo general, la bomba suministra insulina en unos segundos o minutos una vez que se toma la decisión.

- **¿Está cambiando esto?**

Con la mejora de las tecnologías y algoritmos, es posible que los sistemas futuros sean aún más rápidos y eficientes en la toma de decisiones.

- **Restricciones ambientales únicas:**

El sistema debe ser resistente al agua y al sudor, y apto para funcionar en una variedad de entornos, desde interiores hasta exteriores y en diferentes condiciones climáticas.

- **Plataformas de sistema en uso hoy:**

Dispositivos embebidos, generalmente con sistemas operativos especializados y optimizados para tareas específicas.

- **Plataformas futuras:**

Podría haber integración con smartphones o wearables para monitoreo y alertas. También se pueden considerar avances en la computación en la nube para análisis de datos y recomendaciones.

- **¿Necesita su aplicación integrarse con ellas?**

Sería beneficioso integrarse con aplicaciones de monitoreo de salud y diarios de alimentos para obtener una visión más completa del estado del paciente y hacer recomendaciones o ajustes más precisos.

Descripción del producto

Necesidades y características

Necesidad	Prioridad	Características	Planificación de lanzamiento
El sistema debe medir el nivel de azúcar en sangre y suministrar insulina cada 10 minutos, si fuese necesario.	Alta	Sensor de glucosa integrado Alertas personalizables para niveles altos/bajos de glucosa.	Versión 1.0
El sistema debe estar disponible para suministrar insulina cuando sea necesario *(availability)*.	Alta	Monitoreo Continuo: El sistema monitorea constantemente los niveles de glucosa en la sangre y está listo para actuar cuando sea necesario. Sistema de Alerta: El sistema alertará al usuario si se detectan niveles anormales de glucosa en la sangre. Backup Automático: En caso de fallo del sistema principal, un sistema de respaldo toma el control para asegurar la entrega continua de insulina.	Versión 1.0

Necesidad	Prioridad	Características	Planificación de lanzamiento
El usuario puede intercambiar los reservorios de insulina en cualquier instante	Media	Mecanismo de Enganche Fácil: El diseño permite que los reservorios se puedan quitar y reemplazar con facilidad. Detección de Reservorio: El sistema detecta automáticamente cuando un nuevo reservorio se instala y recalibra según sea necesario. Guía Paso a Paso: Instrucciones visuales o auditivas que guían al usuario durante el proceso de intercambio.	Versión 1.1

Tabla 2.9: Visión del Proyecto: Necesidades y Características del producto

Otros requerimientos del producto

Necesidad	Prioridad	Planificación de lanzamiento
El sistema deberá tener un tamaño y peso que sea ergonómico para todo tipo de personas. (*Usability*)	Alta	Versión 2.0
El sistema deberá tener un diseño tal que no permita la configuración o administración accidental de insulina ante el contacto con el panel de control. (*Security*)	Alta	Versión 2.0
El sistema deberá permitir la carga o cambio de baterías en cualquier momento. (*Usability*)	Alta	Versión 1.0
El sistema deberá consumir la menor cantidad de energía posible logrando una autonomía de al menos 1 semana efectiva. (*Efficiency*)	Baja	Versión 3.0

Tabla 2.10: Visión del Proyecto: Otros requerimientos del producto

11.6. Clasificación de requisitos

Requisitos Funcionales

1. El sistema debe medir el nivel de azúcar en sangre y suministrar insulina cada 10 minutos, si fuese necesario.
2. La dosis de insulina a ser suministrada debe ser calculada mediante la medición del nivel actual de azúcar en sangre, comparando éste a un nivel de medición previo.
3. La cantidad de insulina a ser suministrada debe ser calculada según la lectura actual de azúcar, según lo mida el sensor:
   - 3.1. Si la lectura está por debajo del mínimo, no se debe suministrar insulina.
   - 3.2. Si la lectura está entre los parámetros adecuados, la insulina sólo será suministrada si el nivel de azúcar está creciendo y la tasa de incremento del azúcar está aumentando.
   - 3.3. Si la lectura está por sobre el nivel recomendado, la insulina se suministra a menos que el nivel de azúcar en sangre esté decreciendo y la tasa de decremento del mismo esté incrementando.
   - 3.4. La cantidad de insulina realmente suministrada puede ser diferente a la dosis calculada debido a las distintas restricciones de seguridad (*safety*) que se incluyen en el sistema. Hay un límite de la dosis máxima a ser suministrada en cada inyección, y un límite acumulativo por día.
4. El sistema deberá permitir la carga o cambio de baterías al usuario.
5. El sistema debe ejecutar un programa de auto testeo cada 30 segundos.
6. Al inicio de cada período de 24 horas (indicado por clock = 00:00:00), el sistema debe reiniciar la dosis acumulativa de insulina suministrada a 0.
7. El sistema debe permitir al usuario reemplazar el reservorio de insulina por uno nuevo.
8. El sistema debe mantener historiales y registros de mediciones y administraciones de insulina realizadas.
9. El sistema debe admitir un reservorio de 315 ml de insulina.

Requisitos de manejo de errores

10. El sistema debe alertar al usuario la detección de errores en el funcionamiento.
11. El sistema debe detectar errores en el envío y la recepción de señales eléctricas y detener el funcionamiento.
12. El sistema debe detectar una falla de hardware del sensor.
13. El sistema debe detectar voltajes inestables y fallos de conexión de cables.
14. El sistema debe detectar niveles de batería críticamente bajos y alertar al usuario.
15. El sistema debe detectar interferencias provenientes de otros dispositivos.
16. El sistema debe detectar y alertar si la aguja está bloqueada o insertada incorrectamente.
17. El sistema debe detectar una falla de comunicación entre sensores y actuadores.
18. El sistema debe detectar mediciones de glucosa excepcionales.
19. El sistema debe detectar temperaturas extremas que afecten el rendimiento del dispositivo.

Requisitos No Debe (Seguridad o protección contra daños físicos)

20. El sistema no debe permitir al usuario reemplazar el reservorio de insulina durante una administración.
21. El sistema no debe permitir el uso prolongado del dispositivo con niveles críticos de batería que puedan afectar su fiabilidad.
22. El sistema no debe interrumpir el sistema de alarmas ante un fallo en el sistema.
23. El sistema no debe permitir la administración de una sobredosis o subdosis de insulina.
24. El sistema no debe permitir la configuración de dosis/valores excepcionales en parámetros de dosificación.
25. El sistema no debe permitir el uso del dispositivo tras una repetición de 3 alertas de error.
26. El sistema no debe permitir el uso del dispositivo ante una falla eléctrica hasta estabilizar la alimentación de energía.
27. El sistema no debe permitir el uso tras detectar interferencias.
28. El sistema no debe permitir el uso tras detectar una ruptura o bloqueo del dispositivo o la aguja.
29. El sistema no debe permitir el uso prolongado del dispositivo ante la pérdida de conexión entre sensores y actuadores.
30. El sistema no debe permitir el uso prolongado del dispositivo bajo condiciones extremas de temperatura.

Requisitos No Funcionales

Fiabilidad y disponibilidad

31. El sistema debe estar disponible para suministrar insulina cuando sea necesario, presentando una probabilidad de fallo bajo demanda (POFOD) de 0,001, equivalente a una probabilidad de 1/1000 de que se produzca una caída cuando se realiza una demanda.
32. El sistema debe calcular y administrar la dosis correcta de insulina de manera fiable y precisa, presentando una Tasa de ocurrencia de fallos (ROCOF) menor a 0,0001.

Usabilidad

33. El sistema deberá tener un tamaño y peso que sea ergonómico para todo tipo de usuarios.
34. El sistema debe contar con documentación incluyendo manual de usuario.
35. El sistema debe contar con una interfaz intuitiva para todo tipo de usuarios.
36. El sistema debe contar con documentación referida al modo de uso.
37. El sistema debe contar con documentación incluyendo contraindicaciones y reacciones adversas.

Eficiencia

38. El sistema deberá consumir la menor cantidad de energía posible logrando una autonomía de al menos 1 semana efectiva.

11.7. Especificación Formal

A continuación se detalla la especificación del caso de uso "**1. Suministrar dosis de insulina automáticamente**". Este caso de uso es esencial para la seguridad de los usuarios, por eso se describe con el método Z.

En el desarrollo de las especificaciones formales de nuestro sistema, hemos empleado la notación Z, basándonos en la lógica y los principios establecidos en la décima edición del libro "Software Engineering" de Ian Sommerville. Paralelamente, nuestra implementación de la notación Z se ha guiado por el enfoque detallado presentado en "The Z Notation: A Reference Manual" de J. Michael Spivey, lo que nos ha permitido un acercamiento práctico y preciso a esta metodología de especificación formal. Este doble fundamento nos ha proporcionado una base sólida para especificar de manera clara y sin ambigüedades las propiedades y comportamientos del sistema, facilitando así la creación de un software confiable y robusto.

- **Descripción en lenguaje natural. ESTADO_BOMBA_INSULINA**

Este esquema define cómo la bomba de insulina maneja sus operaciones básicas. Incluye la lectura de los niveles de azúcar en la sangre (glucosa), el control de alarmas, pantallas y la administración de dosis de insulina. También establece límites seguros para los niveles de azúcar en la sangre y la cantidad de insulina disponible. Si la dosis acumulada de insulina excede un límite máximo diario, la bomba se detiene para evitar una sobredosis. Por último, especifica cómo se convierte la dosis de insulina a ser administrada en una cadena de texto para su visualización en la pantalla.

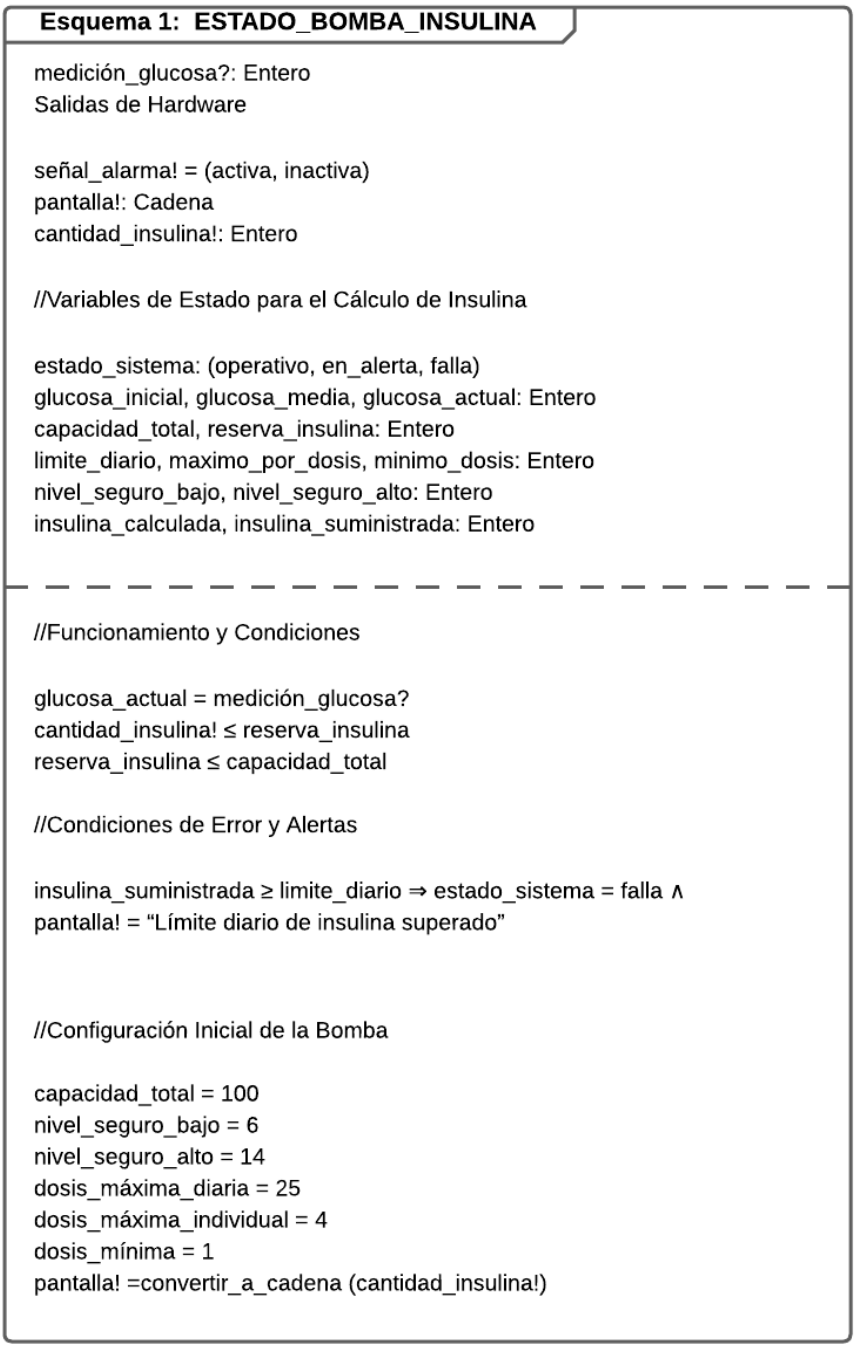


Figura 2.6: Especificación Esquema 1. ESTADO_BOMBA_INSULINA
Basado en (Spivey, 1989)

Descripcion en lenguaje natural: MODO_OPERATIVO

Este esquema se centra en el funcionamiento de la bomba durante su uso normal. Define las condiciones bajo las cuales la bomba administra insulina, como cuando los niveles de azúcar son bajos, normales o altos. Incluye reglas para no administrar insulina si

la dosis calculada es cero, para limitar la dosis si se excede el máximo diario y para ajustar la dosis en función de la cantidad disponible de insulina.

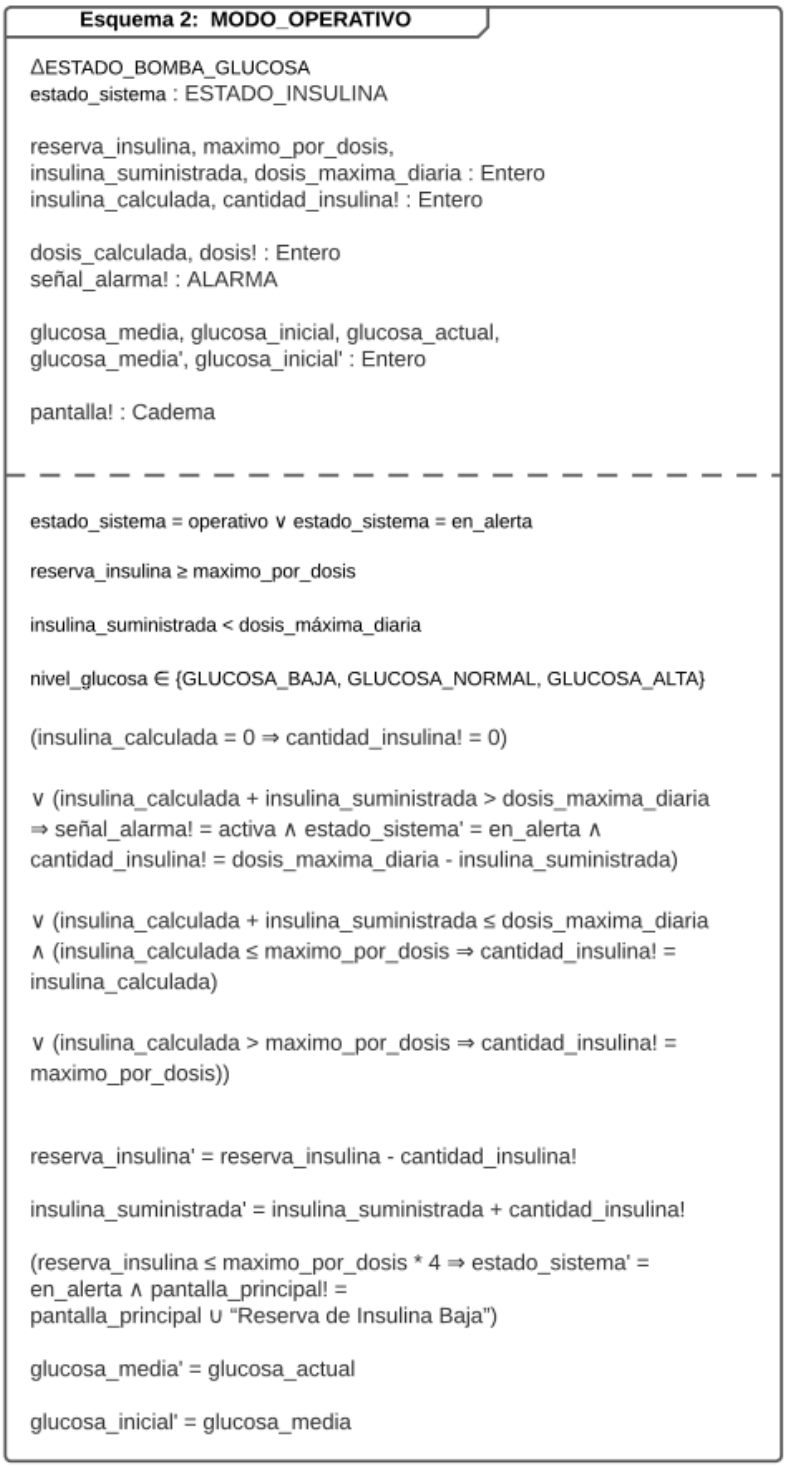


Figura 2.7: Especificación Esquema 2. MODO_OPERATIVO
Basado en (Spivey, 1989)

Descripcion en lenguaje natural : GLUCOSA_BAJA

Este esquema se activa cuando los niveles de azúcar en la sangre están por debajo de un umbral seguro. En este caso, no se administra insulina, se activa una alarma y se cambia el estado de la bomba a alerta. La pantalla muestra un mensaje indicando que el azúcar está baja.

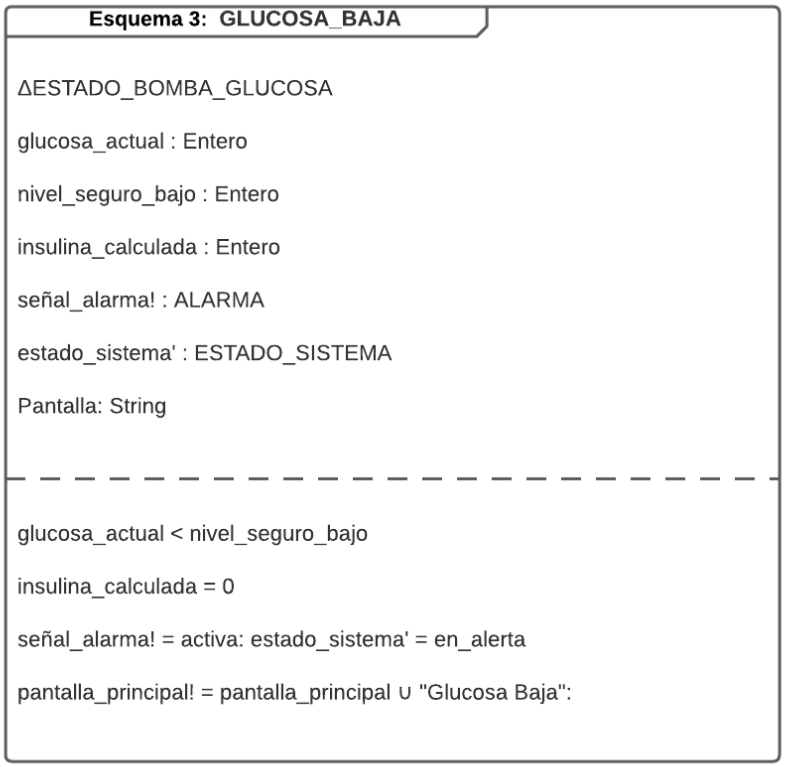


Figura 2.8: Especificación Esquema 3. GLUCOSA_BAJA
Basado en (Spivey, 1989)

Descripcion en lenguaje natural : GLUCOSA_NORMAL

Aquí, la bomba maneja situaciones donde los niveles de azúcar están dentro de rangos seguros. Se establecen diferentes condiciones para ajustar la dosis de insulina basándose en si el nivel de azúcar está estable, aumentando o disminuyendo, y cuán rápido ocurren estos cambios.

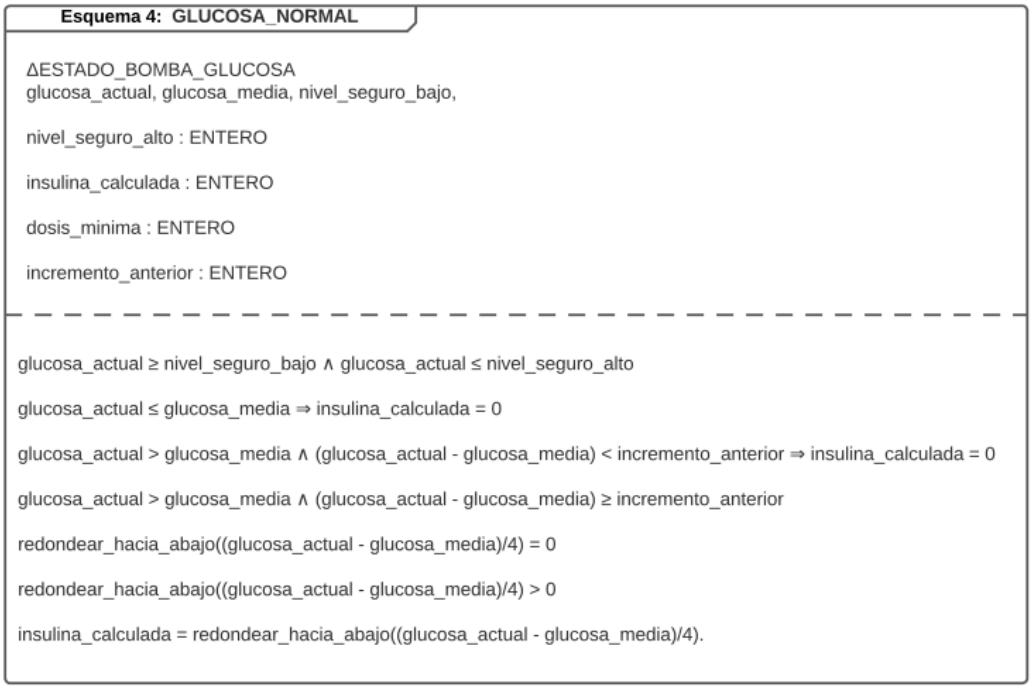


Figura 2.9: Especificación Esquema 4. GLUCOSA_NORMAL
Basado en (Spivey, 1989)

Descripcion en lenguaje natural: GLUCOSA_ALTA

Este esquema se ocupa de situaciones donde los niveles de azúcar en la sangre son demasiado altos. Define cómo calcular la dosis de insulina necesaria, dependiendo de si el nivel de azúcar está aumentando, estable o disminuyendo, y la velocidad de estos cambios.

Esquema 5: GLUCOSA_ALTA

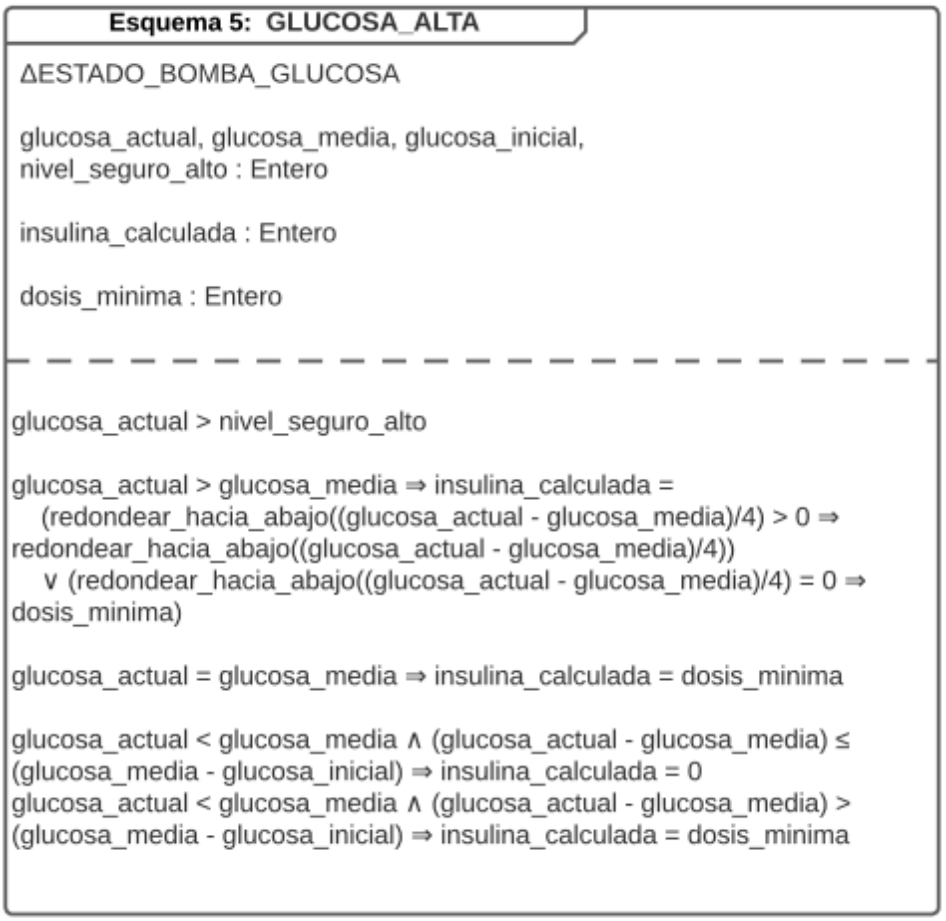


Figura 2.10: Especificación Esquema 5 GLUCOSA_ALTA
Basado en (Spivey, 1989)

Disciplina de Diseño Arquitectónico

11.8. Recopilación de escenarios

1. Suministrar dosis de insulina automáticamente
2. Ejecutar algoritmo de auto testeo
3. Remover aguja
4. Remover reservorio de insulina
5. Reiniciar la cantidad de insulina suministrada en el día

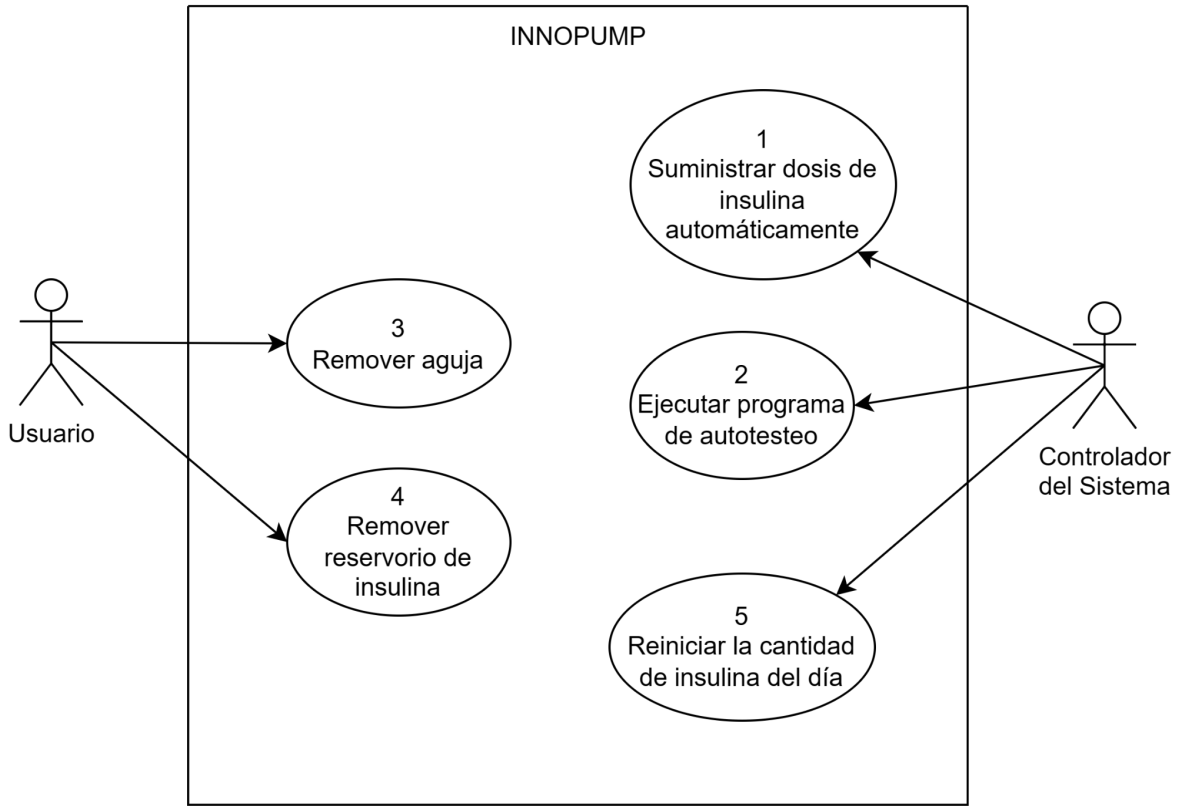


Figura 2.5: Modelo de Casos de Uso para requisitos principales del sistema
Diagrama de Casos de Uso (UML 2.5.1)

11.9. Recopilación de requisitos y restricciones del entorno

Requisitos esenciales para la selección del patrón arquitectónico:

- **Confiabilidad *(dependability)*:** el patrón arquitectónico seleccionado como base debe permitir la aplicación de redundancia y diversidad para asegurar que no se interrumpan o sean ejecutadas de forma incorrecta los servicios críticos del sistema ante una falla. Además, debe permitir la implementación de mecanismos de recuperación, monitoreo, sistemas de notificación y alarmas.

- **Fiabilidad *(reliability)***: el patrón debe permitir al sistema asegurar que provee los servicios como es esperado.
    - **Corrección**: debe garantizar que los servicios del sistema utilizan los datos exactos.
    - **Precisión**: debe garantizar que la información se suministra con el nivel de detalle adecuado, que los algoritmos son consistentes y no brindan resultados ambiguos.
    - **Puntualidad**: garantizar que la información se suministra cuando se necesita, se coordinan eficientemente las tareas y los procesos.
- **Seguridad o protección contra daños físicos *(safety)*:** el sistema debe operar sin fallos catastróficos, por lo que el patrón arquitectónico debe tener algún tipo de sistema de control (monitoreo) de todos los cálculos y algoritmos realizados, datos leídos y producidos, para evitar las imprecisiones en los mismos.

11.10. Identificación de enfoques arquitectónicos

Se tendrán en cuenta los patrones arquitectónicos específicos para Sistemas de software en tiempo real: Observar y reaccionar, Control ambiental y Segmentación de proceso.

Patrón arquitectónico: Observar y reaccionar

Nombre	**Observar y reaccionar (Observe and react)**
Descripción	Se recopilan y analizan los valores de entrada de un conjunto de sensores de los mismos tipos. Dichos valores se despliegan de alguna forma. Si los valores de sensor indican que surgió alguna condición excepcional, entonces se inician acciones para llamar la atención del operador hacia dicho valor y, en ciertos casos, realizar acciones en respuesta al valor excepcional.
Estímulos	Valores de los sensores unidos al sistema.
Respuestas	Salidas a desplegar, activadores de alarma, señales a sistemas que reaccionan.
Procesos	Observador, Análisis, Despliegue, Alarma, Reactor.
Usado en	Sistemas de monitorización, sistemas de alarma.

Tabla 2.11: Patrón arquitectónico “Observar y reaccionar”
Traducción propia de:
(Sommerville, 2016, p. 621, Figure 21.6 The Observe and React pattern)

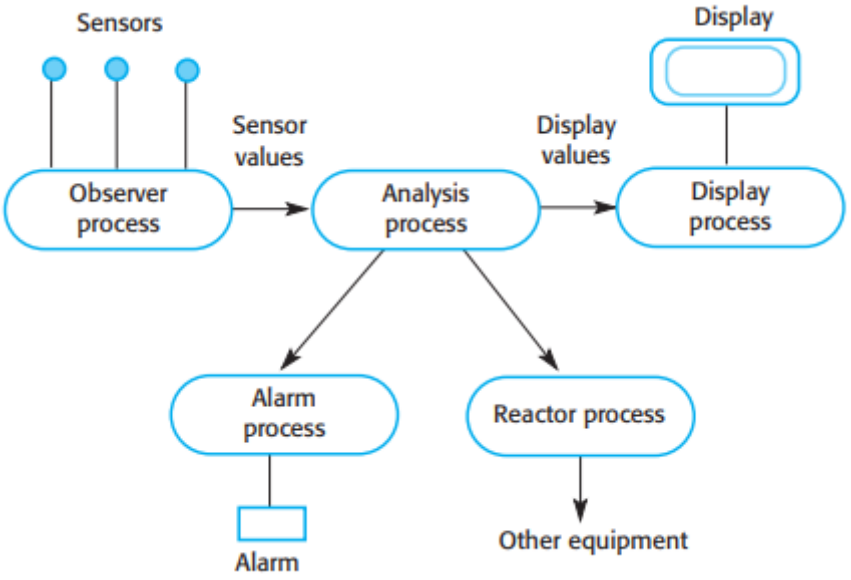


Figura 2.11: Patrón Observar y reaccionar
(Sommerville, 2016, p. 621, Figure 21.7 The Observe and React process structure)

Patrón de hardware: Control ambiental

Nombre	**Control ambiental (Environmental Control Pattern)**
Descripción	El sistema analiza información de un conjunto de sensores que recopilan datos del entorno del sistema. También se puede recopilar más información del estado de los actuadores que se conectan al sistema. Con base en datos de los sensores y actuadores, se envían señales de control a los actuadores, que en ese momento provocan cambios en el entorno del sistema. Puede desplegarse información de los valores del sensor y el estado de los actuadores.
Estímulos	Valores de los sensores unidos al sistema y el estado de los actuadores del sistema
Respuestas	Señales de control a actuadores, despliegue de información.
Componentes	Monitor, de control, de despliegue, controlador de actuador, monitor de actuador.
Usado en	Sistemas de control

Tabla 2.12: Patrón arquitectónico “Control ambiental”
Traducción propia de:
(Sommerville, 2016, p. 623, Figure 21.9 The Environmental Control pattern)

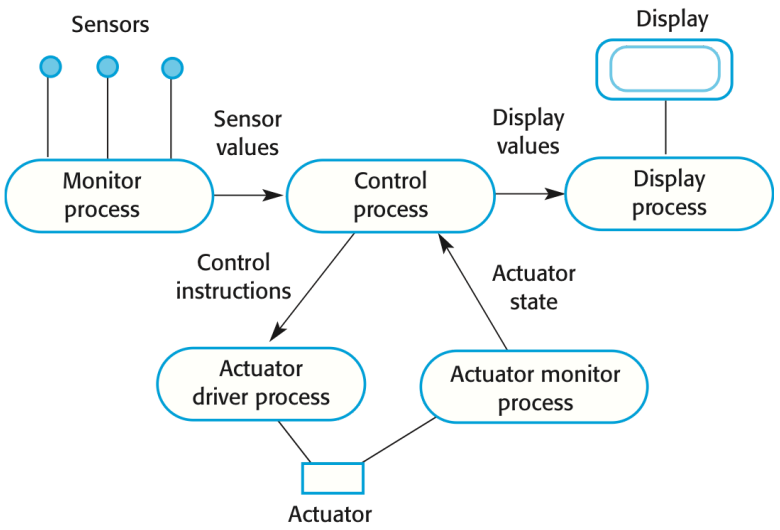


Figura 2.12: Patrón Control Ambiental
(Sommerville, 2016, p. 623, Figure 21.10: The Environmental Control process structure)

Patrón arquitectónico: Segmentación de proceso

Nombre	**Segmentación de proceso (Process pipeline)**
Descripción	Una segmentación (pipeline) de procesos se establece con datos que se mueven en secuencia de un extremo de la "tubería" a otro. Con frecuencia, los procesos están vinculados mediante buffers sincronizados para permitir que los procesos productor y consumidor se ejecuten a diferentes velocidades. La culminación de una segmentación puede desplegarse o almacenar los datos, o la "tubería" puede terminar en un actuador.
Estímulos	Valores de entrada del entorno o algún otro proceso
Respuestas	Valores de salida al entorno o un buffer compartido
Componentes	Productor, Buffer, Consumidor.
Usado en	Sistemas de adquisición de datos, sistemas multimedia.

Tabla 2.13: Patrón arquitectónico "Segmentación de proceso"
Traducción propia de:
(Sommerville, 2016, p. 625, Figure 21.12 The Process Pipeline pattern)

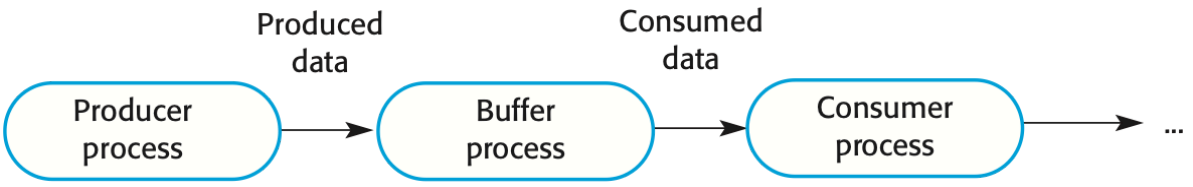


Figura 2.13: Patrón Segmentación de Proceso
(Sommerville, 2016, p. 625, Figure 21.13: Process Pipeline process structure)

11.11. Árbol de utilidad de atributos de calidad

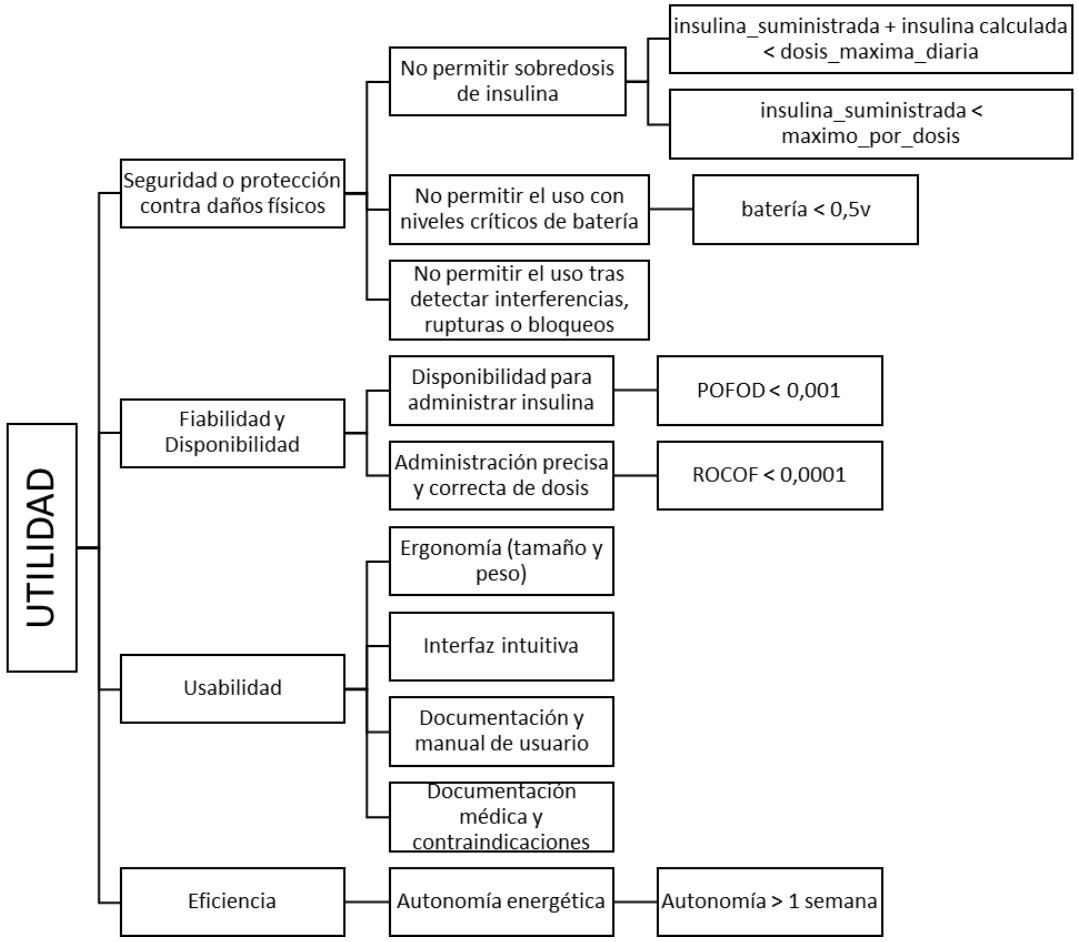


Figura 2.14: Árbol de Utilidad de Atributos de Calidad

11.12. Puntos de sensibilidad y *trade-offs*

Servicios críticos

- Medición de glucosa
- Administración de insulina
- Monitoreo continuo
- Sistema de alertas
- Registro de última medición
- Registro de última dosificación

Servicios no críticos

- Registro histórico de mediciones realizadas
- Registro histórico de dosis administradas

11.13. Evaluación de patrones arquitectónicos

Para el análisis de selección de patrón arquitectónico a ser utilizado se considerará aquella que cumplan con los principales criterios de calidad que están compuestos por requisitos funcionales y no funcionales más críticos y puntos de sensibilidad y *trade-offs*.

Se considerará como principales atributos de calidad a la fiabilidad y protección contra daños físicos.

Segmentación de proceso (Process pipeline)	El sistema de bomba de insulina no es un sistema que requiera de la adquisición de un gran conjunto de datos que deban ser sincronizadas por un buffer por diferencia en velocidad entre procesos consumidor y receptor, esto dado que de forma periódica se verifican los valores del sensor e inmediatamente con los mismos se realizan los cálculos y administración, además no se está recibiendo de forma continua datos a ser procesados. Este patrón afecta la precisión de administración al requerir almacenar inicialmente los datos en el buffer para ser posteriormente consumidos por otros procesos, en consecuencia deteriorando el efecto de la insulina. No cuenta con componentes de control y monitoreo de actuadores requeridos para comprobación de posibles fallas en el sistema.requeridos para comprobación de posibles fallas en el sistema.

Tabla 2.14: Evaluación de patrón Segmentación de proceso

Observar y reaccionar (Observe and react)	Cuenta con componentes de monitoreo de estado de sensores, utilizable para la verificación del correcto funcionamiento de componentes críticos así como el envío de señales de control a lo actuadores (sistema de administración), permitiendo tomar medidas de control como suspensión del servicio de administración de insulina ante una fallo en el mismo y así garantizar la fiabilidad y protección contra daños físicos.

Tabla 2.15: Evaluación de patrón Observar y Reaccionar

Control ambiental (Environmental Control Pattern	Cuenta con las ventajas de observar y reaccionar además del monitoreo del entorno, es decir conocer el estado del actuador, permitiendo así determinar sí el sistema de administración de insulina está en funcionamiento como respuesta de las señales de control y sí la misma se encuentra colocada de forma adecuada.

Tabla 2.16: Evaluación de patrón Control Ambiental

Resumen:

Patrón arquitectónico	**Ventajas**	**Desventajas**
Segmentación de proceso (Process pipeline)	- Eficiente para la recepción de datos de forma continua y a diferentes velocidades respecto al proceso consumidor. - Útil cuando se requiere de varios procesos de transformación o validación de control antes de hacer uso de los datos.	- Mayor latencia producto de la segmentación de procesos y guardado de datos en buffer antes de su procesamiento. - No se recibe un gran conjunto de datos y de forma continua que deban ser procesadas y sincronizadas entre procesos receptor y consumidor. - No cuenta con componentes de monitorización y control sobre actuador.
Observar y reaccionar (Observe and react)	- Cuenta con componentes de monitoreo de valores de sensores de forma periódica. - Aprovecha los recursos al reaccionar solo cuando es necesario(ante valor excepcional). - Ejecución de acciones por medio de actuación ante valores excepcionales(suspender servicio). - Es flexible y puede adaptarse a distintas situaciones. - Proporciona respuestas rápidas que permiten el control en tiempo real	- No cuenta con monitoreo del entorno (actuadores), lo que no permite determinar sí el mismo está funcionando de forma deseada o tomar acción en función de su estado.
Control ambiental (Environmental Control Pattern	- Realiza el monitoreo de su entorno (actuador), lo que permite saber de su estado y realizar ajustes precisos teniendo un mayor control en los cambios del entorno.	- Mayor costo y complejidad de recursos por registro y control sobre actuadores (sensores avanzados).

Tabla 2.16: Evaluación de patrones arquitectónicos

Selección del patrón arquitectónico inicial

Se selecciona el patrón **Observar y reaccionar (Observe and React)** debido a su adaptabilidad al desarrollo de sistemas fiables, confiables y seguros, su flexibilidad para la aplicación de técnicas de redundancia y diversidad, su simplicidad para implementar un sistema de control o automonitoreo. Durante la administración de insulina al detectarse niveles de glucosa altos esto genera una suspensión de la administración de insulina. Por lo que no es requerido un monitoreo constante del actuador.

11.14. Refinamiento de la arquitectura

Arquitectura de Automonitoreo

Los sistemas con una arquitectura de automonitoreo se encargan de controlar su propio funcionamiento y tomar una acción si se detecta un problema. El sistema dispondrá de más de un canal que realizará los cálculos y comparará los resultados: si las salidas difieren, se supone una caída y se lleva al sistema a un estado seguro *(fail-safe state)*.

Esta arquitectura implica utilizar software y hardware diverso en cada canal, para reducir la probabilidad de que se repita el error en ambos canales. (Sommerville, 2016, p. 320)

Se incluirá un canal de monitoreo al patrón base elegido *Observe and React*:

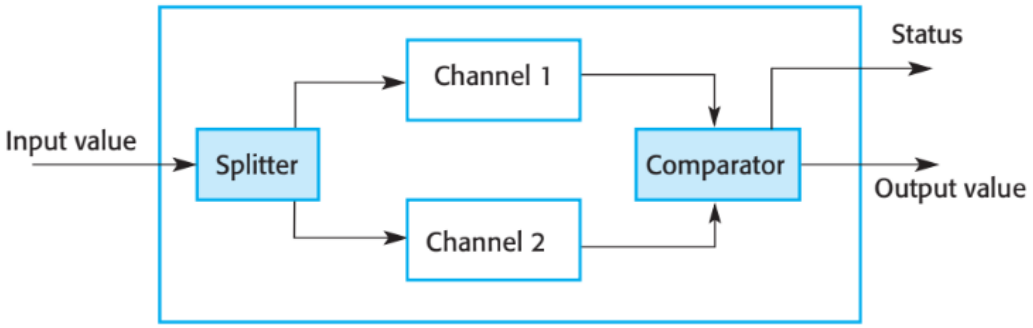


Figura 2.15: Arquitectura de Automonitoreo
(Sommerville, 2016, p. 320 Figure 11.7: Self-monitoring architecture)

Redundancia y Diversidad

La redundancia consiste en incluir más de una sola versión de ciertos componentes, de manera que ante una falla, otra estará disponible.

Consideramos fundamental contar con un **sistema redundante de respaldo** en caso de fallos **en la administración de insulina**, que tome el control del sistema, intervenga deteniendo el proceso de administración que fallo, y realice la administración de insulina el sistema de respaldo mientras se detecta y arregla el fallo del sistema principal. Esto nos brinda más seguridad y continuidad en el funcionamiento del sistema.

Debido a que decidimos aplicar el patrón arquitectónico Observe and React, se deberá evaluar alternativas para implementación de redundancia: aplicar únicamente redundancia en los algoritmos del cálculo de la dosis (1), implementar redundancia de hardware para realizar los cálculos en dos componentes separados (2), o aplicar redundancia de componentes de hardware tanto para el cálculo como para la captura de datos de entrada (sensores) del sistema (3).

En la bomba de insulina, la redundancia será fundamental para garantizar la precisión, evitar cálculos de dosis incorrectos o inconsistentes y, en caso de detectar un mal funcionamiento del dispositivo, entrar en un modo de falla segura *(fail-safe)* que alerte al usuario del problema y descontinúe la administración de insulina.

La diversidad consiste en aplicar funcionalidades de sistema de diferentes maneras, con diferentes componentes, de manera que no fallen de la misma forma. (Sommerville, 2016, p. 295)

Para la bomba de insulina es fundamental plantear un conjunto de algoritmos que apliquen diferente lógica para el cálculo de la dosis de insulina a administrar. El objetivo es obtener resultados precisos y poder compararlos, para detectar posibles inconsistencias de resultados o valores excepcionales.

Alternativas de técnicas de redundancia y diversidad:

A) Sin redundancia de hardware, solo de software.
B) Redundancia de hardware sólo en el monitoreo (1 sensor, 2 microprocesadores).
C) Redundancia de hardware en monitoreo y en sensores (2 sensores, 2 microprocesadores)

Comparación de alternativas de diseño aplicando redundancia y diversidad:

Alternativa			**Ventajas**	**Desventajas**
1	Hardware	**Sin redundancia**	- Costo bajo. - Implementación simple.	- Fiabilidad baja. - Un fallo en la captura de datos podría no ser identificado. - Un fallo en el procesamiento podría no ser identificado. - Ante una caída, el sistema dejará de funcionar.
	Software	Redundancia y diversidad lógica		
2	Hardware	Redundancia en procesamiento: **1 sensor** **2 microprocesadores**	- Costo moderado. - Un fallo en la captura de datos puede ser identificado y corregido.	- Fiabilidad media. - Ante una caída del sensor, el sistema dejará de funcionar.
	Software	Redundancia y diversidad lógica		
3	Hardware	Redundancia en captura y procesamiento: **2 sensores** **2 microprocesadores**	- Fiabilidad alta. - Un fallo en la captura de datos puede ser identificado y corregido. - Un fallo en el procesamiento puede ser identificado y corregido. - Ante una caída en un sensor o en un microprocesador, el otro dispositivo puede soportar el sistema a un nivel menor de fiabilidad.	- Costo muy alto. - Afecta la experiencia del usuario. - Implementación compleja.
	Software	Redundancia y diversidad lógica		

Tabla 2.18: Comparación de alternativas de aplicación de redundancia y diversidad

11.15. Modelado de la Arquitectura

Para el modelado de la arquitectura implementaremos el modelo 4+1:

Vista lógica:

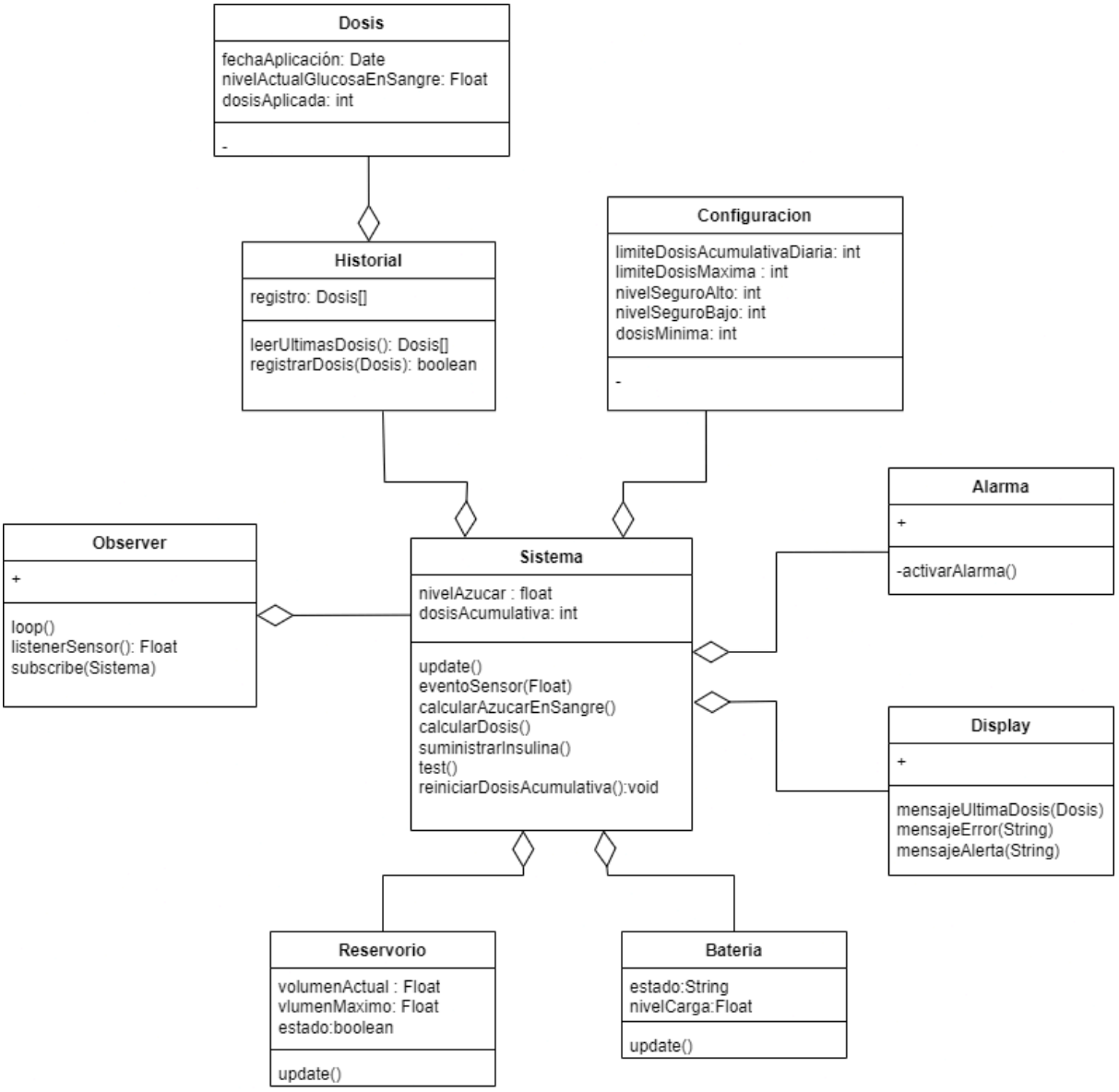


Figura 2.16: Vista lógica del sistema
Diagrama de Clases (UML 2.5.1)

Para apreciar el diagrama de clases de mejor forma, accede al link.

En la Figura 2.16 se puede observar las principales clases, con sus atributos y métodos que interactúan entre sí en el sistema, con sus respectivas relaciones enfocados en la arquitectura Observer and React.

Vista de procesos

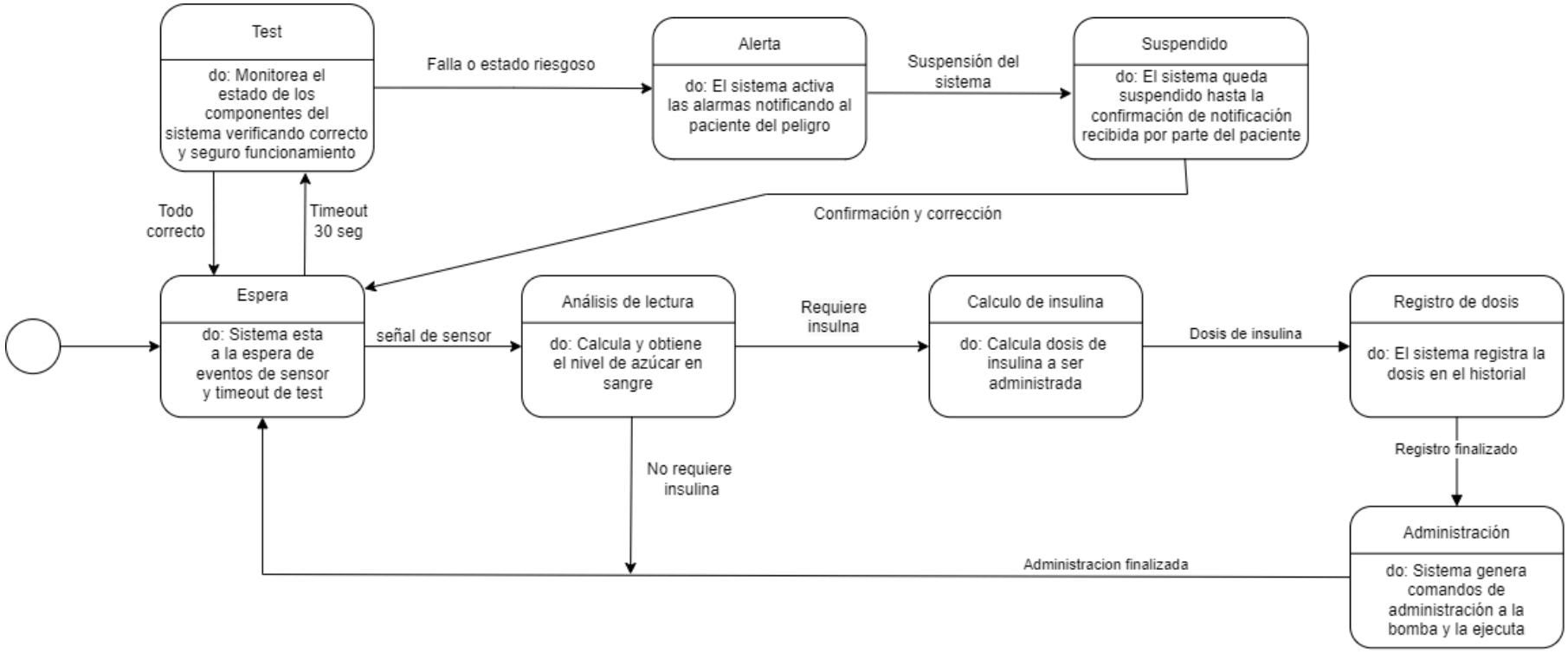


Figura 2.17: Vista de procesos del sistema
Diagrama de Estados (UML 2.5.1)

Para apreciar el diagrama de clases de mejor forma, accede al link.

La Figura 2.17 muestra un diagrama de estado que ilustra el flujo de proceso del sistema, donde cada evento producirá cambio de estado en el sistema. El sistema inicia en un estado de “Espera”, ante un evento de timeout (finalización de tiempo) cada 30 segundos pasa al estado de “Test” donde se ejecuta el automonitoreo del sistema en busca de fallas o estado de riesgo considerados por el sistema (como en nivel de batería por debajo de 0.5V), sí es producido el evento detectado algún fallo o estado de riesgo, el sistema pasa al estado de “Alerta” donde se enviaran notificación auditiva y visual al paciente, posteriormente pasará al estado suspendido hasta ser confirmado la recepción de la respuesta y corregido el fallo, pasando de ser así nuevamente al estado en “Espera” o activo.

Ante un evento de recepción de señal de sensor se pasará al estado de “Analisis de lectura” donde se calculará el nivel de azúcar en sangre y de ser requerido su administración pasará al estado “Cálculo de insulina” donde una vez obtenido la dosis a ser administrada pasar por los estado “Registro de dosis” y “Administración” una vez finalizado el registro, finalmente luego de la administración pasará al estado de “Espera”. Como se observa este es un sistema es continuo, es decir, no finaliza y está pendiente a los eventos todo el tiempo.

Vista de desarrollo y física:

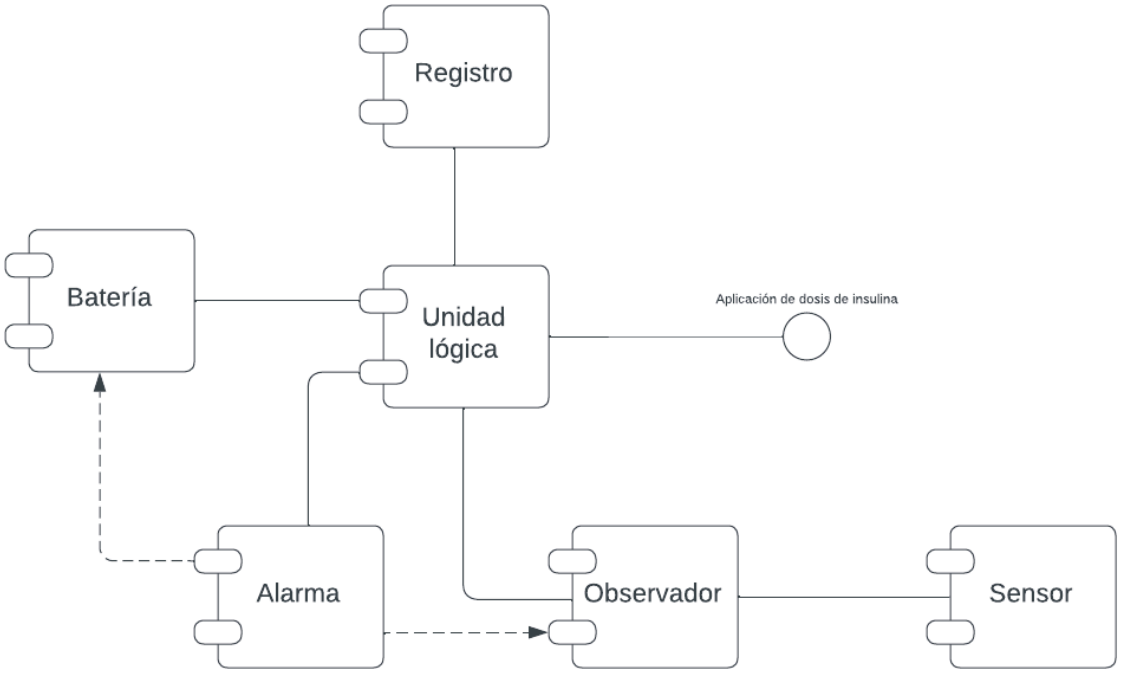


Figura 2.18: Vista de desarrollo y física del sistema
Diagrama de Componentes (UML 2.5.1)

Para apreciar el diagrama de clases de mejor forma, accede al link.

La Figura 2.18 despliega un sistema de bomba de insulina con una arquitectura modular, centrada en una unidad lógica que procesa los datos de glucosa recopilados por un sensor y administra la dosificación de insulina. Los datos son almacenados en un registro y supervisados por el sistema para garantizar la precisión del tratamiento. Una batería asegura la autonomía del sistema y una alarma integrada sirve para alertar en caso de dosis fuera del umbral permitido, o niveles bajo de la batería. Se enfoca en la arquitectura en la arquitectura **Observer and React.**

Proyección del Desarrollo

El desarrollo del sistema Innopump se desarrollará de manera iterativa e incremental. Esta metodología se alinea perfectamente con nuestra necesidad de adaptar y mejorar continuamente el sistema según las exigencias del mercado y las necesidades de los usuarios.

Para futuras iteraciones, tenemos planeado expandir las funcionalidades siendo esta una plataforma Android, proporcionando a los usuarios una interfaz accesible para interactuar con el sistema Innopump. Esta aplicación estará desarrollada utilizando JavaScript con React Native, aprovechando su eficiencia y flexibilidad para crear una experiencia de usuario fluida y responsiva.

La comunicación entre la aplicación y el sistema Innopump será crucial, permitiendo el respaldo y análisis de datos médicos importantes como los niveles de azúcar en sangre y las dosis de insulina administradas. Además, el nuevo sistema se integrará con un servidor en la nube, desarrollado con Java y utilizando Hibernate para manejar eficientemente la persistencia de los datos. Esta integración asegurará que los datos del paciente estén seguros y accesibles para el médico autorizado, facilitando un monitoreo en tiempo real del estado del paciente.

En el lado del hardware, el software embebido en el sistema Innopump será programado en el lenguaje C, dada su eficacia en sistemas de bajo nivel y su capacidad para interactuar eficientemente con el hardware de la bomba de insulina.

Para garantizar la calidad y confiabilidad del software, implementaremos estrategias de prueba exhaustivas. Estas incluirán pruebas unitarias para validar cada componente individual, pruebas de integración para asegurar la correcta interacción entre los diferentes módulos y sistemas, y pruebas de usuario para validar la experiencia general del usuario y la funcionalidad de la aplicación. Además, dada la naturaleza crítica de la aplicación en términos de salud, se realizarán pruebas rigurosas de seguridad y conformidad para cumplir con los estándares médicos y legales.

Este enfoque de desarrollo nos permitirá lanzar una versión del producto de manera oportuna, respondiendo así a las exigencias del mercado, y posteriormente ir mejorando de manera continua basándonos en el uso real y los comentarios de los usuarios como también mejorando la competitividad del producto.

En el desarrollo iterativo e incremental del sistema Innopump, implementaremos un enfoque robusto de pruebas que incluirá el Análisis Estático para la validación del código, utilizando herramientas especializadas como SonarQube y SonarLint.

SonarQube es una plataforma líder en análisis de calidad y seguridad del código. En nuestro proyecto, se utilizará para realizar revisiones automáticas con el fin de detectar bugs, vulnerabilidades de seguridad y “code smells” (códigos que necesitan ser mejorados). Esto es especialmente relevante en el contexto de la salud, donde la precisión y confiabilidad del software son críticas. SonarQube proporcionará un dashboard detallado, permitiéndonos visualizar y gestionar la calidad del código a lo largo del tiempo, lo cual es esencial para mantener altos estándares de calidad y seguridad en el software.

SonarLint, por otro lado, es un plugin IDE que se integra con herramientas de desarrollo como IntelliJ IDEA, Eclipse, Visual Studio, y VSCode. Funciona como un 'linter' en tiempo real, proporcionando feedback inmediato mientras se escribe el código, lo cual ayudará a nuestro equipo a detectar y corregir problemas de calidad y seguridad al momento de la codificación. Esto es particularmente útil para prevenir problemas antes de que el código se integre en la base principal, facilitando un desarrollo más eficiente y reduciendo la carga de trabajo en las etapas posteriores de testing y revisión.

La integración de SonarQube y SonarLint en nuestro flujo de trabajo de desarrollo y validación es fundamental para garantizar que el software del sistema cumpla con los estándares más altos de calidad y seguridad. Estas herramientas nos permitirán identificar y solucionar problemas de manera temprana y continua.

Además de las pruebas unitarias, de integración y de usuario, el análisis estático con SonarQube y SonarLint agrega una capa adicional de seguridad y calidad, ayudando a prevenir errores y vulnerabilidades que podrían afectar la funcionalidad y confiabilidad del sistema y la aplicación.

CONCLUSIONES

12. Cumplimiento de objetivos del Trabajo

ASIMILAR ASPECTOS DE DIVERSIDAD EN LA INGENIERÍA DEL SOFTWARE

En el transcurso de nuestro proyecto, hemos reconocido la importancia de adaptar nuestro enfoque de ingeniería de software según las características y necesidades específicas del sistema que estamos desarrollando. Al estudiar y aplicar diferentes metodologías y herramientas, hemos aprendido a seleccionar y combinar aquellas que mejor se ajustan a cada tipo de aplicación en nuestro caso sistema embebido un software genérico. Esta adaptabilidad no solo ha mejorado la calidad y eficiencia de nuestro trabajo, sino que también nos ha permitido abordar de manera más efectiva los desafíos específicos de cada tipo de sistema, como la seguridad en sistemas embebidos. Consideramos que hemos cumplido satisfactoriamente este objetivo al integrar y adaptar con éxito diversos aspectos de la ingeniería del software en nuestro trabajo.

IDENTIFICAR Y CARACTERIZAR SISTEMAS SOCIO-TÉCNICOS CRÍTICOS

Durante nuestro proyecto, hemos profundizado en la comprensión de los sistemas socio-técnicos, enfocándonos en la interdependencia entre software, hardware, y los elementos no técnicos como procesos, personas y normativas. Hemos aprendido que estos sistemas son más que la suma de sus partes individuales y que su eficacia se manifiesta plenamente cuando todos los componentes trabajan en conjunto. Además con la bomba de insulina, hemos reconocido la importancia de considerar las consecuencias de las fallas en el software para el sistema en su totalidad. Entendemos que una falla en el software puede tener implicaciones mucho más allá de la esfera digital, afectando el entorno físico y humano. Esta comprensión holística de los sistemas socio-técnicos nos ha permitido no solo identificar y caracterizar estos sistemas críticos, sino también diseñar soluciones más efectivas y resilientes.Consideramos que logramos con éxito el objetivo de identificar y caracterizar sistemas socio-técnicos críticos.

ASIMILAR CONCEPTOS INHERENTES A LA INGENIERÍA PARA LA CONFIABILIDAD Y SEGURIDAD DE SISTEMAS

Durante el proyecto también hemos detectado algunos conceptos que para un sistema socio-técnico crítico no pueden faltar, que son la Confiabilidad y la Seguridad (seguridad física y lógica) y de la importancia de los mismos para el desarrollo del trabajo realizado. Por lo tanto, consideramos que cumplimos este objetivo.

SELECCIONAR SISTEMÁTICAMENTE PROCESOS PARA DESARROLLAR PRODUCTOS SOFTWARE CONFIABLES

En el desarrollo de nuestro proyecto, hemos dedicado especial atención a la asimilación de conceptos clave relacionados con la confiabilidad y la seguridad de los sistemas. Hemos comprendido que la confiabilidad no se limita solo a la ausencia de fallos en el software, sino que también implica la capacidad del sistema para funcionar de manera efectiva y predecible en diversas condiciones. Esto incluye la gestión de errores, la resistencia a fallos y la capacidad de recuperación ante incidentes.

ASIMILAR ASPECTOS INHERENTES A ACTIVIDADES FUNDAMENTALES DE LA INGENIERÍA DEL SOFTWARE EN PROCESOS DE DESARROLLO CONFIABLES
En el transcurso de nuestro proyecto, adoptamos una metodología sistemática en la elección de los procesos de desarrollo de software, con el objetivo de que cada fase del proyecto aporte directamente a la creación de un producto final confiable. Estamos convencidos de que este enfoque meticuloso nos ha permitido alcanzar satisfactoriamente el objetivo establecido.

ASIMILAR CONCEPTO DE VISIBILIDAD DE PROCESOS DE DESARROLLO. CONSTRUCCIÓN DE META-MODELOS:
En nuestro proyecto, hemos hecho un esfuerzo consciente para asimilar el concepto de visibilidad en los procesos de desarrollo de software, utilizando la construcción de meta-modelos como una herramienta clave. Entendemos que la visibilidad en el desarrollo de software se refiere a la claridad, transparencia y trazabilidad con la que se pueden observar y comprender los procesos, decisiones y progresos a lo largo del ciclo de vida del desarrollo.

CONCEBIR, ELABORAR, PROYECTAR Y CONSTRUIR PRODUCTOS SOFTWARE CONFIABLES
El grupo considera que se cumplió con éxito el objetivo de concebir, elaborar, proyectar y construir productos de software confiables. Este proceso se inició con la realización de cada una de las actividades tomando en cuenta todos los conceptos desarrollados en la materia.

13. Limitaciones del producto

Dentro de las limitaciones generales en nuestro proyecto de Innopump se encuentra la inexperiencia en el ámbito de la medicina. A pesar de que como estudiantes de ingeniería de software estamos capacitados en el desarrollo de sistemas, nuestro conocimiento sobre medicina, y en particular sobre el tratamiento de la diabetes, es básico. Si bien tenemos una comprensión general sobre la salud, la enfermedad y el funcionamiento de la bomba de insulina, nuestra formación no nos ha provisto del conocimiento detallado en conceptos médicos. Esto representa una limitación significativa, ya que los detalles que se nos puedan escapar en este ámbito podrían ser críticos, impactando nuestra capacidad para comprender completamente las necesidades clínicas del sistema y las implicaciones de cada decisión de diseño en el bienestar del paciente.

Dado que el software de Innopump es un sistema embebido que opera en conjunto con hardware (sensores y la bomba de insulina), nuestra limitada experiencia en este campo también representa un desafío. No estamos completamente preparados para anticipar, diagnosticar y solucionar problemas relacionados con el hardware, lo cual es crucial dado que cualquier fallo en estos componentes podría tener consecuencias graves para la salud de los usuarios.

Adicionalmente, enfrentamos el desafío de comprender y adherirnos a las leyes y regulaciones pertinentes. Innopump, por su impacto potencial en la salud, se encuentra en

una área altamente regulada. Nuestra falta de familiaridad con las regulaciones específicas del sector de la salud y la dificultad para acceder a información detallada y comprensible sobre estas normativas plantean un desafío importante. Esto es crucial para la viabilidad del proyecto y para evitar riesgos legales.

14. Evolución del producto

El equipo considera que la evolución del software desarrollado podría evolucionar.

Línea de producto de software

Innopump podría convertirse en una línea de producto de software, siendo el software desarrollado el primer ejemplar de la familia de productos de software. La ventaja de llevar a cabo esta técnica de reutilización de software

- **Desarrollo acelerado:** Al reutilizar componentes de los ejemplares de la familia de software se reducirá el tiempo de desarrollo del sistema.
- **Reducción de riesgos de desarrollo:** Al no necesitar desarrollar el software desde cero, se reduce el margen de error en el desarrollo.
- **Aumento en la confiabilidad:** Ya que estaremos reutilizando componentes de un sistema ya probado, la confiabilidad del sistema aumentará.

La implementación de una línea de producto de software también brinda la posibilidad de ampliar el alcance del sistema, incluyendo por ejemplo:

- Conexión con dispositivos externos
- Control manual del sistema
- Monitoreo profesional constante de los resultados captados.

Dentro de la línea de producto de software resulta interesante las actividades identificadas a **agregar dentro de proceso de desarrollo de software**:

Disciplina de Requerimientos

Línea de producto de Software

Dentro de la ingeniería de requisitos se propone modificar el **análisis del mercado**, de manera que este se enfoque en los productos de software disponibles en el mercado y en los desarrollados ya por la línea de producto de la empresa.

Disciplina de Diseño Arquitectónico

Familia de producto de Software

Identificar potenciales miembros de la familia de producto a reutilizar

En esta actividad se identificarán los miembros de la familia de productos de software que tengan requisitos similares a los requisitos identificados, como posibles sistemas a reutilizar y seleccionar uno de estos como el sistema a reutilizar.

De esta actividad obtendremos el artefacto “software a reutilizar”, el cual lo utilizaremos para identificar las pruebas a reutilizar y la arquitectura del software a

considerar para la actividad de “Análisis de enfoques arquitectónicos” del metamodelo de diseño.

ANEXOS

ANEXO A: CO EVALUACIÓN

Co Evaluaciones de los Criterios del Trabajo

1. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Completitud y Puntualidad”.
    - **Aspecto positivo:** No se encontró aspectos positivos.
    - **Aspecto regular:** Se entregó el proyecto en el tiempo determinado.
    - **Aspectos negativos:** Se realiza la entrega en tiempo pero no completo, se requirió de más tiempo de lo especulado para el desarrollo del sistema.

2. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Adecuación de la producción a los lineamientos (objetivos, consignas y criterios) que la fundamentan.”
    - **Aspecto positivo:** Se cumplió con los objetivos, consigas y criterios planteado para la producción del proyecto.
    - **Aspecto regulares:** No se encontró aspectos regulares.
    - **Aspectos negativos:** No se encontró aspectos negativos.

3. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Estructura o lógica interna del informe elaborado: portada, índice/s de contenido, figuras, tablas y ecuaciones, capítulos de introducción, desarrollo y conclusiones; con esquema enumerado de capítulos, secciones y subsecciones, y numeración de páginas.“.
    - **Aspecto positivo:** El informe elaborado mantiene una lógica interna ordenada y con la estructura propuesta por la cátedra, se buscó fomentar la claridad y la visibilidad del proceso.
    - **Aspecto regulares:** No se encontró aspectos regulares.
    - **Aspectos negativos:** No se encontró aspectos negativos.

4. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Redacción clara y sin errores gramaticales y ortográficos”.
    - **Aspecto positivos:** Se realizó una redacción clara y sintética de los puntos desarrollados, acompañada de una revisión sistemática del equipo a cargo en revisión de posibles errores o inconsistencias.
    - **Aspecto regulares:** Se realizó una revisión sistemática por parte del equipo pero podría encontrarse errores ortográficos o de falta de claridad.
    - **Aspectos negativos:** No se encontró aspectos negativos

5. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Numeración de Figuras, Tablas o Cuadros, y Ecuaciones”.

- **Aspecto positivos:** Las figuras, tablas y ecuaciones utilizadas y desarrolladas para el informe se encuentran enumeradas y se las ubicó de manera tal que sea útil su aporte en el contexto idóneo.
- **Aspecto regulares:**No se encontró aspectos regulares.
- **Aspectos negativos:** No se encontró aspectos negativos

6. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Precisión en el empleo de nociones básicas, y reconocimiento de nexos entre aspectos conceptuales y prácticos”
   - **Aspecto positivos:** El informe refleja el conocimiento básicos adquiridos y empleados durante el desarrollo del informe, basada en la bibliografía brindada por la cátedra y de material académico acorde al caso de estudio.
   - **Aspecto regulares:** No se encontró aspectos regulares.
   - **Aspectos negativos:** No se encontró aspectos negativos

7. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Correctitud y coherencia del análisis, comparación, integración y síntesis creativas de los contenidos cubiertos”.
   - **Aspecto positivos:** Se integró el conocimiento adquirido y se lo implementó al caso de estudio seleccionado, se realizó una comparación y análisis de los conceptos relevantes al caso de estudio.
   - **Aspecto regulares:** La limitación del caso de estudio también limitó el uso de más conceptos adquiridos en la cursada, por ejemplo la no interacción con componentes externos nos limitó la utilización de conceptos relacionados a la **seguridad de los datos y contra instrucciones y confidencialidad,**
   - **Aspectos negativos:** No se encontró aspectos negativos

8. Aspectos positivos, regulares y negativos con respecto al criterio de evaluación “Ajustado manejo de la bibliografía y CITAS correspondientes. Honrar autor/es refiriendo a su obra (incluir en la cita: el número de página en la obra desde la que parafrasean argumentos)”.
   - **Aspecto positivo:** En el informe se buscó honrar a los autores de los cuales se hizo uso de las citas, conceptos e ideas planteadas por los mismos y utilizadas en el análisis y desarrollo del caso de estudio.
   - **Aspecto regulares:** Aun con la búsqueda puede que dentro del informe se encuentre conceptos, ideas o citas que no hayan sido mencionadas o correctamente citadas.
   - **Aspectos negativos:** No se encontró aspectos negativos

Evaluación:

- Subcriterio 1: 60%
- Subcriterio 2: 100%
- Subcriterio 3: 100%
- Subcriterio 4: 80%
- Subcriterio 5: 100%
- Subcriterio 6: 100%
- Subcriterio 7: 70%
- Subcriterio 8: 70%

Calificación final: 680% / 8 = 85%

Bibliografía

Accu-Check. (2020). *Accu-Chek Spirit Combo*. Accu-Chek. Retrieved September 20, 2023, from https://www.accu-chek.com.ar/infusor-de-insulina/combo

Bosch, J. (2000). *Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach*. Addison-Wesley.

Bucanac, C., & University of Karlskrona/Ronneby. (1999, 04 01). *The V-Model* (1.22 ed.).

Chen, L., Ali Babar, M., & Nuseibeh, B. (2015, 01 16). *Characterizing Architecturally Significant Requirements*. ResearchGate. Retrieved 10 3, 2023, from https://www.researchgate.net/publication/255569055_Characterizing_Architecturally_Significant_Requirements

da Cunha, A. M., Vieira Dias, L. A., & Valdivia de Matos, H. L. (2014, 10). *USING DESIGN PATTERNS FOR SAFETY ASSESSMENT OF INTEGRATED MODULAR AVIONICS*. ResearchGate. Retrieved 10 03, 2023, from https://www.researchgate.net/publication/267327795_USING_DESIGN_PATTERNS_FOR_SAFETY_ASSESSMENT_OF_INTEGRATED_MODULAR_AVIONICS

EcuRed. (2011, 6 29). *Architecture Trade-off Analysis Method (ATAM)*. EcuRed. Retrieved September 20, 2023, from https://www.ecured.cu/Architecture_Trade-off_Analysis_Method_(ATAM)

Kazman, R., Klein, M. H., & Clements, P. C. (2000, 08 1). *ATAM: Method for Architecture Evaluation*. SEI Blog. Retrieved 10 2, 2023, from https://insights.sei.cmu.edu/library/atam-method-for-architecture-evaluation/

Krutchen, P., & Rational Software Corporation. (1995, 11). *Architectural Blueprints—The "4+1" View Model of Software Architecture*. UBC Computer Science. Retrieved September 20, 2023, from https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

Sljivo, I., Uriagereka, G. J., Puri, S., & Gallina, B. (2020). Guiding assurance of architectural design patterns for critical applications. *Journal of Systems Architecture*, 7-8.

Sommerville, I. (2016). *Software Engineering* (Tenth ed.). Pearson.

Spivey, J. M. (1989). *The Z notation: a reference manual*. Prentice Hall.