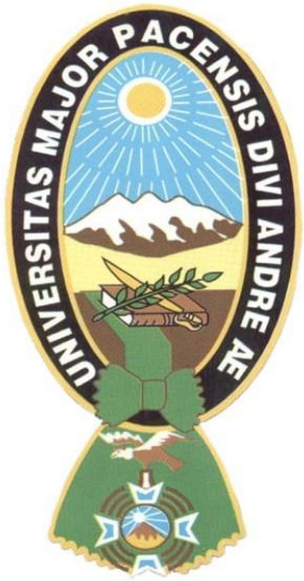
UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA



PROYECTO DE GRADO

"GESTIÓN DE HISTORIAS CLÍNICAS Y CUADROS ESTADÍSTICOS APLICANDO AGENTES INTELIGENTES
CASO: CENTRO DE SALUD ASISTENCIA PÚBLICA"

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERIA DE SISTEMAS INFORMÁTICOS

Postulante : Vanesa Paola Cárdenas Capari

Tutor : Lic. Germán Huanca Ticona

Revisor : Lic. Marcelo Aruquipa Chambi

La Paz - Bolivia
2011

DEDICATORIA

A mi mami, quien me brindó todo el amor, comprensión y apoyo constante, por ser la mejor mama del mundo, mi mejor amiga.

A mi papi, quien me dedico toda la paciencia que necesite para culminar mis metas.

A mis hermanas por todo el apoyo y aliento.

A mis abuelos quienes con todas sus enseñanzas lograron afianzar mis valores humanos.

A toda mi familia, tíos y primos que me acompañaron en todo momento de mi vida.

Son muchas las personas especiales a las que me gustaría agradecer su amistad, apoyo, ánimo y compañía en las diferentes etapas de mi vida. Algunas están conmigo y otras en mis recuerdos y en el corazón. Sin importar en donde estén quiero darles las gracias por formar parte de mí y por todo lo que me brindaron.

Vanesa Paola Cárdenas Capari

AGRADECIMIENTOS

A Dios que me dio la oportunidad de vivir y me regalo una familia maravillosa.

Al Licenciado Germán Huanca por guiar mis pasos en el desarrollo de este proyecto.

Al Licenciado Marcelo Aruquipa, por su ejemplo de profesionalidad que nunca he olvidado. Por el apoyo, supervisión y correcciones que me brindo a lo largo de este proyecto.

Al Director Dr. Felsi Muñoz, a la Trabajadora Social Licenciada Virginia y a la señora Julieta encargada de estadística, por la oportunidad y apoyo que me brindaron para la culminación de este proyecto.

A Wilfredo Nacho por el amor, apoyo, compañía y enseñanzas que me dedica día a día.

A mis amigos por la compañía y comprensión en todos los años de estudio que estuvimos juntos.

….y a todos aquellos que hicieron posible la confección y elaboración de este proyecto.

RESUMEN

El centro de salud Asistencia Pública atiende un gran número de pacientes, además de beneficiarios del Seguro Universal Materno Infantil (SUMI), el Seguro de Salud para el Adulto Mayor (SSPAM) y niños hasta los cinco años. Debido a la cantidad de atenciones que se realizan, se genera un gran volumen de información el cual se administra manualmente, el mismo dificulta el procesamiento de datos para la obtención de informes y reportes que se deben enviar al Servicio Nacional de Información en Salud (SNIS).

El presente proyecto se desarrolló con la metodología ágil Scrum y se implementó un sistema de gestión que cumple con los requerimientos del área de afiliación, asignación de consultas y el diagnóstico médico aplicando agentes inteligentes para el análisis de datos del paciente y el diagnóstico médico. Al implementar este proyecto los departamentos dedicados a la atención del paciente, realizan las mismas tareas de manera eficiente y con información actualizada.

ABSTRACT

The health center Public Assistance serves a large number of patients, in addition to beneficiaries of the Universal Insurance maternal-infant (SUMI), health insurance for the Elderly (SSPAM) and children up to five years. Due to the amount of attentions that made, generates a large volume of information which is administered manually, the same makes data processing for the collection of reports and reports to be sent to the National Information Service for Health (SNIS).

This project was developed with the Scrum agile methodology and implemented a management system that meets the membership requirements of the area, allocation of consultations and medical diagnostics using intelligent agents to analyze patient data and medical diagnostics. By implementing this project, the departments involved in patient care, perform the same tasks efficiently and with updates.

INDICE GENERAL

CAPITULO III

INDICE DE FIGURAS

INDICE DE TABLAS

CAPITULO I

CAPITULO I

1. INTRODUCCIÓN

1.1. PRESENTACIÓN

Los sistemas informáticos se han convertido en herramientas indispensables para automatizar y optimizar tareas que las personas realizan de forma repetitiva, permitiendo realizarlas de manera eficiente, además de disminuir el tiempo y costo que se invierte en ellas.

Por otro lado, la inteligencia artificial es una rama de la informática que es aprovechada en simulaciones de sistemas complejos. Una parte de ella son los agentes inteligentes, los mismos que son entidades de software que a partir de un repositorio de conocimiento realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa. Entre sus propiedades se tienen: autonomía, sociabilidad, capacidad de reacción, iniciativa benevolencia y racionalidad [Wooldridge y Jennings, 1995].

En el área de salud, los sistemas informáticos permiten la administración adecuada de los recursos, porque ayudan a la formación de elementos de juicio para la toma de decisiones luego de que se hayan aprovechado sus funciones de captura, almacenamiento, análisis y visualización de la información procesada.

Bolivia cuenta desde hace más de diez años con un Sistema Nacional de Información de Salud (SNIS), que proporciona información estadística sobre los servicios de salud y morbilidad que está bajo acción programática, así como de indicadores de vigilancia epidemiológica; el SNIS ha ido adecuándose en el tiempo a las exigencias emergentes de las políticas de salud del país.

El Sistema Público de Salud (SPS), tiene como finalidad alcanzar niveles de equidad, calidad y eficiencia en la provisión de servicios de salud, así como la solidaridad y universalidad en el acceso y la cobertura de la población. Estos son aspectos que constituyen un reto sobre todo en términos de alcanzar calidad técnica e

interpersonal, mejorar el desempeño y asegurar recursos financieros acordes a la señalada finalidad.

Es en este sentido que los centros de salud en Bolivia incorporan sistemas de información de uso exclusivo para gestionar sus recursos como ser: la atención, registro, evaluación y apertura de historias clínicas de los pacientes.

El Centro de Salud Asistencia Pública atiende a una gran cantidad de asegurados que resulta en la generación de un gran volumen de información. Entre los afiliados a este Centro de Salud podemos mencionar al sector público, mujeres en estado de gestación (SUMI), personas de la 3º edad (SSPAM), niños menores a cinco años y cualquier persona que así lo requiera.

El incremento de afiliación a la Asistencia Pública conlleva al aumento de historias clínicas, de esto surge la dificultad de organizarlos adecuadamente, además de que hasta el momento las tareas en la Asistencia Pública son realizadas en libros de registro y manualmente. Por tal motivo el tiempo que requieren para finalizar esta tarea es bastante. Al mismo tiempo surge el conflicto de existencia de duplicidad en historias clínicas y a la hora de seleccionar uno es muy difícil decidir cuál es el verdadero o correcto; que influye en el tiempo de espera en sala del paciente, a sí mismo en el tipo de información que se almacena para los informes posteriores que resultan no ser exactos, afectando esto al departamento de estadística.

El departamento de estadística debe contar con la información necesaria para realizar las estadísticas y conjuntamente a esto corresponde realizar los reportes en un formato detallado y preciso para la cual la información debe ser previamente elaborada y clasificada, para proporcionarlos al director de la Asistencia Pública y al Servicio Departamental de Salud, SEDES, para finalmente derivarse al SNIS.

Por tanto se ve la necesidad de desarrollar un sistema de "Gestión de Historias Clínicas y cuadros Estadísticos Aplicando Agentes Inteligentes" que permita la organización de procesos y a la toma de decisiones. Los módulos que se pretenden desarrollar, facilitarán la organización de la información de los afiliados llevando un registro depurado de historias clínicas con el apoyo de agentes inteligentes.

1.2. ANTECEDENTES

El centro de salud "Asistencia Pública" es una entidad del estado que se encuentra en funcionamiento hace 100 años, cuenta con un equipo de profesionales especialistas en salud y administración, con experiencia en áreas de su competencia, centrando sus actividades en distintas áreas como: el proceso de registro, apertura de historias y la atención de afecciones a pacientes, además de obtener reportes sobre la cantidad de personas que visitan el centro de salud.

Tiene como misión resolver la emergencia médico-quirúrgica, efectuar la derivación oportuna y correcta, dentro de un escudo de servicios en salud como: consulta médica de especialidades y de emergencias, servicio de imagenología, laboratorio, farmacia, servicio de ambulancia, evitando el riesgo de complicaciones irreparables del paciente, coadyuvando de manera efectiva a la salud de su población.

Durante los últimos años se realizaron proyectos de grado y tesis que gestionan historias clínicas y otras que se realizaron aplicando los agentes inteligentes como un apoyo al desarrollo de software, entre los cuales podemos mencionar: el Sistema de Seguimiento y Control de Historias Clínicas para el Hospital Juan XXIII, 1998, desarrollado por Elisa Arizaca y Edwin López el cual es un sistema muy extenso ya que cuenta con muchas especialidades y servicios; Sistema de Gestión Hospitalaria modulo de afiliación, admisión y consulta externa caso seguro social universitario Cochabamba, 2007, desarrollado por Marcelo Soria que realiza el registro de las historias clínicas, así como también un detalle de laboratorios y exámenes

complementarios realizados; Sistema de Administración de Historias Clínicas Clínica Sanjinés, 2009, desarrollado por Américo Machicao.

Entre los trabajos que aplican uso de agentes inteligentes podemos mencionar: Agente inteligente como herramienta de aprendizaje, 2002, desarrollado por Braulio Coareti; Arquitectura de agente Inteligente para la atención de clientes en el comercio electrónico, 2003, realizado por Diego Hinojosa; Sistemas de información utilizando agentes inteligentes para el área administrativa y social de la parroquia espíritu santo, 2005, realizado por Ricarda Chambi.

Podemos resaltar el trabajo de gestión de historias clínicas para centros hospitalarios, también el uso de agentes inteligentes para la toma de decisiones en diferentes sistemas y procesos. La aplicación de agentes inteligentes a la gestión de historias clínicas es un trabajo diferente a los ya existentes, es por eso que el presente proyecto alcanzara a destacarse.

1.3. DEFINICION DEL PROBLEMA

El problema de la Asistencia Pública radica principalmente en la gran cantidad de historias clínicas con los que cuenta. Al momento de realizar la apertura de historias se debe verificar que el paciente no esté registrado previamente y constatar que no exista duplicidad de historias clínicas. Al encontrarse con conflictos en estos procesos resultan afectando el tiempo de espera del paciente para ser atendido y más adelante a la toma correcta de decisiones.

Como los principales problemas que presenta la Asistencia Pública podemos ver:

- La redundancia de información, se tiene un gran tamaño de información que en muchos casos resultan ser repetidas.
- Pérdida de tiempo en la realización de apertura de historias clínicas, antes de realizar este registro se debe verificar que no se cuenta con uno y realizar esta búsqueda y verificación manualmente.

- Información desactualizada y no disponible en el momento preciso, debido a que la información no está bien clasificada es complicado obtener los datos al momento.
- Calidad de atención retardada, debido a los problemas mencionados anteriormente el número promedio de consultas que se debe realizar por día y por medico no se cumplen.
- Información no fidedigna de indicadores de cobertura e indicadores de salud. Al realizar la evaluación de los datos para la obtención de estos informes estadísticos, no es posible contar con información y datos reales al momento para realizar el formulario del SNIS.

1.4. PLANTEAMIENTO DEL PROBLEMA

¿El sistema de gestión de historias clínicas aplicando agentes inteligentes optimizara la obtención de información para la mejor atención a los pacientes y a la toma de decisiones?

1.5. JUSTIFICACIÓN

1.5.1. Justificación Económica

El presente proyecto se justifica económicamente, en el sentido de que el Municipio de La Paz del cual depende la Asistencia Pública no incurrirá en gastos de desarrollo e implementación. Además de que cuentan con equipos computacionales que hará posible la implementación del sistema sin costo alguno. Asimismo el ingreso económico será mayor:

- Al alcanzar el número de pacientes promedio por día.
- Al reducir el esfuerzo que se invierte en la realización de reportes.

Por tanto el proyecto es justificable económicamente porque existen los medios para el desarrollo e implementación del sistema.

1.5.2. Justificación Social

Se brindara atención eficiente a los pacientes, el tiempo de espera para la consulta disminuirá, la organización de los procesos ayudara a la mejor asignación de historias clínicas, la información para la Dirección y para el envió de los formularios al SNIS será oportuna y de datos exactos. Lo que permitirá mejorar la atención a los pacientes, por esta razón se justifica socialmente.

1.5.3. Justificación Técnica

El centro de salud "Asistencia Pública" cuenta con el hardware y software necesario para el desarrollo e implementación del sistema de gestión, como ser: Servidores de aplicaciones y de bases de datos, equipos clientes que permitan el ingreso de datos de pacientes y sistemas operativos Windows con licencia.

Para evitar gastos innecesarios en cuanto a la herramienta de desarrollo, se hará uso de un lenguaje de programación y un gestor de base de datos con licencia libre; de esta manera, el proyecto se justifica técnicamente siendo factible su implementación.

1.6. OBJETIVOS

1.6.1. Objetivo general

Desarrollar un sistema de gestión de historias clínicas aplicando agentes inteligentes que optimizara la obtención de información para la mejor atención a los pacientes y que colabore a la toma de decisiones a corto y mediano plazo.

1.6.2. Objetivos Específicos

- Implementar agentes inteligentes que eviten la duplicidad en las historias clínicas y que colaboren al generar un número único de identificación del paciente.
- Modelar una base de datos que optimice el tiempo de búsqueda de información de los pacientes, de esta manera acelerar la apertura, gestión y de consulta de las historias clínicas.

- Implementar interfaces que permitan obtener reportes e informes de forma rápida y con información fidedigna.
- Establecer políticas de recepción para mejorar la calidad y el tiempo que se brinda a cada paciente.
- Optimizar el manejo de la información para proporcionar datos inequívocos y colaborar con el departamento de estadística en la elaboración de informes presentados al Servicio Departamental de Salud (SEDES).

1.7. ALCANCES Y APORTES

Una vez identificado el problema, es necesario delimitar los alcances del proyecto. Esto significa determinar y especificar el ámbito de acción del proyecto:

- El trabajo implica el desarrollo de un sistema de gestión que permita tener una base de datos centralizada en el centro de salud "Asistencia Pública" en la ciudad de La Paz, registrando los datos generales de los pacientes.
- Los módulos a implementar serán: Gestión de historias clínicas y diagnóstico médico, despliegue de información estadística de los pacientes.
- Se aplicará agentes inteligentes para la toma de decisiones en el registro y depuración de historias clínicas.
- El sistema se implementará en un entorno de escritorio, debido a que la institución no cuenta con Proveedor de Servicios de Internet (IPS).
- El sistema manejará una arquitectura cliente-servidor, la misma requerirá la configuración de equipos servidores y clientes dentro la institución.

El aporte propuesto por el presente proyecto, es que el sistema realizará la evaluación de los datos para la obtención de informes estadísticos sobre los servicios de salud y morbilidad, así como de indicadores de vigilancia epidemiológica.

El aporte a la carrera de informática es el uso de los agentes inteligentes aplicado a la gestión de historias clínicas como apoyo a la toma de decisiones.

1.8. METODOLOGÍAS EMPLEADAS

El desarrollo del presente proyecto considera el uso de las siguientes técnicas, métodos, metodologías y herramientas.

a. Método científico. Mediante las técnicas de observación, identificación de variables que intervienen en el planteamiento de problemas y soluciones, se realizará el trabajo de investigación científico enmarcado dentro las normas planteadas.
b. Ingeniería del Software orientada a objetos. Se utiliza como método para el desarrollo del producto software con calidad. Ésta plantea construir sistemas fiables, utilizables y adaptables basados en un enfoque disciplinado para el desarrollo de estos productos.
c. Programación Orientada a Objetos (POO). La programación orientada a objetos es la base para la construcción de aplicaciones con un grado de complejidad bastante alto, es por eso que se hará uso de esta técnica para la construcción del sistema.
d. Lenguaje Unificado de Modelado (UML). Este lenguaje permitirá modelar la aplicación de tal forma que se puedan representar los más mínimos detalles y se evite incurrir en errores de diseño que a la larga provocan una serie de retrasos en la construcción del proyecto.
e. SCRUM. Esta metodología permitirá la construcción del sistema de gestión, haciendo hincapié en la definición correcta de las "Tareas" y los "Tiempos" de implementación en base a una buena Ingeniería de requerimientos y el uso de todas sus bondades para la construcción del presente proyecto.

CAPITULO II

2. MARCO TEÓRICO

2.1. SISTEMA DE INFORMACIÓN

Un sistema de información puede definirse técnicamente como "un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir la información para apoyar a la toma de decisiones y el control en una institución". Un sistema de información es un conjunto de personas, datos y procedimientos que funcionan en conjunto, el énfasis en sistemas significa que los variados componentes buscan un objetivo común para apoyar las actividades de la organización. Estas incluyen las operaciones diarias de las empresas, la comunicación de los datos e informes, la administración de las actividades y la toma de decisiones. [Senn1990]

2.2. INGENIERÍA DEL SOFTWARE

Hoy en día, las grandes corporaciones conocen la importancia estratégica de sus políticas en relación con las tecnologías de información y el papel fundamental del software como parte de esas tecnologías [Leite1997].

Según Nauer y Randall el término Ingeniería de Software significa "el establecimiento y uso de principios de ingeniería para obtener en forma económica, software confiable y que trabaje eficientemente en máquinas reales". [Nauer1969]

Guezzi define la Ingeniería de Software como "el campo de la ciencia de la computación que trata la construcción de sistemas de software que son tan grandes o tan complejos que tienen que ser construidos por un equipo o equipos de ingenieros". [Guezzi1991]

Según Alan Davis Ingeniería de Software es "la aplicación de principios científicos a: 1) la transformación ordenada de un problema en una solución de software, y 2) el

mantenimiento subsiguiente de ese software hasta el final de su vida útil". [Thayer1997]

Pressman establece los objetivos de la ingeniería de software: "Los objetivos claves de la Ingeniería de Software son definir, crear y aplicar 1) una metodología bien definida dirigida a un ciclo de vida de planeamiento, desarrollo, y mantenimiento; 2) un conjunto establecido de componentes de software que documenta cada paso en el ciclo de vida y muestra un seguimiento paso a paso, y 3) un conjunto de hitos predecibles que pueden ser revisados a intervalos regulares a través del ciclo de vida del software".

Ha habido grandes avances en la tecnología de Ingeniería de Software en todas las áreas: análisis de requisitos, estrategias de implementación, modelos de costos, etc. Sin embargo, la Ingeniería de Software aún está por debajo de las necesidades de calidad demandadas por sistemas cada vez más complejos. En este contexto, existen considerables esfuerzos de investigación y desarrollo con el objetivo de perfeccionar el proceso de producción de software, tanto a través de estudios teóricos, como de estudios aplicados.

A pesar de la creciente participación del software en el mundo actual, y de los avances producidos, el software como producto aún utiliza procesos de producción bastante insatisfactorios. Son varios los casos en que errores en el software han traído problemas no sólo financieros, sino también en vidas humanas. Es función de la Ingeniería de Software evitar que estos errores ocurran, produciendo productos de software más robustos y proveyendo procesos de producción más confiables.

El punto de partida del proceso de producción de software es el momento en que se define lo que se quiere. Por lo tanto, el desarrollo del software sólo puede ser iniciado cuando se tiene bien establecido lo que se quiere producir. Cuando se trata de sistemas complejos, esta definición de lo que se quiere no es leve. Por ello, muchos productos de software no se comportan como sería deseable. Hacer una definición

que cubra todas las necesidades de un sistema complejo es una tarea difícil. Y es más difícil aún si no se cuenta con métodos, técnicas y herramientas adecuadas.

En este ámbito se sitúa la Ingeniería de Requisitos, como un área de investigación que procura abordar un punto fundamental en el proceso de producción, que es la definición de lo que se quiere producir. Contiene a la Ingeniería de Requisitos, como subárea de la Ingeniería de Software, proponer métodos, técnicas y herramientas que faciliten el trabajo de definición de lo que se quiere de un software. Por lo tanto, la Ingeniería de Requisitos tiene una interacción muy fuerte con aquéllos que demandan un producto de software, sean éstos el mercado o los clientes de una aplicación que será especialmente construida.

2.3. METODOLOGÍAS ÁGILES

Las metodologías ágiles aparecen como alternativa atractiva para adaptarse a este entorno. Son apropiadas cuando los requisitos son emergentes y cambian rápidamente. De este modo, presentan diversas ventajas en el contexto actual [Nerur2005]:

- ✓ Capacidad de respuesta a cambios a lo largo del desarrollo ya que no los perciben como un impedimento sino como una oportunidad para mejorar el sistema e incrementar la satisfacción del cliente, considerando la gestión de cambios como un aspecto característico del propio proceso de desarrollo software.
- ✓ Entrega continúa y en plazos breves de software funcional lo que permite al cliente verificar in situ[1] el desarrollo del proyecto, ir disfrutando de la funcionalidad del producto progresivamente y comprobando si satisface sus necesidades, mejorando de esta forma su satisfacción. Además, el desarrollo en ciclos de corta duración favorece que los riesgos y dificultades se repartan a lo largo del desarrollo del producto, principalmente al comienzo del mismo y permite ir aprendiendo de estos riesgos y dificultades (figura 1).

[1] In situ: En el lugar

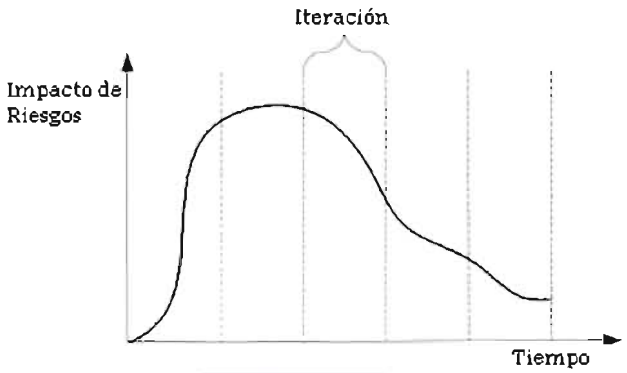


Figura 2.1. Distribución del riesgo en un desarrollo ágil

Fuente: [Rodriguez2008]

- ✓ Trabajo conjunto entre el cliente y el equipo de desarrollo con una comunicación directa que pretende mitigar malentendidos, que constituyen una de las principales fuentes de errores en productos software, y exceso de documentación improductiva.
- ✓ Importancia de la simplicidad, eliminando el trabajo innecesario que no aporta valor al negocio.
- ✓ Atención continúa a la excelencia técnica y al buen diseño para mantener una alta calidad de los productos.
- ✓ Mejora continua de los procesos y el equipo de desarrollo, entendiendo que el éxito, tal y como se indica en la figura 2, depende de tres factores: éxito técnico, éxito personal y éxito organizacional.

Figura 2.2. Tipos de éxito

Fuente: [Rodriguez2008]

2.3.1. Modelo XP

De la mano de Kent Beck, XP ha conformado un extenso grupo de seguidores en todo el mundo, disparando una gran cantidad de libros a los que dio comienzo el mismo Beck en [Beck, 2000]. Inclusive Addison-Wesley ha creado una serie de libros denominada The XP Series. Fue la misma gente del proyecto C3 la que produjo también otro de los libros importantes de XP [Jeffries, 2001] en el que se bajaban los conceptos de Beck a la puesta en práctica en un proyecto.

La imagen mental de Beck al crear XP [Beck, 2000] era la de perillas en un tablero de control. Cada perilla representaba una práctica que de su experiencia sabía que trabajaba bien. Entonces, Beck decidió girar todas las perillas al máximo para ver que ocurría. Así fue como dio inicio a XP. [Schenone2004]

Los principios de XP citados verbatim de Beck:

- ✓ El juego de Planeamiento: Rápidamente determinar el alcance del próximo release mediante la combinación de prioridades del negocio y estimaciones técnicas. A medida que la realidad va cambiando el plan, actualizar el mismo.
- ✓ Pequeños Releases: Poner un sistema simple en producción rápidamente, luego liberar nuevas versiones en ciclos muy cortos.
- ✓ Metáfora: Guiar todo el desarrollo con una historia simple y compartida de cómo funciona todo el sistema.
- ✓ Diseño Simple: El sistema deberá ser diseñado tan simple como sea posible en cada momento. Complejidad extra es removida apenas es descubierta.
- ✓ Testing: Los programadores continuamente escriben pruebas unitarias, las cuales deben correr sin problemas para que el desarrollo continúe. Los clientes escriben pruebas demostrando que las funcionalidades están terminadas.
- ✓ Refactoring: Los programadores reestructuran el sistema sin cambiar su comportamiento para remover duplicación, mejorar la comunicación, simplificar, o añadir flexibilidad.
- ✓ Programación de a Pares: Todo el código de producción es escrito por dos programadores en una máquina.

- ✓ Propiedad Colectiva del Código: Cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento.
- ✓ Integración Continua: Integrar y hacer builds del sistema varias veces por día, cada vez que una tarea se completa.
- ✓ Semana de 40-horas: Trabajar no más de 40 horas semanales como regla. Nunca trabajar horas extras durante dos semanas consecutivas.
- ✓ Cliente en el lugar de Desarrollo: Incluir un cliente real en el equipo, disponible de forma full-time para responder preguntas.
- ✓ Estándares de Codificación: Los programadores escriben todo el código de acuerdo con reglas que enfatizan la comunicación a través del mismo. [Schenone2004]

Como se observan, muchas de las prácticas propuestas contribuyen a maximizar la comunicación entre las personas, permitiendo de esa forma una mayor transferencia de conocimiento entre los desarrolladores y con el cliente, quien también es parte del equipo. Esto es logrado en la práctica gracias a la disposición física del lugar de trabajo.

La idea es reunir a todas las personas en una misma oficina manteniendo una distribución denominada "cavernas y común". En la misma se observan escritorios dispuestos en el centro con varias computadoras, cada una con dos sillas dispuestas de forma de permitir la programación de a pares. En las paredes se observan pequeños boxes o escritorios con sillas, los cuales pueden ser usados por los programadores en forma privada, para realizar llamados, consultar mail, o simplemente descansar la mente. Consecuentemente se logra el objetivo mencionado de maximizar la comunicación y la transferencia de información en el área común, mientras que se mantiene la individualidad de las personas en las mencionadas cavernas.

Asimismo, XP impone un alto nivel de disciplina entre los programadores. El mismo permite mantener un mínimo nivel de documentación, lo cual a su vez se traduce en una gran velocidad en el desarrollo. Sin embargo, una desventaja que deviene de esta falta de documentación es la incapacidad de persistir la arquitectura y demás

cuestiones de análisis, diseño e implementación, aún después de que el proyecto haya concluido. [Schenone2004]

El énfasis que pone en XP en las personas se manifiesta en las diversas prácticas que indican que se deben dar más responsabilidades a los programadores para que estimen su trabajo, puedan entender el diseño de todo el código producido, y mantengan una metáfora mediante la cual se nombra las clases y métodos de forma consistente. La práctica denominada Semana de 40 horas indica la necesidad de mantener un horario fijo, sin horas extras ya que esto conlleva al desgaste del equipo y a la posible deserción de sus miembros. Beck afirma que como máximo se podría llegar a trabajar durante una semana con horas extras, pero si pasando ese tiempo las cosas no han mejorado entonces se deberá hacer un análisis de las estimaciones de cada iteración para que estén acordes a la capacidad de desarrollo del equipo.

Si bien XP es la metodología ágil de más renombre en la actualidad, se diferencia de las demás metodologías que forman este grupo en un aspecto en particular: el alto nivel de disciplina de las personas que participan en el proyecto. [Schenone2004]

2.3.2. Modelo Cristal Clear

Alistair Cockburn es el propulsor detrás de la serie de metodologías Crystal. Las mismas presentan un enfoque ágil, con gran énfasis en la comunicación, y con cierta tolerancia que la hace ideal en los casos en que sea inaplicable la disciplina requerida por XP. Crystal "Clear" es la encarnación más ágil de la serie y de la que más documentación se dispone. La misma se define con mucho énfasis en la comunicación, y de forma muy liviana en relación a los entregables [Cockburn, 2001b]. Crystal maneja iteraciones cortas con feedback frecuente por parte de los usuarios/clientes, minimizando de esta forma la necesidad de productos intermedios. Otra de las cuestiones planteadas es la necesidad de disponer de un usuario real aunque sea de medio tiempo para realizar validaciones sobre la Interface del usuario y para participar en la definición de los requerimientos funcionales y no funcionales del software.

Una cuestión interesante que surge del análisis de la serie Crystal es el pragmatismo con que se personaliza el proceso. Las personas involucradas escogen aquellos

principios que les resultan efectivos y mediante la aplicación de la metodología en diversos proyectos agregan o remueven principios en base al consenso grupal del equipo de desarrollo. [Schenone2004]

2.3.3. Modelo Scrum

2.3.3.1. El origen

Scrum es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80.

Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.

Jeff Sutherland aplicó el modelo Scrum al desarrollo de software en 1993 en Easel Corporation (Empresa que en los macro-juegos de compras y fusiones se integraría en VMARK, luego en Informix y finalmente en Ascential Software Corporation). En 1996 lo presentó junto con Ken Schwaber como proceso formal, también para gestión del desarrollo de software en OOPSLA 96. Más tarde, en 2001 serían dos de lo promulgadores del Manifiesto_ágil. En el desarrollo de software scrum está considerado como modelo ágil por la Agile Alliance.

2.3.3.2. Introducción al modelo

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

Scrum es una metodología ágil, y como tal:

Es un modo de desarrollo de carácter adaptable más que predictivo.

Orientado a las personas más que a los procesos.

Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

Se comienza con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve (normalmente de 30 días).

Cada uno de estos periodos de desarrollo es una iteración que finaliza con la producción de un incremento operativo del producto.

Estas iteraciones son la base del desarrollo ágil, y Scrum gestiona su evolución a través de reuniones breves diarias en las que todo el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente.

2.3.3.3. Control de la evolución del proyecto

Scrum controla de forma empírica y adaptable la evolución del proyecto, empleando las siguientes prácticas de la gestión ágil:

- **Revisión de las Iteraciones**

Al finalizar cada iteración (normalmente 30 días) se lleva a cabo una revisión con todas las personas implicadas en el proyecto. Este es el periodo máximo que se tarda en reconducir una desviación en el proyecto o en las circunstancias del producto.

- **Desarrollo incremental**

Durante el proyecto, las personas implicadas no trabajan con diseños o abstracciones.

El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.

- **Desarrollo evolutivo**

Los modelos de gestión ágil se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos.

Intentar predecir en las fases iniciales cómo será el producto final, y sobre dicha predicción desarrollar el diseño y la arquitectura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces.

Para qué predecir los estados finales de la arquitectura o del diseño si van a estar cambiando. En Scrum se toma a la inestabilidad como una premisa, y se adoptan

técnicas de trabajo para permitir esa evolución sin degradar la calidad de la arquitectura que se irá generando durante el desarrollo.

El desarrollo Scrum va generando el diseño y la arquitectura final de forma evolutiva durante todo el proyecto. No los considera como productos que deban realizarse en la primera "fase" del proyecto. (El desarrollo ágil no es un desarrollo en fases).

- **Auto-organización**

Durante el desarrollo de un proyecto son muchos los factores impredecibles que surgen en todas las áreas y niveles. La gestión predictiva confía la responsabilidad de su resolución al gestor de proyectos.

En Scrum los equipos son auto-organizados (no auto-dirigidos), con margen de decisión suficiente para tomar las decisiones que consideren oportunas.

- **Colaboración**

Las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo. Ésta es necesaria, porque para que funcione la auto-organización como un control eficaz cada miembro del equipo debe colaborar de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.

2.3.3.4. Visión general del proceso

Scrum denomina "sprint" a cada iteración de desarrollo y recomienda realizarlas con duraciones de 30 días.

El sprint es por tanto el núcleo central que proporciona la base de desarrollo iterativo e incremental.

Los elementos que conforman el desarrollo Scrum son:

- **Las reuniones**

Planificación de sprint: Jornada de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el trabajo y los objetivos que se deben cumplir en esa iteración.

Reunión diaria: Breve revisión del equipo del trabajo realizado hasta la fecha y la previsión para el día siguiente.

Revisión de sprint: Análisis y revisión del incremento generado.

- **Los elementos**

Pila del producto: lista de requisitos de usuario que se origina con la visión inicial del producto y va creciendo y evolucionando durante el desarrollo.

Pila del sprint: Lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto.

Incremento: Resultado de cada sprint

- **Los roles**

Scrum clasifica a todas las personas que intervienen o tienen interés en el desarrollo del proyecto en: propietario del producto, equipo, gestor de Scrum (también Scrum Manager o Scrum Master) y "otros interesados".

2.3.3.5. Valores

Scrum es una "carrocería" para dar forma a los principios ágiles. Es una ayuda para organizar a las personas y el flujo de trabajo; como lo pueden ser otras propuestas de formas de trabajo ágil: Cristal, DSDM, etc.

La carrocería sin motor, sin los valores que dan sentido al desarrollo ágil, no funciona.

Delegación de atribuciones (empowerment) al equipo para que pueda auto-organizarse y tomar las decisiones sobre el desarrollo.

Respeto entre las personas. Los miembros del equipo deben confiar entre ellos y respetar sus conocimientos y capacidades.

Responsabilidad y auto-disciplina (no disciplina impuesta).

Trabajo centrado en el desarrollo de lo comprometido

Información, transparencia y visibilidad del desarrollo del proyecto.

2.4. UML COMO HERRAMIENTA DE MODELADO

2.4.1. Definición Lenguaje de Modelado Unificado (UML)

El Lenguaje de Modelado Unificado (UML) es un lenguaje estándar para la escritura de proyectos de software. El UML puede ser usado para visualizar, especificar, construir y documentar los componentes de un sistema de software extenso.

Este es un lenguaje muy expresivo, que abarca todos los panoramas necesarios para desarrollar y estructurar tales sistemas UML es un proceso independiente que óptimamente debe ser usado en un proceso que es un manejador de casos de uso, con arquitectura central, iterativa e incremental.

2.4.2. UML como lenguaje

Un lenguaje proporciona un vocabulario y reglas para permitir una comunicación. Este caso, este lenguaje se centra en la representación grafica de un sistema. Este lenguaje nos indica cómo crear y leer los modelos, pero no dice como crearlos. Este último es el objetivo de las metodologías de desarrollo.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- **Visualizar:** UML permite expresar de una forma grafica un sistema de forma que otro lo puede entender.
- **Especificar:** UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema de desarrollo que pueden servir para su futura revisión.

Aunque UML está pensado en modelar sistemas complejos con una gran cantidad de software, el lenguaje es lo suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo (workflow) en una empresa, diseño de la estructura de una organización y por supuesto, en el diseño de la estructura de una organización y por supuesto, en el diseño de hardware.

Un modelo UML está compuesto por tres clases de bloques de construcción:

- **Elementos:** Los elementos son abstractos de cosas reales o ficticias (objetos, acciones, etc.).
- **Relaciones:** Relacionan los elementos entre sí.
- **Diagramas:** Son colecciones de elementos con sus relaciones.

2.5. ARQUITECTURA DE SOFTWARE

2.5.1. Arquitectura de tres niveles

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos.

Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actual se suelen usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas). [Scott1999]

2.5.1.1. Capas y niveles

- **Capa de presentación:** es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.
- **Capa de negocio:** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso.

Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

- **Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si, por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de negocio, y otra serie de ordenadores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

- Presentación.
- Lógica de Negocio.
- Datos.

En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:
Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y un nivel.
Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos ordenadores (presentación+lógica por un lado; lógica+datos por el otro lado). Se dice que la arquitectura de la solución es de tres capas y dos niveles. [Scott1999]

2.6. MÉTRICAS DE CALIDAD

Uno de los problemas que se afrontan en la computación es la calidad del software. Desde la década de los setenta, este tema ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los cuales han realizado gran cantidad de investigaciones al respecto con dos objetivos fundamentales:

- ¿Cómo obtener un software de calidad?
- ¿Cómo evaluar la calidad del software?

Ambas preguntas llevan a amplias respuestas, pero están estrechamente ligadas con la definición de métricas de calidad de software.
La calidad del software, en general persigue tres objetivos fundamentales: ayudarnos a entender que ocurre durante el desarrollo y el mantenimiento, permitirnos controlar que es lo ocurre en nuestros proyectos y poder mejorar nuestros procesos y productos, es por eso que se definen las métricas.
El concepto de métrica es el término que describe variados casos de medición. Siendo una métrica una medida estadística no cuantitativa que se aplica a todos los aspectos de calidad de software.

2.6.1. Confiabilidad

La confiabilidad puede definirse como la probabilidad de que un sistema este aun funcionando favorablemente, bajo un intervalo de tiempo t.

De ello podemos decir que la confiabilidad R(t) de un componente en un determinado medio durante un periodo t se define como la probabilidad de que su tiempo para fallar excede en t es decir:

$$\mathbf{R(t) = P[T > t] = 1 - F(t)}$$

Donde:

R(t)= función de confiabilidad de un componente en un tiempo t.

P(t)=probabilidad de falla de un componente o subsistema en el tiempo t.

T=tiempo para fallar o la duración del tiempo de trabajo sin falla del componente.

Tomando en cuenta que el tiempo T para fallar es una variable aleatoria exponencial, se define como:

$$\mathbf{R(t) = 1 - [1 - e^{\lambda t}]}$$

$$\mathbf{R(t) = e^{\lambda t}}$$

Donde:

λ: tasa constante de fallo (λ=Nro de fallas de acceso/Nro total de acceso al sistema).

T: periodo de operación de tiempo.

Para hallar la confiabilidad total del sistema, se toma en cuenta dos situaciones.

Teorema 1: Si n componentes independientes están conectados en serie y el i-ésimo componente tiene la confiabilidad R(t) la confiabilidad del sistema competo es:

$$\mathbf{R(t) = R_1(t) * R_2(t) * ... R_{n-1}(t) * R_n(t)}$$

Teorema 2: Si n componentes independientes que actúan en paralelo y si el i-ésimo componente tiene la confiabilidad R(t) entonces la confiabilidad del sistema R(t) está dada por:

$$\mathbf{R(t) = 1 - (1 - R_1(t) * ... * \left(1 - R_n(t)\right) = 1 - (1 - e) * ... * (1 - e) = 1 - (1 - e)}$$

2.6.2. Métrica basadas en funciones – Punto Función

Las métricas orientadas a la función fueron propuestas por primera vez por Albretch quien sugirió la medida llamada Punto Función.

La métrica de punto de función (PF) puede usarse de manera efectiva como medio para medir la funcionalidad que entra a un sistema. Al usar datos históricos, la métrica PF puede entonces usarse para 1) estimar el costo o esfuerzo requerido para

diseñar, codificar y probar el software; 2) predecir el numero de errores que se encontraran durante las pruebas y; 3) prever el número de componentes y/o de líneas de fuente proyectadas en el sistema implementado.
Los puntos de función se derivan usando una relación empírica basada en medidas contables (directas) del dominio de información del software y en valoraciones cualitativas de la complejidad del software.

- **Número de entradas externas (EE)**

Se origina de un usuario o se transmite desde otra aplicación, y proporciona distintos orientados a aplicación o información de control. Con frecuencia las entradas se usan para actualizar archivos lógicos internos (ALI). Las entradas deben distinguirse de las consultas, que se cuentan por separado.

- **Número de salidas externas (SE)**

Cada salida externa es datos derivados dentro de la aplicación que ofrecen información al usuario. En este contexto, salida externa se refiere a reportes, pantallas, mensajes de error, etc. Los ítems de datos individuales dentro de un reporte no se cuentan por separado.

- **Número de consultas externas (CE)**

Una consulta externa se define como una entrada en línea que da como resultado la generación de alguna respuesta de software inmediata en la forma de una salida en línea (con frecuencia recuperada de un ALI).

- **Número de archivos lógicos internos (ALI)**

Cada archivo lógico interno es un agrupamiento lógico de datos que reside dentro de la frontera de la aplicación y se mantiene mediante entradas externas.

- **Número de archivos de interfaz externos (AIE)**

Cada archivo de interfaz externo es un agrupamiento lógico de datos que reside fuera de la aplicación, pero que proporciona información que puede usar la aplicación.

Una vez recolectados dichos datos, la tabla 2.1 se completa y un valor complejidad se asocia con cada conteo. Las organizaciones que usan métodos de punto de

función desarrollan criterios para determinar si una entrada en particular es simple, promedio o compleja. No obstante, la determinación de la complejidad es un tanto subjetiva.

Valor de dominio de información		Factor ponderado			
	Conteo	Simple	Promedio	Complejo	
Entradas externas (EE)		x 3	4	6	=
Salidas externas(SE)		x 4	5	7	=
Consultas externas(CE)		x 3	4	6	=
Archivos lógicos internos(ALI)		x 7	10	15	=
Archivos de interfaces externos(AIE)		x 5	7	10	=
Conteo total					

Tabla 2.1. Punto Función
Fuente: [Pressman2010]

Para calcular los puntos función (PF), se utiliza la siguiente relación:

$$\mathrm{PF} = \text{conteo total} * [0.65 + 0.01 * \sum(\mathrm{F_i})]$$

Donde conteo total es la suma de todas las entradas PF obtenidas de la Tabla 1.
Los $\mathrm{F_i}$ $(\mathbf{i} = 1 \text{ a } 14)$ son factores de ajuste de valor (FAV), con base en respuestas de una serie de preguntas de usuario una escala con rangos de 0 hasta 5.

La medida de punto función se diseño originalmente para aplicarse a aplicaciones de sistemas de información de gestión. Para acomodar estas aplicaciones, se enfatizo la dimensión de datos para la exclusión de dimensiones funcionales y de comportamiento. Por esta razón, la medida de punto función era adecuada para muchos sistemas así que para remediar esto se ha propuesto un número de extensiones a la métrica de punto función básica.

2.7. TEORIA DE COSTOS

2.7.1. COCOMO

El modelo constructivo de costes (o COCOMO, por su acrónimo del inglés Constructive Cost Model) es un modelo de estimación de costes de software que incluye tres sub modelos, donde cada uno ofrece un nivel de detalle y aproximación cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado. [Boehm1981]

2.8. AGENTES INTELIGENTES

La palabra "agente" se refiere a todo ente que posee la habilidad, capacidad y autorización para actuar en nombre de otro. A diario, los agentes humanos asisten a las personas en tareas que requieren recursos especializados o conocimiento específico en un dominio. Por ejemplo, una secretaria atiende y resuelve situaciones en nombre de su jefe: administra la agenda, coordina las reuniones, recibe a los visitantes [Tolosa1999].

Los agentes cumplen con los requerimientos para los cuales fueron entrenados. El usuario "delega" en el agente una o varias tareas que debe llevar a cabo quedando a la espera de los resultados. Dichas tareas, son a menudo fácil de especificar pero – en algunos casos – complejas de realizar. Hay que tomar en cuenta que los investigadores en el campo de los agentes computacionales han dado varias definiciones al término, cada uno desde su óptica particular, fundamentada básicamente en la línea de investigación en la cual trabajan (Inteligencia Artificial, Ingeniería de Software, Sistemas Autónomos).

Definición: Un agente es una entidad computacional que percibe y actúa autónomamente en un ambiente abierto y dinámico, aprendiendo y cooperando con otros agentes (el usuario mismo) para brindar un beneficio a su usuario. [Ramírez2004]

2.8.1. Propiedades de los agentes

En base a la definición anterior es posible extraer algunas características que deben tener los agentes: deben ser parte de un ambiente dinámico, deben poder censar su entorno y actuar sobre él, y deben responder según los objetivos para los cuales fueron diseñados.

Deben – entonces – poseer una serie de atributos o propiedades que lo definen como agente [Wooldridge1996]:

- **Autonomía:** Capacidad de actuar sin la intervención directa de una persona o de otro agente. Un agente debe poder controlar sus propias acciones y estado interno. Una vez que el usuario activa el agente indicando algún objetivo de alto nivel, éste actúa independientemente, seleccionando estrategias y monitoreando el progreso en busca de la meta. Si falla con una estrategia, usará otra, pero sin intervención humana o con la mínima indispensable.
- **Aprendizaje:** Un agente aprende en la medida que su conocimiento es completo y no presenta inconsistencias. Este aprendizaje es de parte del usuario, de su ambiente y de los otros agentes con que convive.
- **Colaboración:** Un agente debe colaborar. Debe tener habilidad para interactuar con otros agentes o incluso con alguna persona (el usuario), para solicitar información o bien para exponer los resultados obtenidos de la ejecución de las tareas agendadas. La naturaleza de la comunicación dependerá del tipo de agente con quien se comunique (humanos o no), en ambos casos se deberá establecer un protocolo común de intercambio de información entre ambas partes.

2.8.2. Taxonomías de agentes

Los agentes pueden clasificarse de varias maneras, teniendo en cuenta algunas de las propiedades que poseen o bien haciendo hincapié en alguna en particular. De esta manera puede armarse un árbol taxonómico que abarque todas las combinaciones de propiedades y tareas que se quieran. Nosotros consideramos una clasificación de acuerdo a líneas de investigación y desarrollo de agentes, donde la

prioridad es la función u objetivo principal del agente. De acuerdo a lo anterior consideramos la siguiente clasificación [Berney1996]:

- Agentes de interfaz
- Agentes colaborativos
- Agentes móviles
- Agentes de recuperación de información

2.8.2.1. Agentes de interfaz

Un agente de interfaz es un software casi-inteligente que asiste a un usuario cuando interactúa con una o más aplicaciones. Son asistentes personales que reducen el trabajo por la sobrecarga de información, por ejemplo, el filtrado de los mensajes de correo electrónico o la recuperación de archivos de Internet.

Esta categoría de agentes apoyan y proveen asistencia a su usuario. El agente observa y monitorea las acciones que toma el usuario en la interfaz, aprende nuevos atajos, y sugiere mejores formas de hacer las tareas. La idea es que el agente pueda adaptarse a las preferencias y hábitos de sus usuarios.

Enfatizan la autonomía y el aprendizaje para llevar a cabo tareas para sus dueños y trabajan en el mismo ambiente que estos.

A su vez, de los agentes de interfaz pueden encontrarse subdivisiones debido a diferentes tareas para las cuales son construidos. Las más comunes son [Nwana1996]:

Asistentes: Trabajan realizando tareas típicas como el manejo de la agenda. Estos agentes ayudan al usuario a planificar las reuniones. Sus acciones incluyen negociar, aceptar o rechazar reuniones.

Filtros: Su tarea principal es la de analizar información de acuerdo a un conjunto de reglas dadas por el usuario. La aplicación típica es el filtrado de mensajes de correo electrónico.

Guías: Asisten a los usuarios en el uso de una aplicación. Estos agentes monitorean las acciones de los usuarios e intentan sugerir qué pasos a realizar para alcanzar el objetivo. Algunos ayudan a navegar por Internet.

2.8.2.2. Agentes colaborativos

Los agentes colaborativos constituyen un sistema multiagente, es decir, existe más de un agente dedicado a satisfacer los requerimientos de sus usuarios. Para ello, es necesario contar con esquemas de comunicación entre agentes que permitan la cooperación y el intercambio de conocimiento. Además, deben poseer un alto grado de autonomía para interactuar con los demás agentes.

La motivación detrás de la construcción de agentes colaborativos es que los sistemas construidos con unidades relativamente simples proveen mayor funcionalidad que un ente mayor, pudiendo extender la funcionalidad del sistema más allá de las capacidades de uno de sus miembros.

Estas arquitecturas posibilitan contar con mayor confiabilidad (debido a la redundancia) y mayor velocidad (debido al paralelismo) en el sistema conjunto.

Las áreas de aplicación de este tipo de agentes incluyen [Berney1996]:

1. Resolución de problemas demasiado grandes.
2. Interconexión de múltiples sistemas.
3. Manejo de información proveniente de fuentes distribuidas.

2.8.2.3. Agentes móviles

Los agentes móviles son procesos capaces de "viajar" por una red de computadoras, interactuando con hosts externos, recolectando información en nombre de su dueño y retornando a "casa'" luego de completar las tareas establecidas.

Los agentes forman un nivel de abstracción más para el usuario, detrás del cual se encuentran soluciones a cuestiones técnicas en algunos casos complicadas. Una de estas cuestiones es la distribución, es decir, como manejar recursos computacionales distribuidos. Con la idea de agentes móviles los recursos distribuidos no están completamente ocultos al usuario pero tampoco completamente expuestos.

La noción de movilidad viene del objetivo de reducir el tráfico innecesario dentro de una red, con lo que se pueden reducir los costos de comunicación. Además, al aportar una nueva forma de computación distribuida posibilita el mejor

aprovechamiento de los recursos de la red y permite que los usuarios tengan acceso a una cantidad mayor de recursos.

Por ejemplo, debido a que las sesiones en busca de un recurso determinado ocasionalmente son largas, la idea de agentes móviles provee una solución. Un usuario delega la tarea de búsqueda de información a un agente, establece una comunicación con la red y "envía" al agente a cumplir con su misión. La próxima vez que el usuario se conecte, el agente "retorna" con los resultados obtenidos.

Para soportar la movilidad, debe existir una infraestructura de transporte que mueva el código del agente de una ubicación a otra. Además, se debe contar con un entorno de ejecución de agentes, donde los agentes "viven", compuesto por todas las computadoras que los proveen.

Finalmente, para construir sistemas con agentes móviles es necesario resolver algunas cuestiones fundamentales tales como [Nwana1996]:

1. **Transporte:** ¿Cómo se mueven de lugar en lugar?
2. **Ejecución:** ¿Cómo ejecutar el agente de forma remota?
3. **Autenticación:** ¿Cómo saber si el agente es quien dice ser y a quién representa?
4. **Privacidad:** ¿Cómo asegurar que el agente mantenga resguardado su estado interno?
5. **Seguridad:** ¿Cómo protegerlo de virus? ¿Cómo prevenir que el agente entre en bucles infinitos o falle?

2.8.2.4. Agentes de recuperación de información

El objetivo principal de los agentes dedicados específicamente a la recuperación de información es obtener información por el usuario.

Las tecnologías de la información han expandido los horizontes de los usuarios en cuanto a las formas de generar y acceder a la misma. Pero esta amplia variedad de información distribuida plantea desafíos en cuanto a las formas de manejar su complejidad y heterogeneidad. [Nwana1996]

Hoy en día, la información se produce en múltiples contextos, se difunde por medios muy variados y se utiliza en todas partes. El rápido crecimiento de la cantidad de

documentos (especialmente en Internet) presenta la dificultad de poder acceder a la información relevante.

Se produce así el fenómeno conocido como "sobrecarga de información", entonces se trata de mejorar, pero no eliminar el problema específico de la sobrecarga y administración de la información. Además, existe el problema que debido al gran volumen de información disponible, se mezclan el ruido o desperdicio con la información útil o necesaria, lo que determina la gran dificultad de hallar lo que realmente resulta de interés para las personas.

Las soluciones actuales a este problema se basan en la construcción de motores de búsqueda, con mecanismos de indexación de documentos, combinados con interfaces de consulta apropiadas a esta tarea, o bien, índices manuales multinivel (o directorios), los cuales presentan clasificaciones de los documentos según el criterio de sus autores. Ambas técnicas poseen debilidades visibles. Los primeros, son muy propensos a "recuperar" demasiada cantidad de documentos "no deseables" ya que trabajan mediante técnica de búsqueda de la ocurrencia de los términos buscados en los documentos. Por otra parte, los índices manuales, solamente tienen "parte" de los posibles documentos, debido al alto costo que posee la recuperación, manipulación y clasificación manual de los mismos.

La motivación es poder diseñar una técnica que permita describir los cientos de millones de documentos disponibles de manera precisa, creando un índice de alta calidad, con una forma eficaz y eficiente de acceder a éste (ya sea de manera manual o automática). Una de las soluciones posibles se basa en los agentes de recuperación de información. Estos agentes pueden asistir a un usuario novato en la formulación de consultas avanzadas, en base a sus necesidades de información. Además, permiten acceder e integrar fuentes heterogéneas y manejar diferentes tipos de formatos de información.
Los agentes de recuperación de información poseen métodos para permitir el rápido acceso y recuperación de información relevante. Tienen la tarea de administrar,

manipular y juntar información de fuentes distribuidas. Pueden tener mecanismos de búsqueda y navegación flexibles y algoritmos de clasificación poderosos [Nwana1996].

El objetivo es construir agentes capaces de "armar" un diario personalizado, sabiendo dónde buscar, cómo encontrar lo buscado y cómo armarlo luego [33]. Los agentes se presentan como una herramienta muy útil en la tarea de resolver el problema de la sobrecarga de información, debido a que éstos pueden realizar sus tareas mucho más rápido que las personas y, además, se encuentran disponibles las veinticuatro horas.

2.8.3. Arquitecturas para agentes Deliberativos

En se define a una arquitectura de agente deliberativo, como una que contiene un mundo representado explícitamente y un modelo lógico del mismo, y en la cual las decisiones (por ejemplo acerca de las acciones a realizar) son hechas por medio de un razonamiento lógico (o por lo menos pseudo-lógico), basado en concordancia de patrones y manipulación simbólica. [Weiss1999]

2.8.3.1. Agentes de Planeación

Desde inicios de los setenta, la comunidad de Inteligencia Artificial dedicada a la planeación ha estado fuertemente relacionada con el diseño de agentes. Parece razonable entonces, que muchas de las innovaciones en el diseño de agentes provengan de esa comunidad. [Ramírez2004]

STRIPS: La arquitectura de este sistema, contiene una descripción simbólica del mundo y el estado deseado de objetivos, y del conjunto de descripciones de las acciones que caracterizan las pre y post condiciones asociadas con varias acciones. Tomando como base este conocimiento intenta encontrar una secuencia de acciones que lograrán el objetivo, utilizando un análisis simple. El algoritmo de planeación de STRIP es muy sencillo y se probó que es ineficiente en problemas de complejidad moderada. [Weiss1999]

Se han realizado varios intentos para construir agentes cuyo componente principal sea un planeador. Por ejemplo: el sistema Integrated Planning, Execution and Monitoring (IPEM) que está basado en un planeador sofisticado no lineal.

Enseguida presentamos la arquitectura que más se utiliza en el diseño e implementación de agentes lógicos, los conocidos como agentes BDI de sus siglas en inglés (Beliefs, Desires and Intentions).

2.8.3.2. Agentes BDI

Las creencias (Beliefs), deseos (Desires) e intenciones (Intention) - son componentes mentales presentes en muchas arquitecturas de agentes. Las creencias representan el conocimiento del agente, los deseos representan los objetivos y las intenciones otorgan deliberación al agente. La exacta definición de estos términos varía según del autor. [Geogeroff1995]

Creencias: Componente informativo necesario para determinar el estado más probable del entorno (que quizá no coincida con el estado real). Es una noción poco diferente a la del conocimiento.
Las creencias de un agente representan el conocimiento del agente. El contenido del conocimiento puede ser cualquiera, por ejemplo, conocimiento acerca del ambiente del agente o acerca de su historia. [Geogeroff1995]

Deseos: Representación del estado que motiva al agente y costo/beneficio. Los deseos son un conjunto de objetivos a largo plazo. Un objetivo es típicamente una descripción de un estado deseado del ambiente.
Los deseos proveen al agente de la motivación para actuar. Los objetivos constituyentes de los deseos pueden ser contradictorios, entonces el sistema tiene que poder, de cierta forma, elegir qué objetivo alcanzar primero, es aquí donde aparecen las intenciones. [Geogeroff1995]

Intenciones: Captura el componente deliberativo del agente e incluye planes formados por acciones, para conseguir los objetivos.

Las intenciones pueden ser consideradas como un conjunto de planes para lograr los objetivos que constituyen los deseos. Una intención es un compromiso para realizar planes que, en el momento de formularlo es sólo parcial, ya que depende del estado en el momento de su ejecución. Las intenciones son el núcleo del modelo. [Geogeroff1995]

Una intención es una elección a la que se compromete:

- Por defecto, las creencias persisten.
- Buena fé: Solo se compromete uno con lo que se siente capacitado.
- Las intenciones son las que guían el razonamiento medios-fines.
- Introspección para revisar periódicamente los estados mentales.

Decidir cómo se logra el objetivo (razonamiento medios-fines). Al comprometerse con una intención se debe [Geogeroff1995]:

- Seguir una línea de acción razonable para conseguirla, es decir, hacer los que se cree que satisface mejor la intención.
- Persistir en la intención (si falla la línea actual, intentar encontrar otra) hasta conseguirla o llegar a creer que no es posible conseguirla.
- No perder tiempo considerando opciones incompatibles.
- Cada cierto tiempo, reconsiderar las intenciones.
- Encontrar el balance entre una actitud reactiva y una pro-activa.

Este tipo de arquitectura ve al sistema como un agente racional que tiene ciertas actitudes mentales, tales como: creencias, deseos e intenciones, representando respectivamente, los estados de información, motivacional y deliberativos del agente. Estas actitudes mentales determinan el comportamiento del sistema y son críticos para lograr el desempeño adecuado u óptimo cuando la deliberación está sujeta a recursos limitados. [Geogeroff1995]

En, no es necesario, que un sistema especificado en términos de creencias, deseos e intenciones sea diseñado por medio de estructuras de datos correspondientes a cada uno de estos componentes. Sin embargo tal diseño podría simplificar la construcción, mantenimiento y verificación del sistema resultante. [Geogeroff1995]

Un agente tiene limitado sus recursos, y su capacidad de comprensión y un conocimiento incompleto del entorno en el que vive. Tiene "creencias" sobre el mundo y "deseos" que satisfacer que le llevan a formular intenciones de actuar.
El agente ejecuta las acciones que intenta realizar sin volver a razonar, hasta que se ve forzado a revisar sus intenciones, dado que se producen cambios en sus creencias o deseos.
Procesos de razonamiento práctico en el modelo BDI [Geogeroff1995]:

- Decidir qué objetivo se quiere lograr (proceso de liberación).
- Determinar opciones disponibles, según creencias y deseos.
- Elegir una o varias opciones.
- Establecer el compromiso de seguir las intenciones adoptadas.

Conceptos del modelo BDI:

- Creencias, deseos e intenciones.
- Actitudes sobre la información (creencia o conocimiento).
- Pro-actitudes (deseos, intención, elección, compromiso, actuación, etc.,...).

Propiedades de modelo:

- Consistencia (entre intenciones y creencias, etc.,...).
- Persistencia.

2.9. METODOLOGÍAS ORIENTADAS A AGENTES

La construcción del software con agentes inteligentes es una disciplina cuyas directrices las establece la Ingeniería Software Orientada a Agentes (ISOA), y que ha evolucionado de manera notable en los últimos años, con multitud de propuestas de modelado abarcando un amplio abanico de posibilidades. Metodologías como MaSE, GAIA e INGENIAS (Pavón & Gómez Sanz, 2003), definen un marco de trabajo, basado en meta modelos que permiten la especificación de sistemas Multiagente. En los siguientes puntos se hará un repaso a estas tres metodologías como ejemplos, entre otras, del Modelado de Software Orientado a Agentes. [Ramos2011]

2.9.1. GAIA

GAIA (Wooldridge & Jennings, 2000) es una metodología donde los procesos de análisis y diseño están orientados al modelado de agentes y cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global basándose en una serie de modelos de análisis y diseño. Pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que, según los autores, esté lo suficientemente detallado como para ser implementado directamente (Figura 3). [Ramos2011]

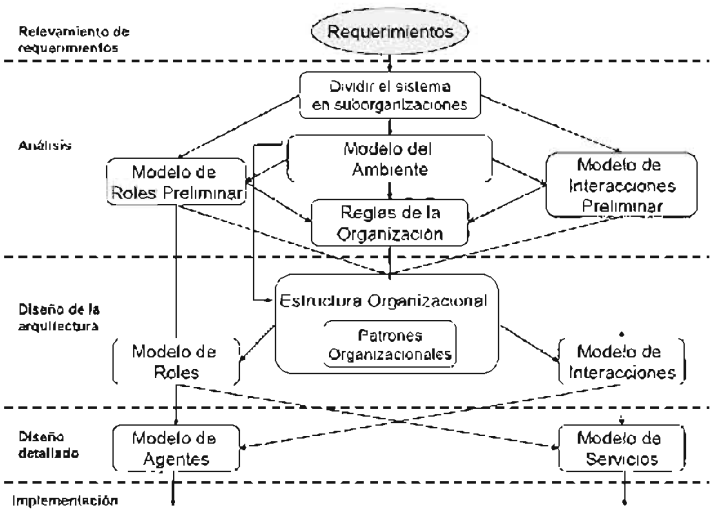


Figura 2.3 Fases de GAIA

Fuente: [Ramos2011]

En GAIA se hace referencia a 5 fases (Requerimiento, Análisis, Diseño de arquitectura, Diseño detallado e Implementación) y presenta dos conceptos principales que son las entidades abstractas y las concretas. Las entidades abstractas son aquellas empleadas en el análisis para obtener un conocimiento conceptual del sistema, aunque no necesariamente tienen una implementación directa dentro del mismo. Las entidades concretas son empleadas en la fase de diseño y tienen su representación directa en el sistema.

La etapa de análisis en GAIA busca el entendimiento del sistema y su estructura como una organización. Una organización es una entidad abstracta que puede ser

vista como un conjunto de roles que mantienen ciertos tipos de interacción y ciertas relaciones con otros roles del sistema tomando parte en patrones institucionalizados de interacción. Un rol se define por cuatro atributos: las responsabilidades, los permisos, las actividades y los protocolos.

Las responsabilidades, a su vez, pueden fraccionarse en dos categorías: las propiedades de vida y las propiedades de seguridad. Las primeras describen estados de las tareas que un agente debe efectuar, indicando qué debe ser. Las responsabilidades determinan la funcionalidad y son tal vez los atributos claves asociados con un rol.

Los permisos identifican los recursos que el rol tiene disponibles para realizar sus responsabilidades. Por lo general, estos tienden a ser recursos de información y con cada recurso tiene asociado unos derechos vinculados como, por ejemplo, leer, cambiar o generar el recurso.

Las actividades representan acciones privadas u operaciones que el agente puede completar sin la necesidad de interactuar con otros agentes en el sistema.

Los protocolos definen las formas en que un agente, asumiendo dicho rol, podría interactuar con otros. [Ramos2011]

Durante el análisis con GAIA se generan dos tipos de modelos: de roles y de interacción.

El Modelo de Roles: Permite identificar los distintos tipos de roles que se encuentran en el sistema. Con cada uno hay asociados permisos y responsabilidades. Las responsabilidades son de dos tipos: de vida y de seguridad. Con estos conceptos se puede especificar un esquema de rol para cada uno de ellos en la organización y conformar así el modelo de roles. Un esquema de rol presenta su descripción, sus protocolos y actividades, sus permisos y las responsabilidades de vida y seguridad. El modelo de rol preliminar identifica las capacidades básicas requeridas de la organización.

El Modelo de Interacción: Es una serie de definiciones de protocolo, una para cada tipo distinto de interacción entre los roles. Cada definición de protocolo consiste de un propósito en la interacción, los roles iniciador y receptor, las entradas y salidas y el procesamiento ejecutado. El modelo de interacción preliminar identifica las interacciones básicas que son necesarias para poder cumplir con los roles preliminares.

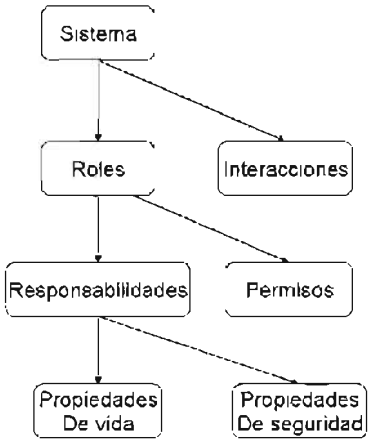


Figura 2.4 Fase de análisis
Fuente: [Ramos2011]

En resumen, el proceso de análisis en GAIA debe identificar los roles del sistema, identificar y documentar cada uno de los protocolos asociados y detallar el modelo de roles usando como base el modelo de protocolos.

El proceso de diseño en GAIA involucra la transformación de los modelos de análisis a un nivel de abstracción lo suficientemente bajo como para que sea posible implementar los agentes haciendo uso de las técnicas de diseño tradicionales, tales como las de orientación a objetos. En la etapa de diseño, GAIA hace uso de tres tipos de modelos: el modelo de agentes, el modelo de servicios y el modelo de conocidos.

El Modelo de Agentes: Permite definir los diferentes tipos que se pueden encontrar en el sistema y el número de instancias que se tienen de cada uno en tiempo de ejecución. Un tipo de agente es un conjunto de roles, es decir, un agente puede asumir uno o más roles aunque lo contrario no es cierto. El modelo se construye

mediante un árbol de tipos de agentes, en el cual los nodos hoja corresponden a los roles y los restantes a los tipos de agentes. Para cada tipo se debe definir un cuantificador de instancia, el cual precisa el número de instancias que se tendrán en el sistema en tiempo de ejecución.

El Modelo de Servicios: Expone los servicios que cada tipo de agente va a implementar, entendiendo por servicio cierta funcionalidad. Cada uno es derivado de las actividades y protocolos, así como de sus propiedades de vida y seguridad encontradas en la etapa de análisis. El modelo de servicios se compone de las propiedades de cada uno de los servicios: las entradas, las salidas, las precondiciones y las post condiciones. Las entradas y salidas proceden de forma directa del modelo de protocolos. Las precondiciones y las post condiciones constituyen límites en los servicios y son derivadas de las propiedades de un rol.

El Modelo de Conocidos: Permite precisar los enlaces de comunicación que existen entre tipos de agentes e identificar posibles problemas de embotellamiento surgidos por el uso de estos canales de comunicación. El modelo es un grafo dirigido, en donde cada nodo corresponde a un tipo de agente y las aristas se relacionan con los caminos de comunicación. Así pues, un grafo [A $\rightarrow$ B], muestra que hay un camino de A a B, pero no necesariamente de B a A.

El modelo de diseño debe, en definitiva, crear un agente, asociando los roles a los tipos de agentes y documentando las instancias de cada uno de los tipos de agente. A su vez, tiene que desplegar un modelo de servicios, examinando protocolos y propiedades de vida y seguridad, y desarrollar un modelo de conocimiento por medio del modelo de interacción y el modelo de agente. [Ramos2011]

2.9.2. Ingenias

La metodología INGENIAS (Pavón & Gómez Sanz, 2003) se centra en el desarrollo de sistemas multiagente (MAS) y la herramienta de soporte INGENIAS Developtment Kit (IDK) (Pavón, Gómez-Sanz, & Fuentes, 2006). Está basada en MESSAGE (Caire,

y otros, 2001), por lo que aprovecha la definición de meta-modelos (una descripción de alto nivel acerca de los elementos que posee el modelo) para especificar los elementos que se pueden usar para describir cada uno de los aspectos que constituyen un sistema multiagente. Usa como notación una extensión de UML y adopta el Proceso Unificado de Rational (RUP) como ciclo de vida, definiendo las actividades que se deben llevar a cabo en sus diferentes etapas. Además, proporciona una herramienta que permite modelar, documentar y realizar una generación de código automática para crear un sistema multiagente: IDK. [Ramos2011]

INGENIAS mejora MESSAGE en los siguientes aspectos:

Integra las vistas del diseño del sistema, conectando los conceptos de los diferentes diagramas. Integra el ciclo de vida del desarrollo software creando una relación más estrecha entre el RUP e INGENIAS.

Da soporte tecnológico al lenguaje a través del IDK, una herramienta de código abierto que permite la especificación y generación automática de código para implementar Sistemas Multiagente. Se ha desarrollado con el propósito de construir un framework completo de desarrollo portable, extensible y configurable, haciendo uso de ficheros XML que proporcionan un mecanismo altamente flexible para el intercambio de información con otros sistemas.

Permite la implementación de Sistemas Multiagente partiendo del modelo diseñado, generando el código fuente para la creación y control de los agentes.

INGENIAS usa el concepto del metamodelo especificando los Sistemas Multiagente desde 5 puntos de vista: organización, agentes, tareas/objetivos, interacciones y dominio. [Ramos2011]

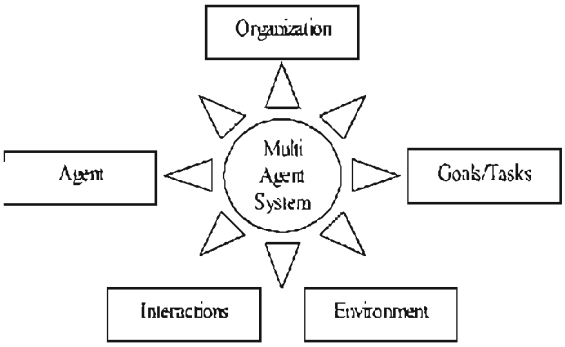


Figura 2.5 Cinco puntos de vista de un Sistema Multiagente
Fuente: [Ramos2011]

El método de desarrollo de SMA propuesto en INGENIAS concibe el SMA como la representación de un conjunto de modelos. Cada uno de estos modelos muestra una visión parcial del SMA: los agentes que lo componen, las interacciones que existen entre ellos, cómo se organizan para proporcionar la funcionalidad del sistema, qué información es relevante en el dominio y cómo es el entorno en el que se ubica el sistema a desarrollar. [Ramos2011]

El metamodelo de organización define cómo se agrupan los agentes, la funcionalidad del sistema y qué restricciones hay que imponer sobre el comportamiento de los agentes.

El metamodelo de agente describe agentes particulares y los estados mentales en que se encontrarán a lo largo de su vida.

El metamodelo de entorno define qué existe alrededor del nuevo sistema y cómo lo percibe cada agente.

El metamodelo de interacción describe que interacciones existen entre los agentes y los roles.

El metamodelo de objetivos y tareas identifica objetivos generales y los descompone en objetivos más concretos que se pueden asignar a agentes.

El modelado en INGENIAS se basa en la herramienta IDK, que proporciona un mecanismo de implementación de Sistemas Multiagente basados en los principios del Desarrollo de Software Dirigido por Modelos (DSDM) (Pavón, Gómez-Sanz, & Fuentes, 2006). El usuario define la especificación del Sistema Multiagente con el

editor del IDK, representando el modelo que dirige el desarrollo del Sistema Multiagente. El IDK emplea las especificaciones creadas para generar automáticamente código, que puede ser modificado por el desarrollador y, si lo desea, usar el IDK para actualizar los modelos originales. Esto le confiere un soporte bidireccional entre las especificaciones y el código generado a partir de ellas, pudiendo abordar el problema desde ambos puntos de vista. [Ramos2011]

2.9.3. MaSE

MaSE (Multiagent System Engineering) es una metodología completa basada en el ciclo de vida clásico del software, con un entorno propio de desarrollo (agentTool) para analizar, diseñar y construir sistemas multiagente heterogéneos.
En esta metodología, los agentes no son considerados entes autónomos, preactivos y sociales, sino simples procesos que se comunican para conseguir el objetivo global del sistema. Los agentes son vistos únicamente como una abstracción conveniente que podría (o no) tener inteligencia. [Gallego2004]

Según Sycara [Sycara,1998], las seis batallas principales en el camino de los sistemas multiagente son: [Gallego2004]

1. Descomposición de los problemas y asignación de tareas a agentes individuales.
2. Coordinación del control de los agentes y las comunicaciones.
3. Hacer que múltiples agentes actúen de una forma coherente.
4. Razonamiento acerca de otros agentes y el estado de coordinación.
5. Resolver problemas de agentes con objetivos conflictivos.
6. Ingeniar sistemas multiagente prácticos.

El objetivo de MaSE es resolver el sexto punto de Sycara [Sycara, 1998] y proporcionar un marco de trabajo para la resolución de los cinco primeros puntos. Para esto, MaSE utiliza una serie de modelos gráficos que describen el sistema y sus

interfaces interagente, así como el funcionamiento interno de los agentes, de forma independiente a la plataforma.

MaSE se divide, fundamentalmente, en dos fases: análisis y diseño. A su vez, la primera fase se subdivide en 3 pasos: captura de objetivos, aplicación de casos de uso y refinamiento de roles. La fase de diseño también está subdividida, aunque en 4 pasos: creación de las clases de agentes, construcción de las conversaciones, ensamblaje de las clases y diseño del sistema (Figura 6).

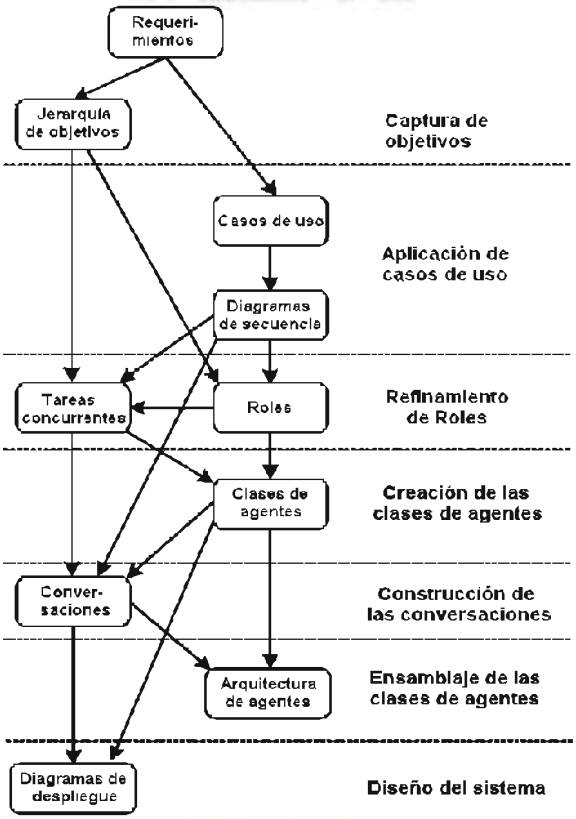


Figura 2.6 Proceso de desarrollo en MaSE.
Fuente: [Gallego2004]

2.9.3.1. Captura de objetivos

En esta fase, el analista debe cumplir 2 tareas: identificar y estructurar los objetivos. Primero, a partir de los documentos de requerimientos, se extraen los objetivos principales del sistema, sin tener en cuenta las actividades o desarrollos que llevan hasta estos. Esto se hace así porque los objetivos son lo que menos tiende a

cambiar en el tiempo. Después, estos objetivos son analizados y estructurados según su importancia en un diagrama de jerarquía de objetivos (Figura 2.7). Los subobjetivos son necesarios para la consecución de los objetivos principales.

Es importante notar que todos los objetivos son siempre de nivel de sistema. En algunos casos, el analista puede asociar un rol a cada objetivo.

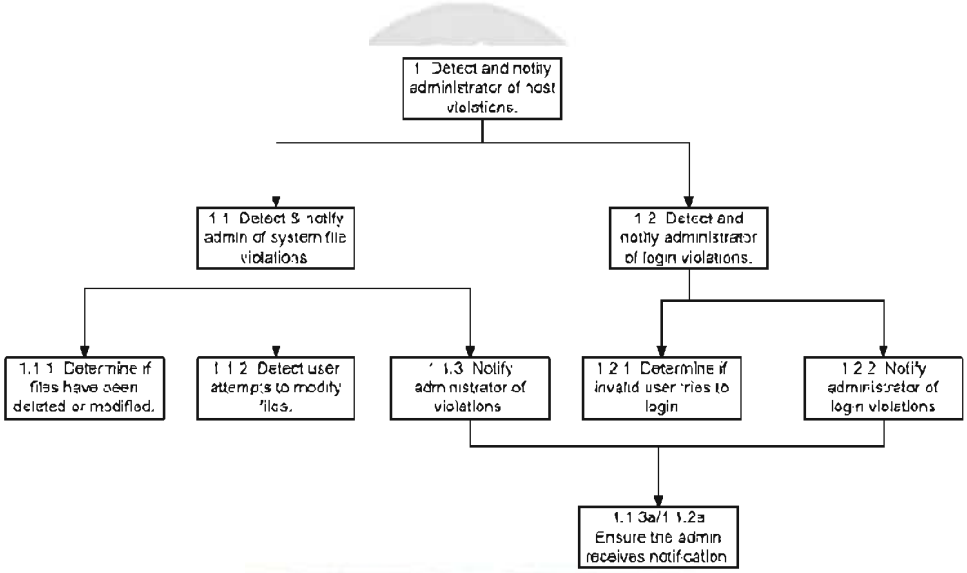


Figura 2.7. Diagrama de jerarquía de objetivos en MaSE.
Fuente: [Gallego2004]

2.9.3.2. Aplicación de casos de uso

Este paso es crucial para convertir objetivos en roles y tareas asociadas. El analista dibuja casos de uso para explicar el comportamiento deseado del sistema. Después, estructura los casos de uso en diagramas de secuencia mostrando la secuencia de eventos entre roles y, como resultado, define la mínima comunicación necesaria entre ellos. [Gallego2004]

2.9.3.3. Refinamiento de roles

El tercer paso en MaSE es para asegurar que hemos identificado todos los roles necesarios y desarrollar las tareas que definen el comportamiento de los mismos y los patrones de comunicación.

Se asegura que todos los objetivos son tenidos en cuenta asignando un rol a cada uno, siendo desempeñado este último por al menos un agente en el diseño final. Pese a que, generalmente, cada objetivo es mapeado a un rol individual, existen situaciones en las que conviene asignar más de un objetivo a un mismo rol, por motivos de eficiencia. Este tipo de decisiones son basadas en conceptos clásicos de ingeniería del software como la cohesión funcional, comunicacional, procedural o temporal. También podría ser por distribución de recursos o por cuestiones especiales de las interfaces. Finalmente, los roles son captados en el modelo de roles (Figura 2.8). [Gallego2004]

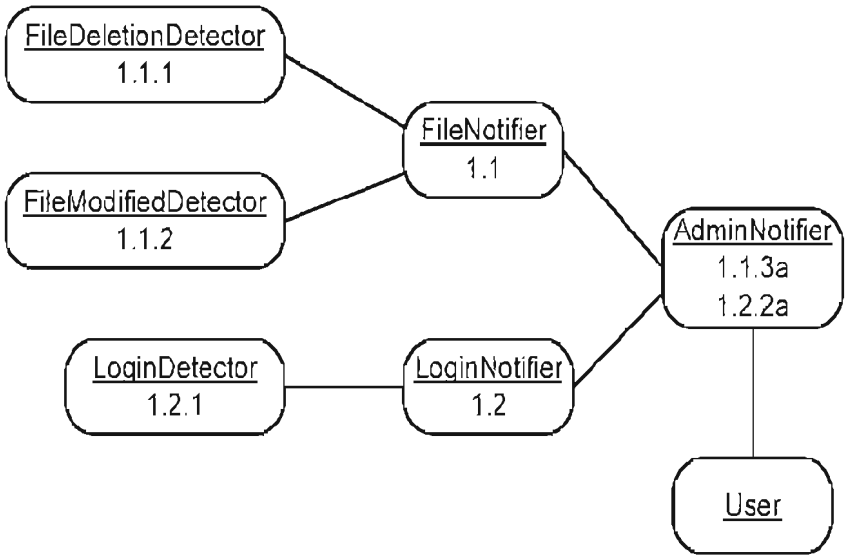


Figura 2.8. Modelo de roles en MaSE.

Fuente: [Gallego2004]

Una vez obtenidos los roles, la parte más difícil de MaSE consiste en convertirlos en clases de agentes con su comportamiento interno y sus conversaciones entre ellos. Para conseguir esto, necesitamos definir tareas de alto nivel que puedan ser transformadas en funcionalidades específicas de agentes. Estas funcionalidades nos ayudan a definir los componentes internos de los agentes y los detalles de las conversaciones en que participan. Todo esto queda reflejado en el modelo de roles detallado (Figura 2.9). [Gallego2004]

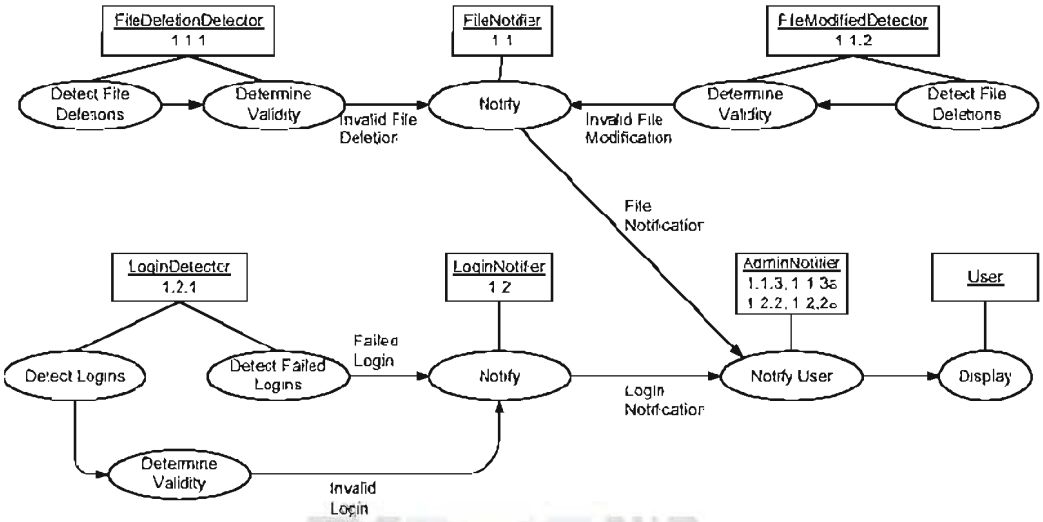


Figura 2.9. Modelo de roles detallado en MaSE.

Fuente: [Gallego2004]

2.9.3.4. Creación de las clases de agentes

Las clases de agentes son identificadas a partir de los roles y descritas en el diagrama de clases de agentes como muestra la (Figura 2.10). De nuevo, lo más corriente es establecer una correspondencia uno a uno entre roles y clases de agente. Sin embargo, el diseñador puede decidir unificar varios roles en una clase o crear varias clases de un mismo rol. Una vez creado el diagrama de clases, la organización del sistema queda definida. En este punto es bueno fusionar roles que tengan en común un gran volumen de tráfico de mensajes para optimizar el sistema.

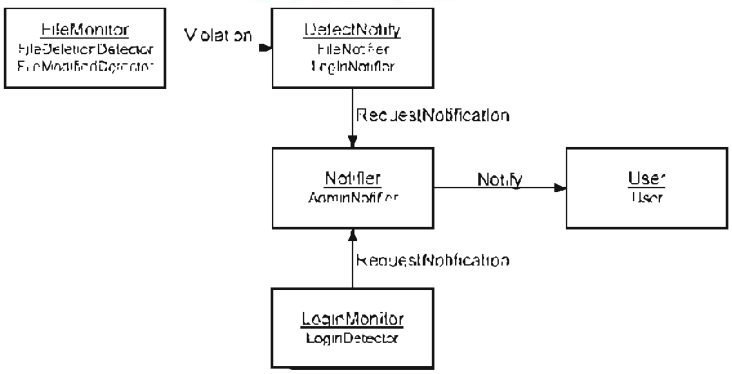


Figura 2.10. Diagrama de clases de agentes en MaSE.

Fuente: [Gallego2004]

2.9.3.5. Construcción de conversaciones

Este paso se encuentra íntimamente ligado con el siguiente (Ensamblaje de agentes). Una conversación MaSE define un protocolo de coordinación entre dos agentes. Específicamente, una conversación consiste en dos diagramas de clases de comunicación, uno para el emisor y otro para el receptor. Un diagrama de clases de comunicación es un par de máquinas de estados finitos que definen una conversación entre dos clases agente participantes (Figura 2.11). [Gallego2004]

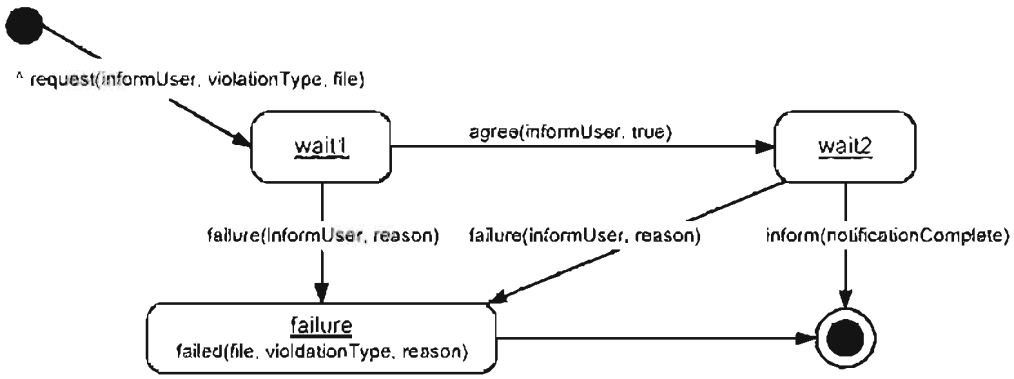


Figura 2.11. Diagrama de clases de comunicación en MaSE.
Fuente: [Gallego2004]

2.9.3.6. Ensamblaje de agentes

En este paso se crea el interior de los agentes. Este proceso se simplifica usando un lenguaje de modelado arquitectónico que combina la naturaleza abstracta de los lenguajes tradicionales de descripción arquitectónica con el leguaje de restricción de objetos, que permite al diseñador especificar detalles de bajo nivel. [Gallego2004]

2.9.3.7. Despliegue final del sistema

En esta fase se decide la configuración final del sistema a ser implementado. Hasta ahora en MaSE sólo se consideran sistemas estáticos no móviles. En MaSE se define la arquitectura global del sistema mediante diagramas de despliegue para mostrar el número, tipo y localización de los agentes.

En este momento también se toman las decisiones de implementación que no se habían tomado antes, como el lenguaje de programación a usar o el "framework" para las comunicaciones. [Gallego2004]

2.9.3.8. AgentTool

La herramienta agentTool es el software de soporte que actualmente implementa los siete pasos del proceso MaSE con soporte para transformar modelos de análisis en modelos de diseño. Permite crear de modo visual todos los diagramas del proceso de desarrollo, pudiendo realizar cualquiera de los 7 pasos en el orden que se prefiera (siempre respetando las dependencias entre unos y otros). Además, dispone de una base de conocimiento persistente, un verificador de conversaciones y generación automática de código. [Gallego2004]

2.9.3.9. Movilidad

Los estudios recientes sobre MaSE y agentTool se concentran en dar soporte a agentes móviles. El paso inicial para dar este soporte es el de incluir una actividad de movimiento que, básicamente, pide que el agente cambie de lugar. El proceso devuelve un booleano indicando si el agente se ha movido o no. Este añadido a la fase de análisis permite al analista especificar cuándo debe ocurrir un movimiento, su lugar de destino, y saber si se ha podido realizar la operación.

En la fase de diseño todo se complica más. MaSE debe poder informar a cada componente de la petición de movimiento, así como salvar el nuevo estado, una vez que se haya realizado el movimiento. [Gallego2004]

2.10. HERRAMIENTAS DE DESARROLLO

2.10.1. Bases de Datos MySQL

MySQL es un sistema de bases de datos relacional. A pesar de su popularidad, este sistema todavía no tiene algunas de las características que poseen otros sistemas comerciales. Sin embargo, estas diferencias son cada vez más pequeñas. Algunas

de las compañías que usan MySQL son Yahoo!, BBC News, CNET, Nokia, YouTube, Flickr, Google, etc. [Martín2009]

Las principales característica de MySQL son:

- Sistema de base de datos relacionales
- Arquitectura cliente/servidor
- Compatibilidad con SQL.
- SubSELECTS: Es capaz de procesar consultas del tipo:
  SELECT * FROM table1 WHERE x IN (SELECT y FROM table2)
- Vistas (Views). Son consultas SQL cuyos resultados son tratados como si fueran una tabla diferente, permitiendo tener diferentes vistas de una misma tabla.
- Triggers: Son comandos SQL que se ejecutan automáticamente en ciertas operaciones (INSERT, UPDATE y DELETE).
- Replicación: Permite copiar el contenido de una base de datos a múltiples ordenadores. Esta característica se usa para mejorar la protección contra fallos, y parta acelerar las consultas.
- Transacciones: Una transacción es un conjunto de operaciones sobre una base de datos que funcionan como un bloque. Es sistema asegura que o bien se han ejecutado todas las operaciones, o ninguna. Esto es válido incluso si hay una caída del sistema, un fallo eléctrico, o cualquier otro desastre.
- Restricciones sobre claves externas (foreign keys). Son reglas con las que se asegura la integridad de las relaciones entre tablas.
- Lenguajes de programación: Hay un gran número de APIs y librerías para desarrollar aplicaciones MySQL. Por ejemplo, se pueden usar lenguajes como C, C++, Java, Perl, PHP, Python y Tcl.
- ODBC: Permite que MySQL sea usada con la interfície ODBC de Microsoft, lo que hace que pueda ser usada por lenguajes como Dephi, VisualBasic, etc. [Martín2009]

2.10.2. Lenguaje de Programación Visual Studio.Net

2.10.2.1. Plataforma .NET

.NET es un conjunto de tecnologías construidas a partir de una estrategia de Microsoft para la cual ha destinado gran parte de su presupuesto de investigación. Esta estrategia surge, a su vez, del nuevo mapa de necesidades y requerimientos que Microsoft advirtió que se suscitarían en el futuro.

.NET son aplicaciones de servidor (SQL Server 2000, Exchange 2000, etc.), es un entorno de desarrollo (Visual Studio .NET), son componentes clave que se integran al sistema operativo, son servicios, y es, finalmente, una plataforma de desarrollo denominada framework .NET. [Microsoft2011]

2.10.2.2. El framework .NET

Es una infraestructura de desarrollo que está compuesta por diversos recursos, entre los cuales se destaca el más importante, que es una máquina virtual conocida como CLR (Common Language Runtime), sobre la cual se ejecutan las aplicaciones.

De este modo, nuestros programas ya no poseerán código nativo de ningún microprocesador en particular, sino instrucciones MSIL (Microsoft Intermediate Language) que serán traducidas a código nativo en el momento de su ejecución (por medio de un compilador Just In Time).

El framework también define una librería base de clases, BCL (Base Class Library), a la cual puede acceder cualquier desde lenguaje desarrollado para la plataforma.

Por encima de la infraestructura se ubicará un conjunto de reglas básicas que debe implementar un lenguaje para poder ser parte de la familia .NET. Esta especificación es conocida como CLS (Common Language Specificeation).

Finalmente, se encuentra el conjunto de lenguajes que cumplan con la especificación CLS, como el C#, el VB.NET, Managed C++, etc.

Es posible crear recursos en cualquiera de estos lenguajes que hagan uso de recursos escritos en otros; de hecho, es posible crear una clase en C# que herede de una clase creada en Managed C++.[Microsoft2011]

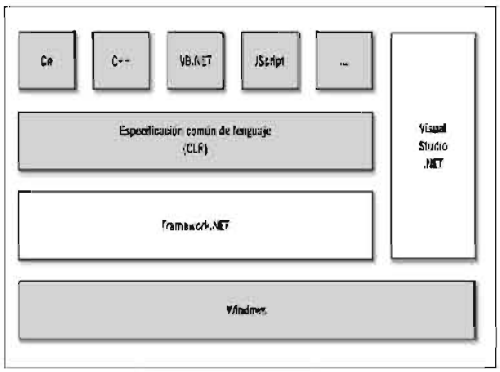


Figura 2.12 Arquitectura básica de la infraestructura de desarrollo

Fuente: [Microsoft2011]

2.10.2.3. El lenguaje C#

C# es un lenguaje que cumple con la especificación CLS. El código que crearemos con él será traducido a instrucciones MSIL para entonces ser traducido, justo antes de su ejecución, a instrucciones nativas que correspondan a la plataforma concreta sobre la cual estemos trabajando.

Cabe destacar que el compilador JIT (Just In Time) traduce el código MSIL a código nativo no de manera monolítica, sino por métodos, módulos y componentes.

Por lo tanto, a grandes rasgos: código que no sea ejecutado no será compilado.

El código MSIL generado a partir de la compilación de código C# es idéntico al código MSIL generado a partir de cualquier otro lenguaje CLS. Esto podría abrir el interrogante de ¿por qué programar en C# en lugar de hacerlo en VB.NET o en Managed C++ o, incluso, en Delphi .NET? Esta pregunta podría responderse con otra: ¿por qué programar en C++ en lugar de hacerlo en C o Pascal, o en cualquier otro lenguaje compilado, si todos generan el mismo código Intel x86?

Cada lenguaje posee sus características que lo tornan ideal para ciertos usos; además, presenta diversos grados de expresividad que pueden permitir implementar el mismo algoritmo de maneras diversas, por lo que un modo puede resultar más eficiente que otro. [Microsoft2011]

2.10.2.4. Características fundamentales del lenguaje

C# es un lenguaje moderno y altamente expresivo que se ajusta al paradigma de programación orientada a objetos. Su sintaxis es similar a C++ y Java. El lenguaje

fue desarrollado en gran parte por Anders Hejlsberg (creador del mítico compilador Turbo Pascal1 y uno de los diseñadores líder del lenguaje de programación Delphi). En C# no existe el concepto de función global o variable fuera de una clase u objeto. Por su buen apego a la POO, es posible sobrecargar métodos y operadores. Soporta definición de interfaces. Ninguna clase puede poseer más de un padre (no se permite la herencia múltiple), pero sí puede suscribir un contrato con diversas interfaces. Soporta la definición de estructuras, pero, a diferencia de C++, aquí no son tan parecidas a las clases y poseen ciertas restricciones que veremos luego.

Permite además la declaración de propiedades, eventos y atributos (que son construcciones declarativas).

Por último, una característica distintiva cada vez más presente en lenguajes modernos es la gestión automática de memoria y el uso de referencias en lugar de punteros. Gracias a esta gestión automática de memoria ya no tendremos que preocuparnos por la existencia de memory leaks (zonas de memoria que permanecen reservadas pero ya no son utilizadas debido a errores de programación). [Microsoft2011]

CAPITULO III

3. MARCO APLICATIVO

3.1. INTRODUCCIÓN

En el presente capitulo se describe los procesos de análisis diseño implementación y pruebas, teniendo como base metodológica a Scrum. Se incluyen los diagramas de casos de uso, actividad, secuencia, clases, componentes y despliegue. Además de las pruebas e integración de la interface con el sistema.

Como se mencionó anteriormente la metodología Scrum consta de las siguientes etapas que serán detalladas durante el proceso de modelado:

- Identificar los requerimientos en base a las historias de usuario.
- Construir las pilas de producto.
- Identificar la pila de sprint.
- Definir las de tareas.

3.2. ANALISIS DE LOS PROCESOS ACTUALES

En la primera parte de este capítulo se reflejan los procesos que se llevan a cabo en la Asistencia Pública, los cuales son presentados por medio de los siguientes diagramas de actividad: asignación de consulta, registro y actualización de nuevos pacientes, verificación de historias clínicas y diagnostico medico.

Cada diagrama da una visión clara sobre las necesidades y dificultades que se tienen al pasar por cada proceso.

3.2.1. Proceso de asignación de consulta

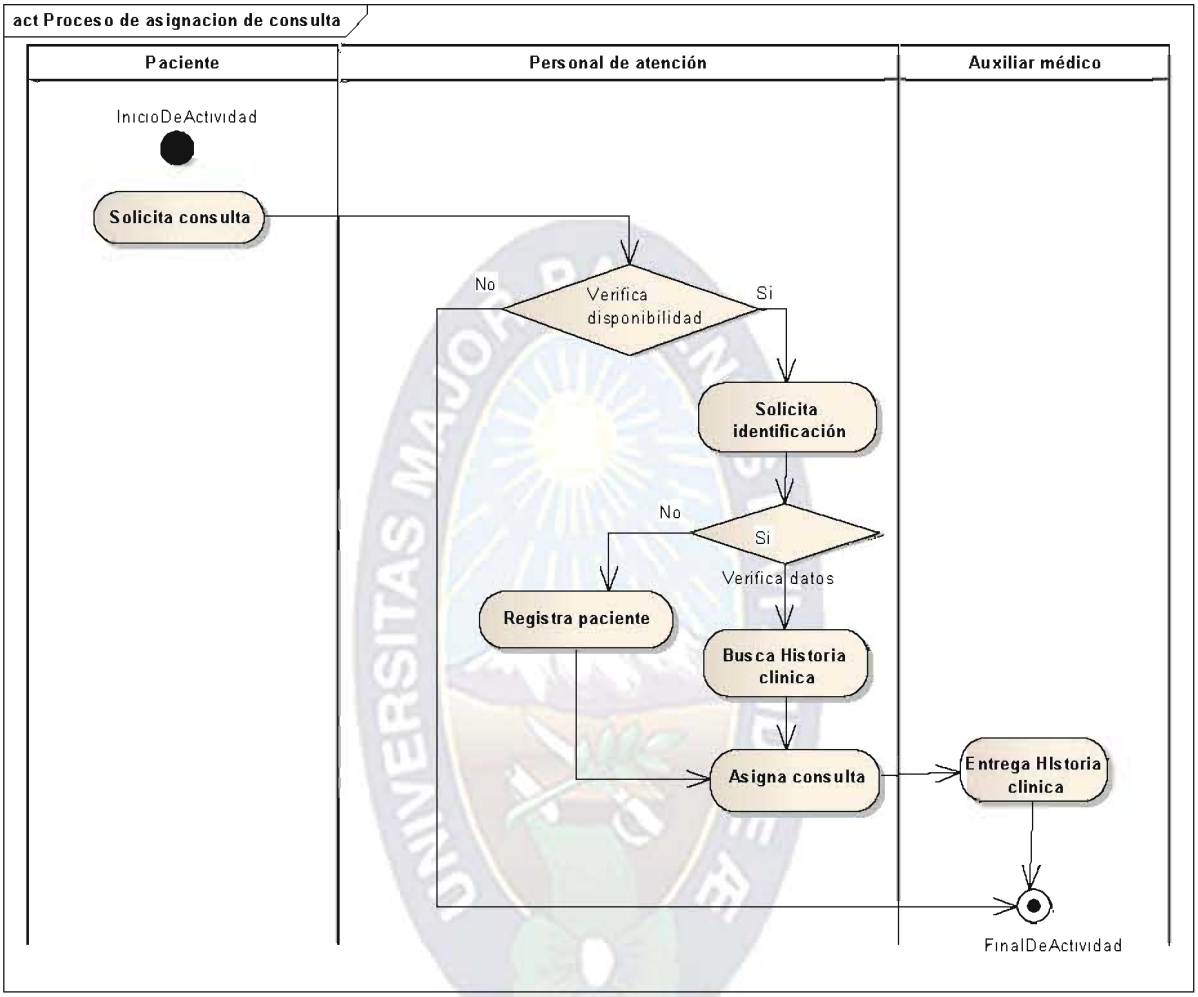


Figura 3.1. Proceso de Asignación de consulta

Fuente: Elaboración según [I. Jacobsob2000]

3.2.2. Descripción del proceso de asignación de consulta

Objetivo: El objetivo de este proceso es el de asignar a un paciente la consulta médica que requiera.

Funcionarios participantes:

- Personal de atención
- Auxiliar médico

Flujo:

1. El paciente solicita consulta médica
2. El personal de atención verifica la disponibilidad de atención
3. Si existe disponibilidad el personal de la asistencia pública solicita los documentos del paciente.
4. Realiza la verificación de los datos, si el paciente ya existe se busca la historia clínica según numero de historia asignado, si el paciente no existe se registra como nuevo paciente.
5. El personal de atención asigna la consulta.
6. El personal de atención entrega las historias clínicas al auxiliar medico.

3.2.3. Proceso de diagnóstico medico

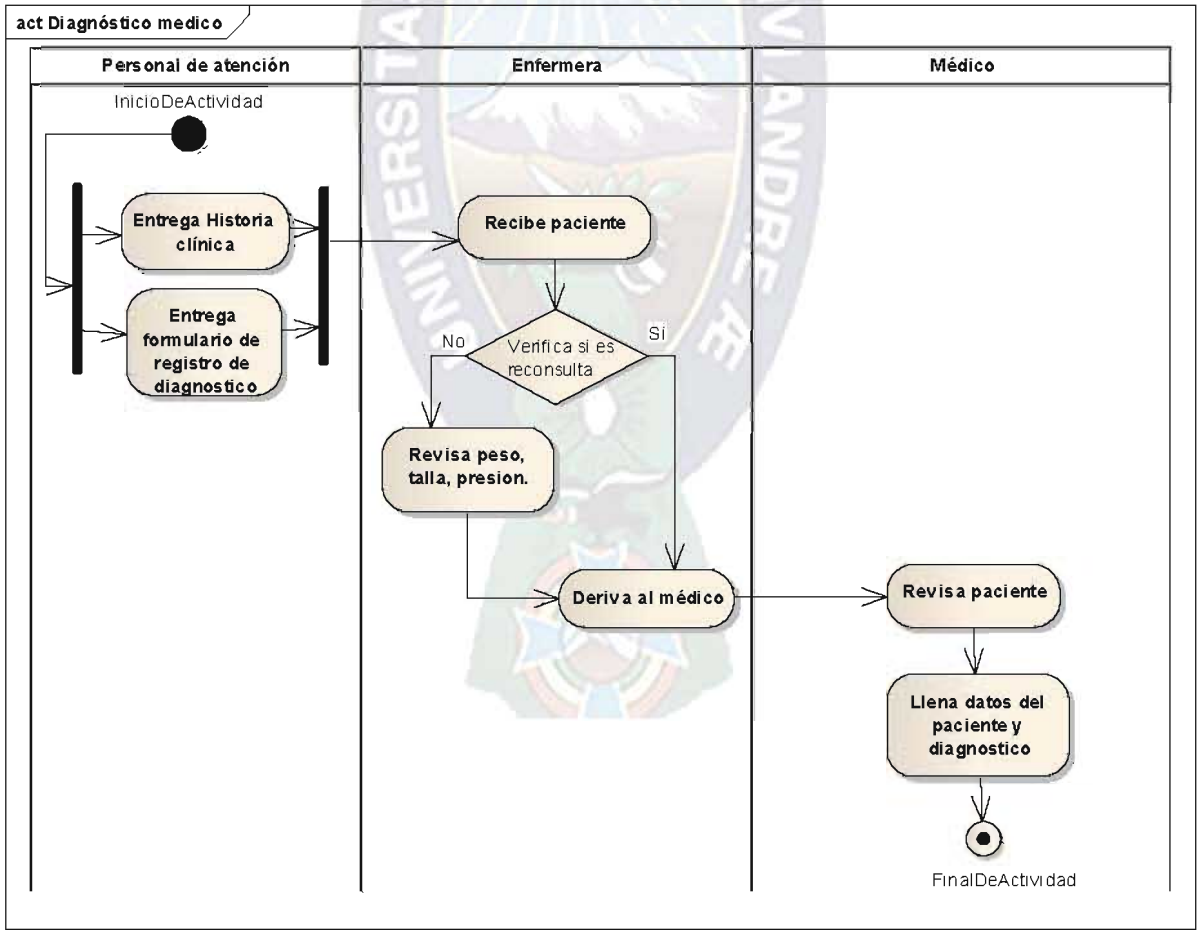


Figura 3.2. Proceso de Diagnostico médico

Fuente: Elaboración según [I. Jacobsob2000]

3.2.4. Descripción del proceso de diagnostico medico

Objetivo: El objetivo de este proceso es el de asignar al paciente un diagnostico medico.

Funcionarios participantes:

- Personal de atención
- Médico
- Enfermera

Flujo:

1. El personal de atención entrega la historia clínica y el formulario de registro diario de diagnostico a la enfermera.
2. La enfermera recibe a los pacientes y verifica si son nuevos pacientes o re consulta.
3. Si es re consulta deriva directamente al paciente con el médico, si es nuevo paciente la enfermera le toma la talla el peso la presión y todo lo necesario para la consulta y luego lo deriva con el médico.
4. El médico revisa al paciente, y registra en el formulario de de registro diario de diagnostico los datos del paciente y su diagnostico.

3.2.5. Proceso de registro y actualización de pacientes

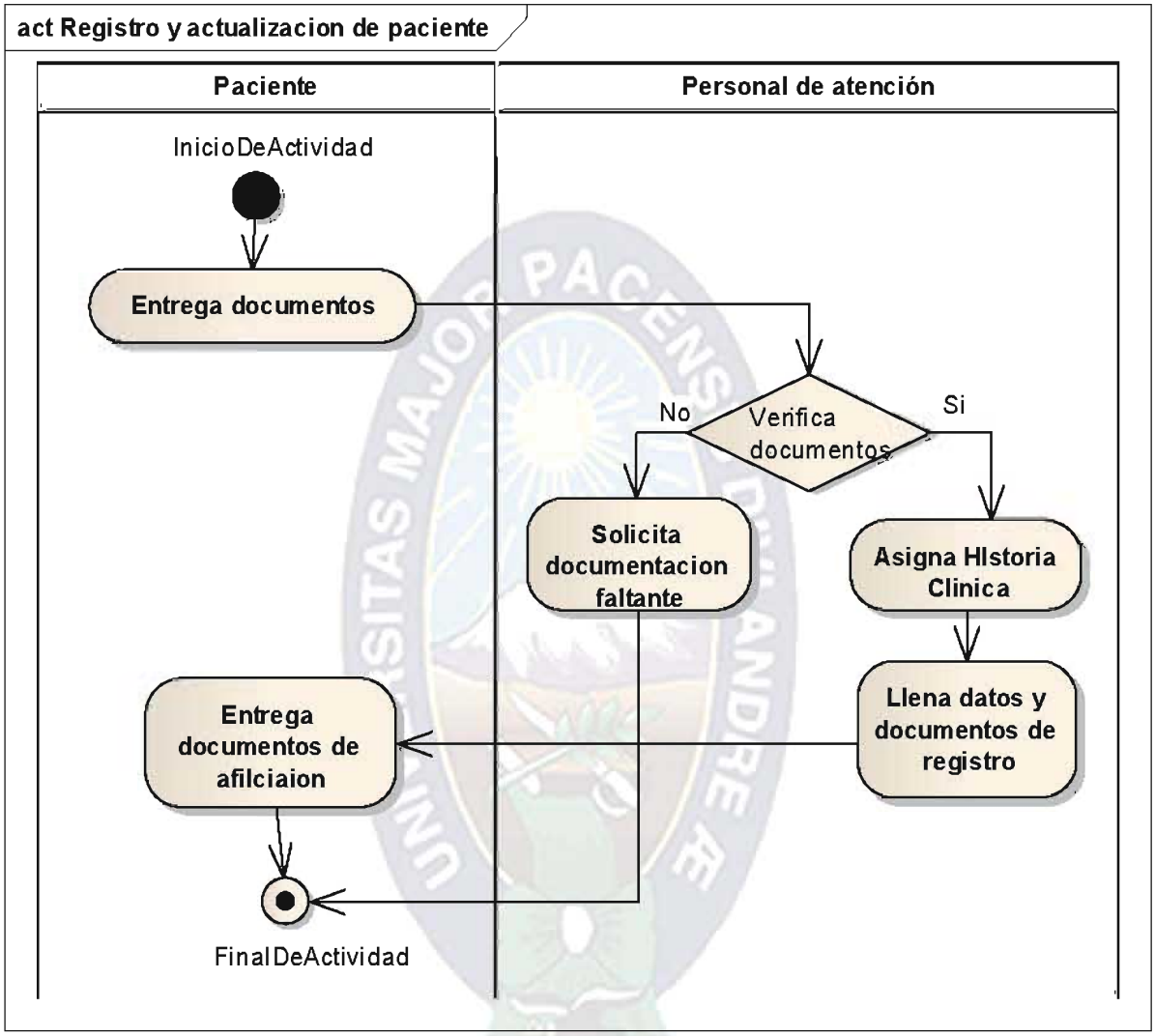


Figura 3.3. Proceso de Registro y actualización de pacientes

Fuente: Elaboración según [I. Jacobsob2000]

3.2.6. Descripción del proceso de registro y actualización de pacientes

Objetivo: El objetivo de este proceso es el registrar a nuevos pacientes también el actualizar datos necesarios del paciente.

Funcionarios participantes:

- Personal de atención.
- Paciente.

Flujo:

1. El paciente entrega los documentos necesarios para su afiliación.
2. El personal de atención verifica si los documentos están completos.
3. Si los documentos no son correctos le pide al paciente que regrese con los documentos faltantes.
4. Si los documentos están correctos le asigna número de historia clínica, llena los datos correspondientes en la historia clínica.
5. El personal de atención llena los documentos de identificación del paciente.
6. El personal de atención le entrega los documentos de afiliación al paciente.

3.2.7. Proceso de verificación de historia clínica

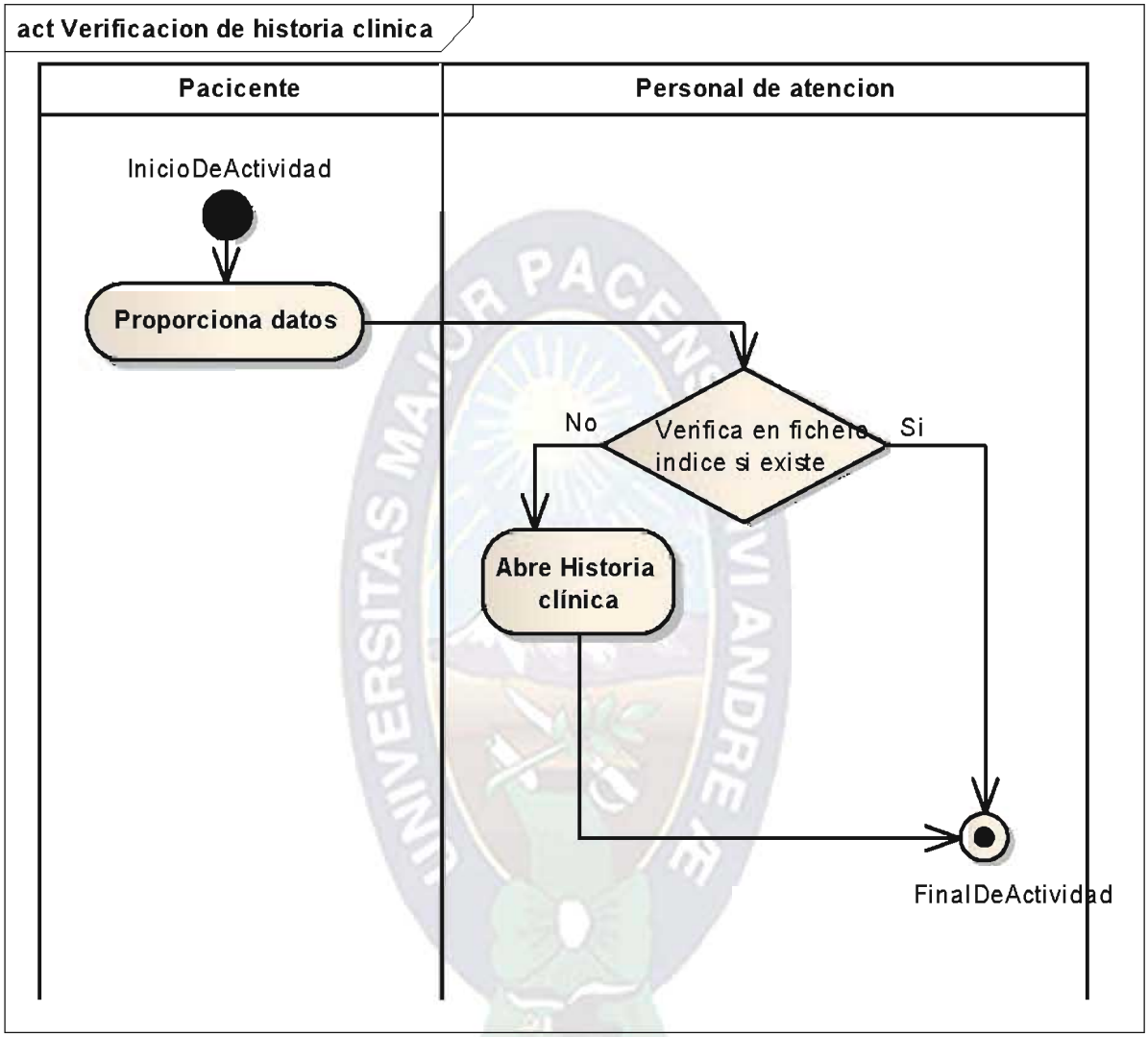


Figura 3.4. Proceso de Verificación de historias clínicas

Fuente: Elaboración según [I. Jacobsob2000]

3.2.8. Descripción del proceso de verificación de historia clínica

Objetivo: El objetivo de este proceso es realizar la verificación si existe la historia clínica.

Funcionarios participantes:

- Paciente.
- Personal de atención.

Flujo:

1. El paciente proporciona los datos.
2. El personal de atención verifica en el fichero índice si existe el paciente.
3. Si existe el paciente no se crea ninguna historia clínica.
4. Si el paciente no se encuentra se crea nueva historia clínica.

3.3. ELABORACIÓN DEL SISTEMA

3.3.1. Casos de Uso

3.3.1.1. Casos de Uso General

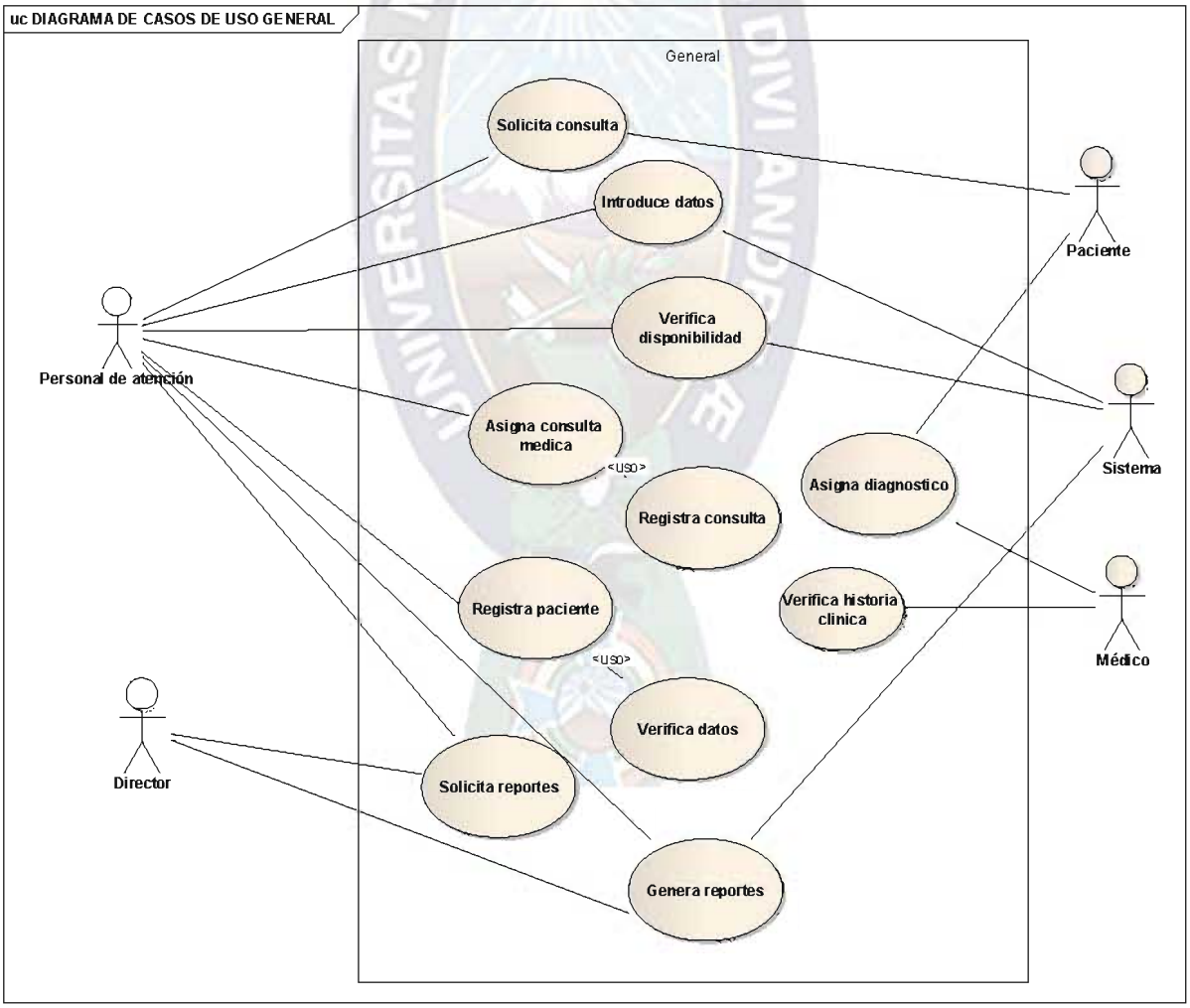


Figura 3.5. Casos de Uso general

Fuente: Elaboración según [I. Jacobsob2000]

3.3.1.2. Casos de Uso Especifico

A continuación se detallan los tres casos de uso más relevantes como ser: registro de nuevo paciente, asignación de consulta médica y diagnostico medico. El resto se encuentra detallado en la parte de anexos.

Registro de nuevo paciente

CASO DE USO: Registro de nuevo paciente

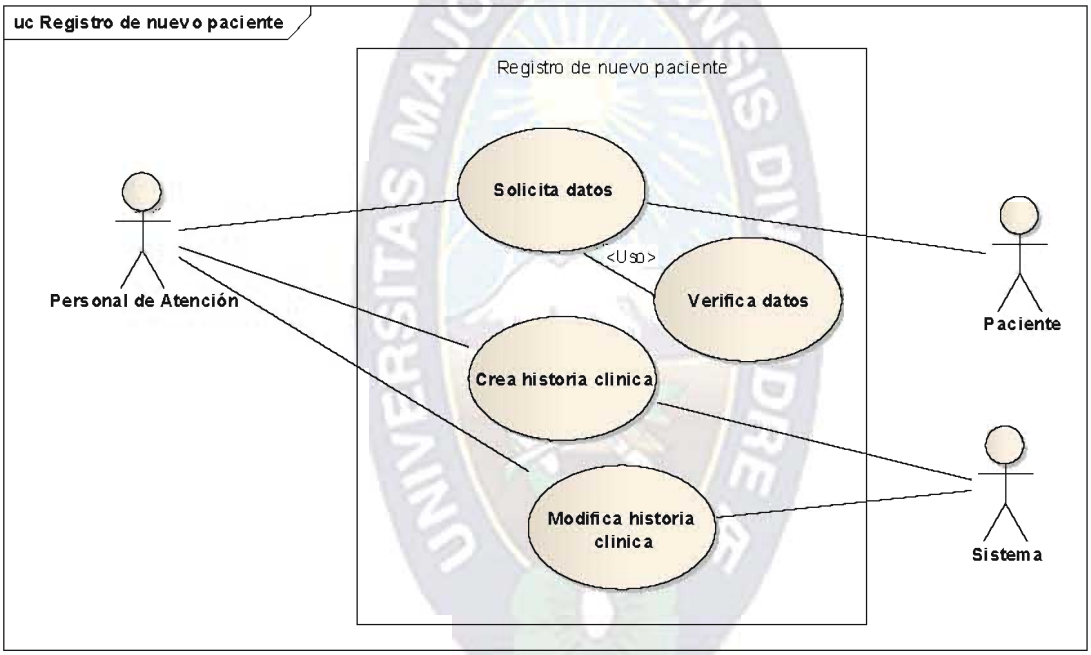


Figura 3.6. Casos de Uso de Registro de nuevo paciente

Fuente: Elaboración según [I. Jacobsob2000]

DESCRIPCIÓN DEL CASO DE USO:

Caso de uso: Registro de nuevo paciente
Descripción: El usuario del sistema puede realizar registros de nuevos pacientes y realizar modificaciones en los datos.
Actores: Usuario, personal de atención de la asistencia pública, paciente., sistema.
Requisitos: Ser usuario del sistema.

Flujo normal	Flujo alternativo
1. El paciente proporciona sus datos.	
2. El usuario ingresa al sistema e ingresa los datos del paciente.	
3. El sistema verifica si el paciente se encuentra registrado.	3.1 Si el paciente está registrado no realiza ninguna tarea
4. El usuario llena datos adicionales de registro y asigna historia clínica al paciente.	4.1 Si el paciente se encuentra registrado se puede realizar modificaciones en los datos si así se requiere.
Resultados: Registro del nuevo paciente, modificación de datos de un paciente ya existente.	

Tabla 3.1. Descripción de los Casos de Uso de registro de nuevo paciente

Fuente: Elaboración según [I. Jacobsob2000]

Asignación de consulta médica

CASO DE USO: Asignación de consulta médica

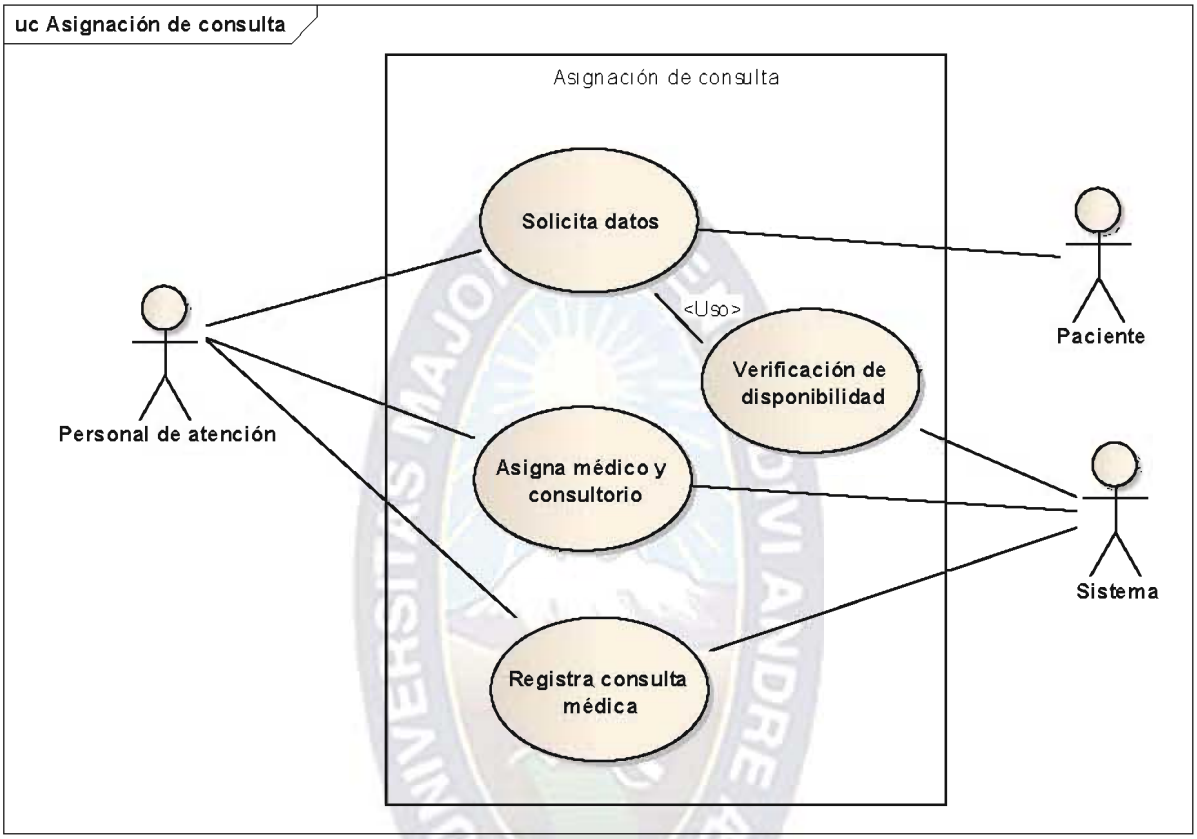


Figura 3.7. Casos de Uso de Asignación de consulta

Fuente: Elaboración según [I. Jacobsob2000]

DESCRIPCIÓN DEL CASO DE USO:

Caso de uso: Asignación de consulta médica	
Descripción: El usuario del sistema puede realizar la asignación de pacientes a las consultas médicas	
Actores: Paciente, personal de atención, sistema.	
Requisitos: Ser un usuario del sistema	
Flujo normal	**Flujo alternativo**

1. El paciente proporciona datos y especifica el tipo de atención que requiere	
2. El personal de atención ingresa satos del paciente y tipo de atención.	
3. El sistema verifica la disponibilidad de consulta.	3.1 Si no hay disponibilidad se asigna consulta por la tarde o para el día siguiente.
4. El personal de atención asigna medico y consultorio	
5. El sistema registra la consulta médica.	
Resultados: Asignar consulta a los pacientes.	

Tabla 3.2. Descripción de los Casos de Uso de asignación de consulta

Fuente: Elaboración según [I. Jacobsob2000]

Diagnóstico médico

CASO DE USO: Diagnóstico médico

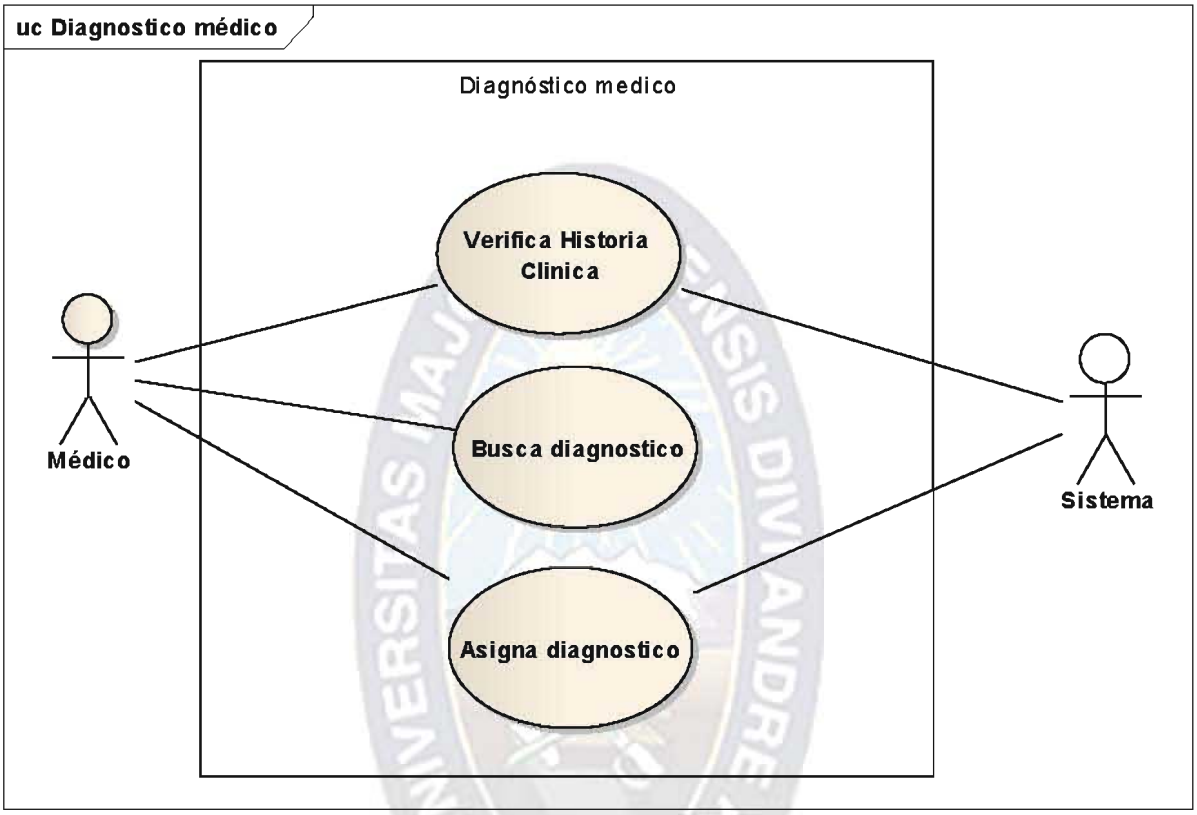


Figura 3.8. Casos de Uso de Diagnóstico médico

Fuente: Elaboración según [I. Jacobsob2000]

DESCRIPCIÓN DEL CASO DE USO:

Caso de uso: Diagnóstico médico	
Descripción: El usuario del sistema puede asignar el diagnostico medico al paciente.	
Actores: Médico, sistema.	
Requisitos: Ser un usuario del sistema	
Flujo normal	**Flujo alternativo**
1. El sistema muestra la	

cantidad de pacientes por atender en el turno.	
2. El médico verifica historia clínica del paciente en consultorio.	2.1 Si el paciente no se encuentra en sala para la consulta el médico lo registra como no atendido.
3. El médico revisa al paciente.	
4. El médico ingresa diagnostico.	
5. El sistema registra en la historia clínica correspondiente el diagnostico asignado por el médico.	
Resultados: Asignar diagnostico al paciente.	

Tabla 3.3. Descripción de los Casos de Uso de diagnostico medico
Fuente: Elaboración según [I. Jacobsob2000]

3.4. DIAGRAMAS DE ACTIVIDAD

A continuación se detallan los tres diagramas de actividades más relevantes como ser: registro de nuevo paciente, asignación de consulta médica y diagnostico medico. El resto se encuentra detallado en la parte de anexos.

3.4.1. Registro de nuevo paciente

En el siguiente diagrama se describe el conjunto de actividades para el registro de nuevos pacientes y la modificación de los datos de pacientes ya existentes.

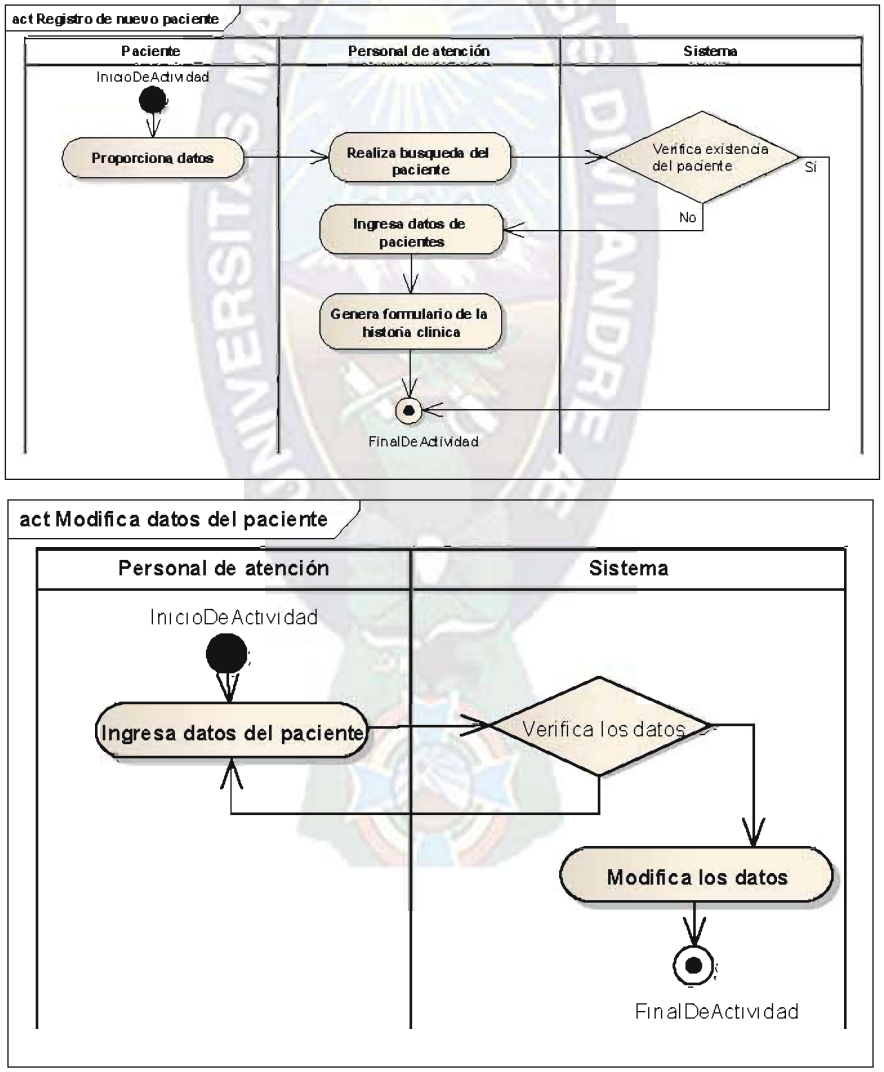


Figura 3.9. Diagrama de actividad registra y modifica paciente

Fuente: Elaboración según [I. Jacobsob2000]

3.4.2. Asignación de consulta médica

En el siguiente diagrama se describe el conjunto de actividades para realizar la asignación de consulta al paciente que lo solicite, previa verificación de la disponibilidad.

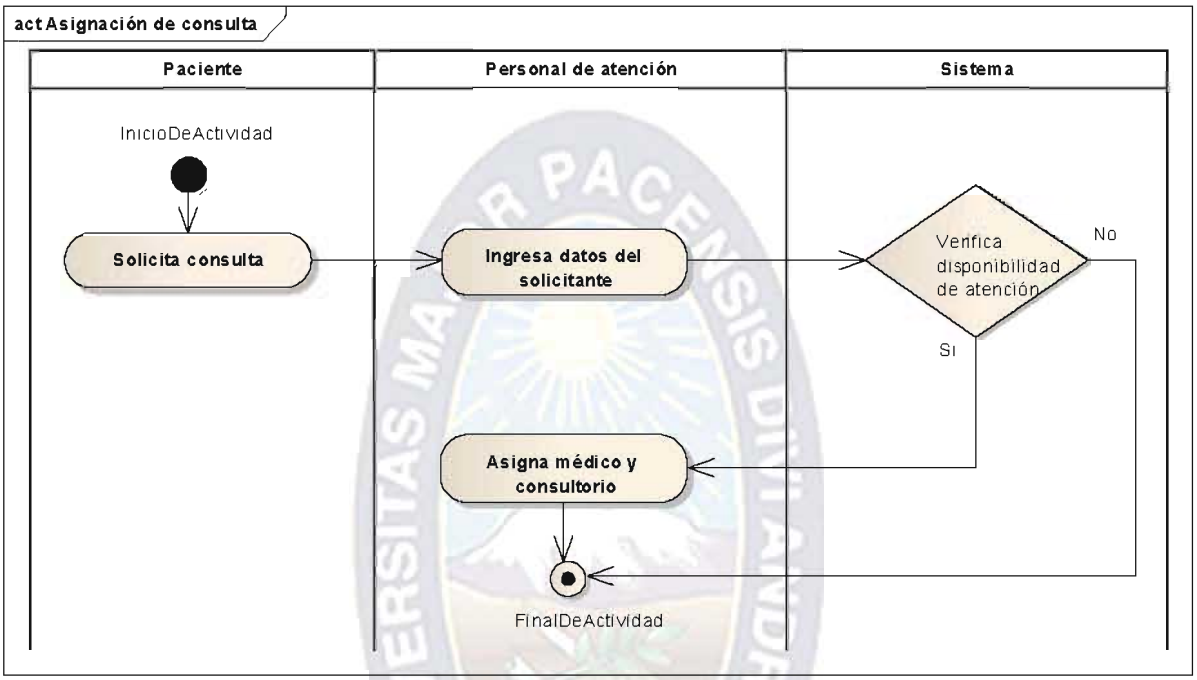


Figura 3.10. Diagrama de actividad de asignación de consulta médica

Fuente: Elaboración según [I. Jacobsob2000]

3.4.3. Diagnóstico médico

En el siguiente diagrama se describe el conjunto de actividades que realiza el médico interactuando con el sistema para asignar diagnostico medico al paciente que se encuentra en consultorio.

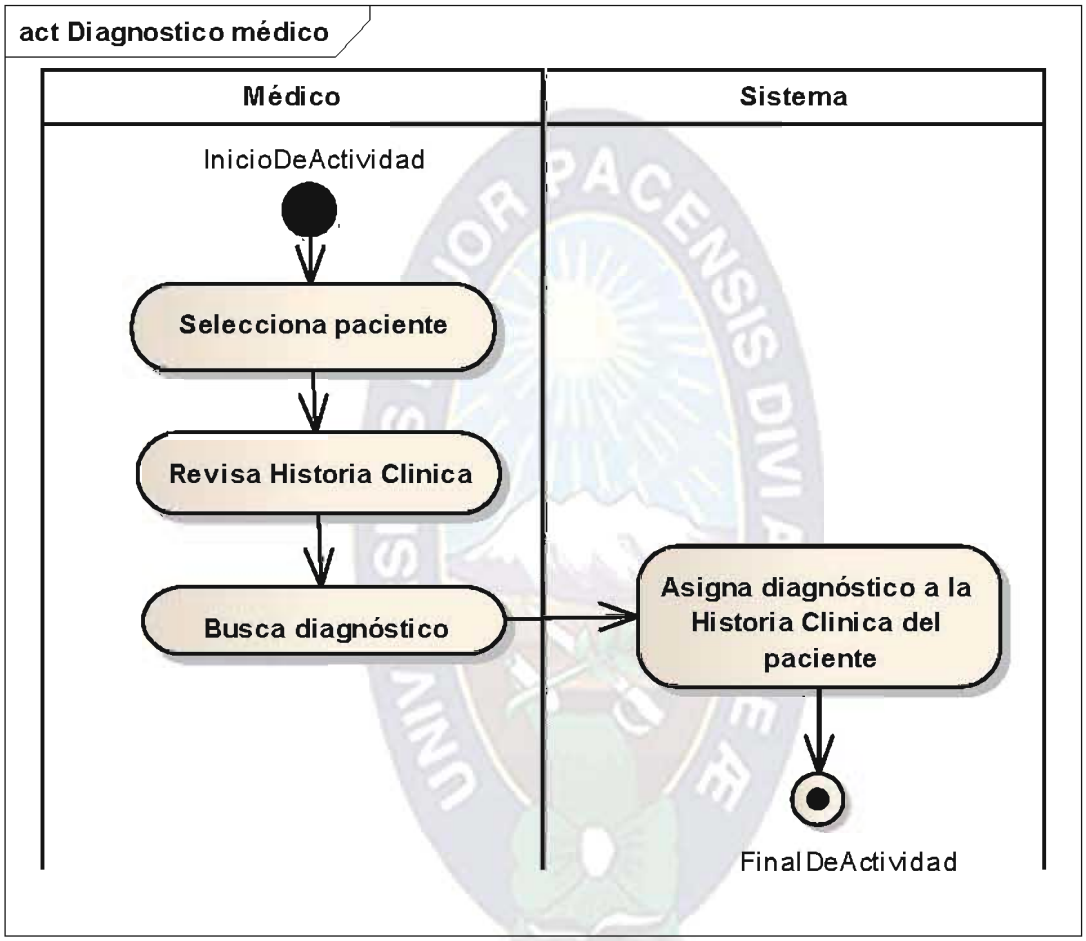


Figura 3.11. Diagrama de actividad de diagnostico medico

Fuente: Elaboración según [I. Jacobsob2000]

3.5. DIAGRAMA DE CLASES

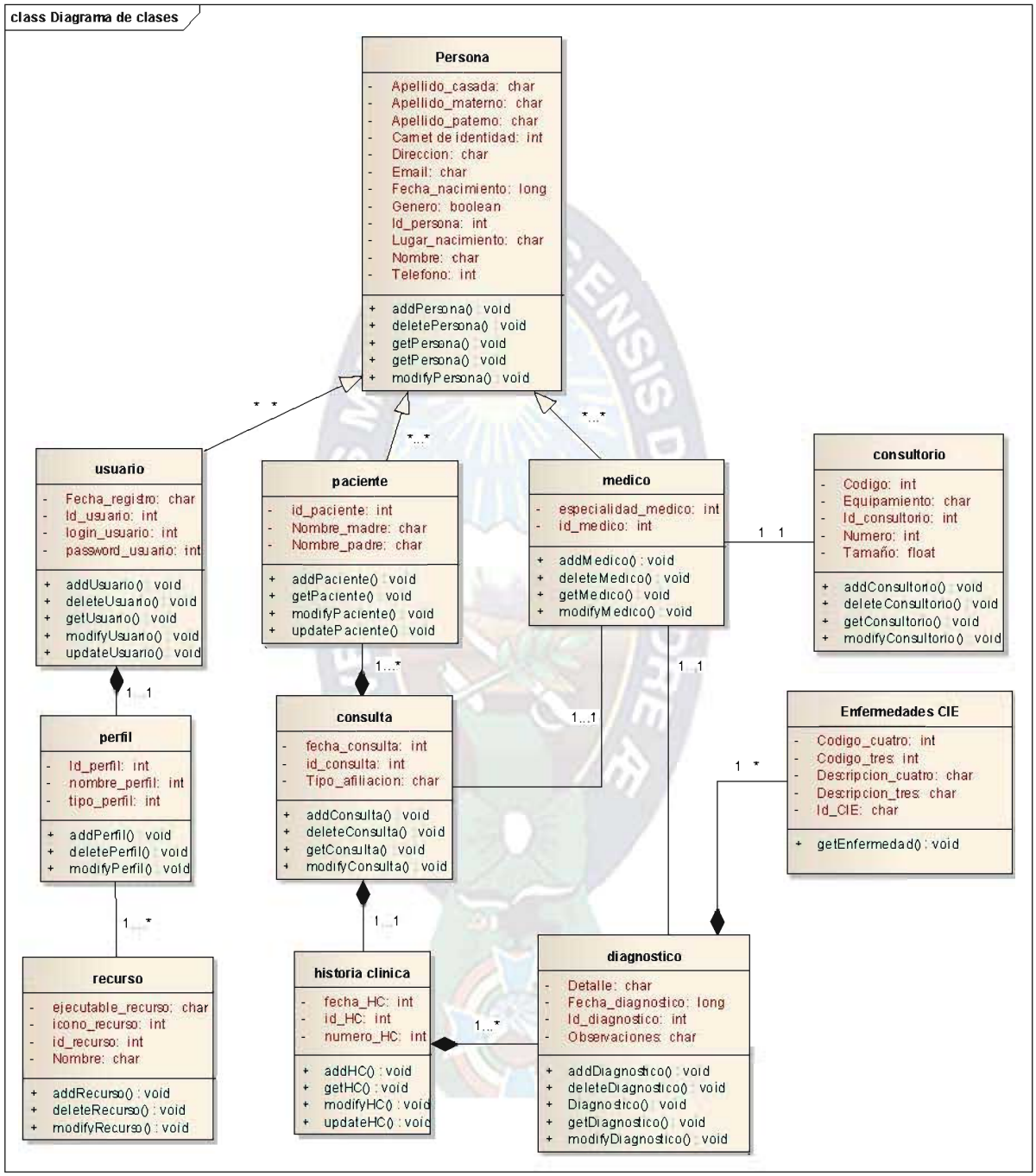


Figura 3.12. Diagrama de clases

Fuente: Elaboración según [I. Jacobsob2000]

3.6. MODELO ENTIDAD RELACIÓN

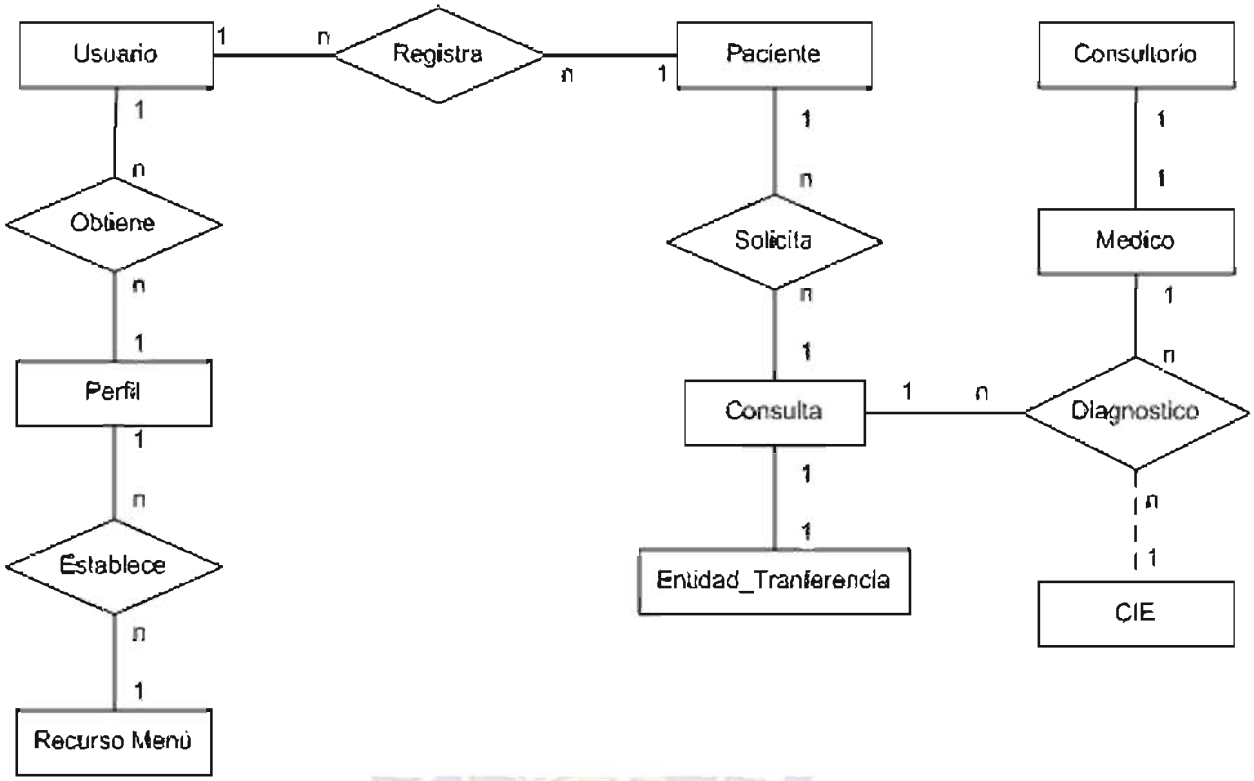


Figura 3.13. Diagrama entidad relación

Fuente: Elaboración propia

3.7. MODELO RELACIONAL DE TABLAS

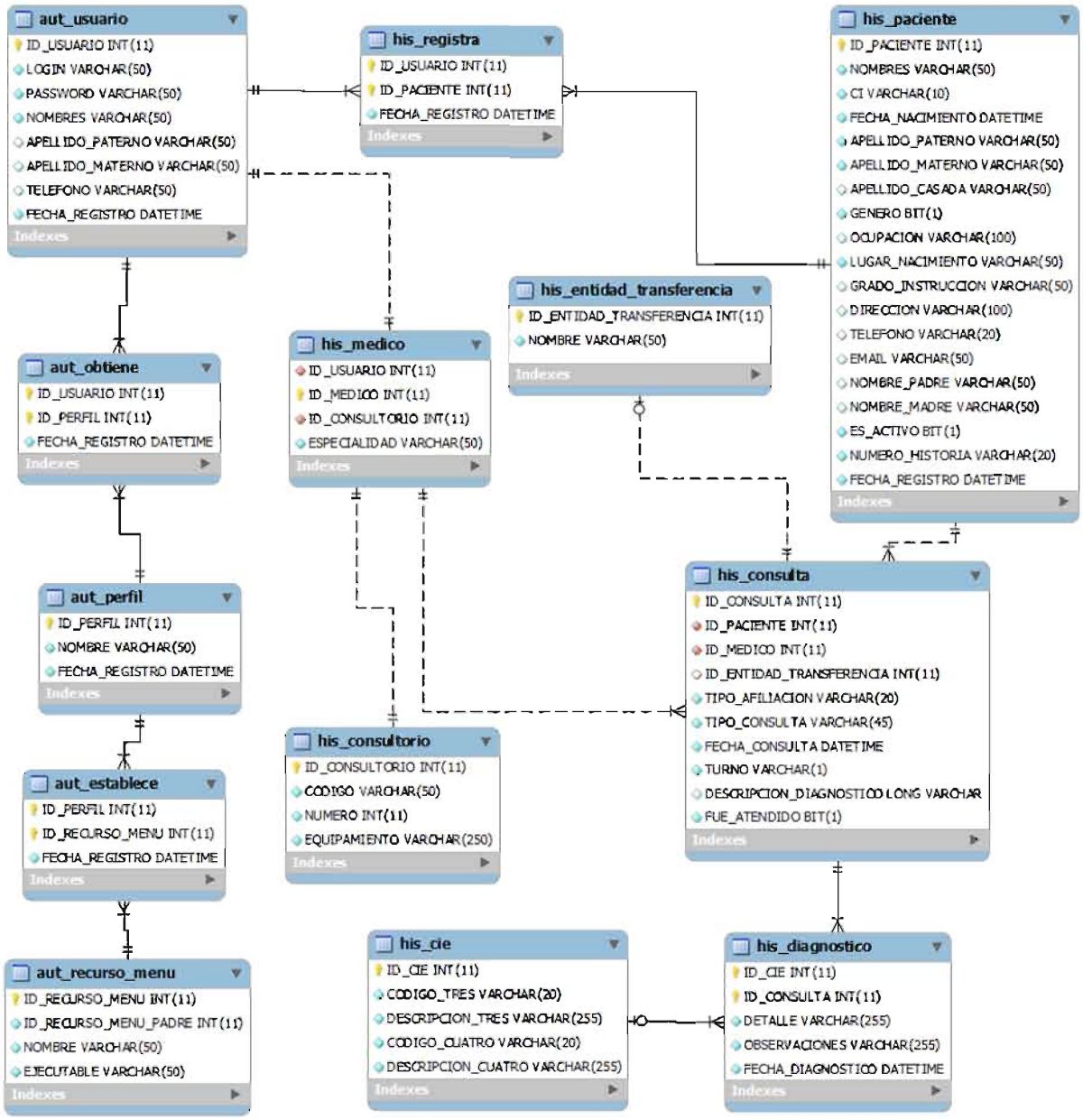


Figura 3.14. Diagrama relacional

Fuente: Elaboración propia

3.8. ARQUITECTURA DEL SISTEMA

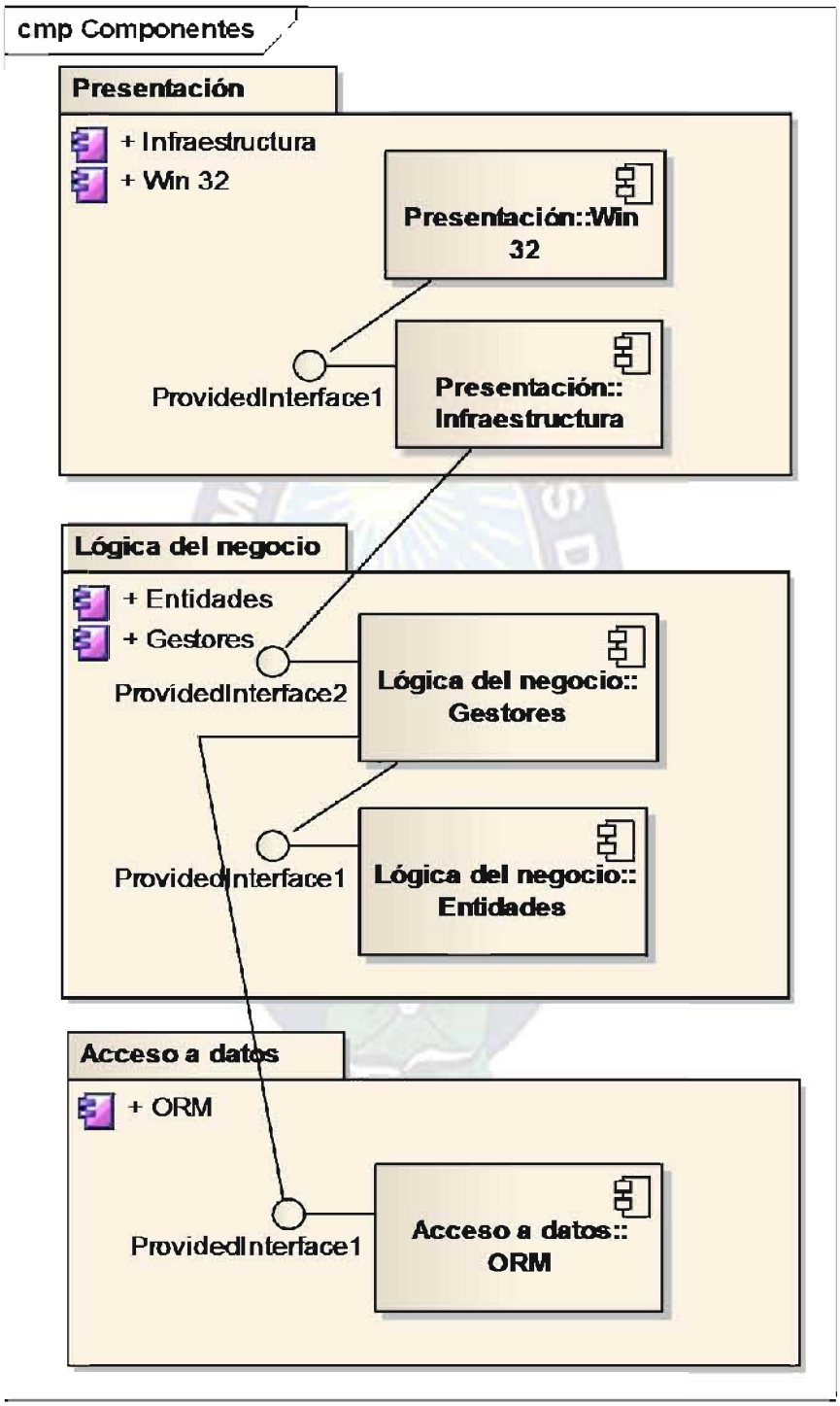


Figura 3.15. Diagrama de la arquitectura del sistema

Fuente: Elaboración según [I. Jacobsob2000]

3.9. DESARROLLO DEL AGENTE

En esta sección del documento se presenta el modelado de los agentes de búsqueda y de colaboración, aplicando la metodología orientada a agentes MaSE.

Como se menciona en el marco teórico la metodología MaSE consta de las siguientes etapas para el análisis y modelado de los agentes:

- Construcción de la tabla de percepciones, acciones, metas y ambiente (PAMA).
- Captura de objetivos (diagrama de jerarquía de objetivos).
- Aplicación de diagramas UML (casos de uso, diagramas de secuencia y diagramas de despliegue).
- Refinamiento de roles.
- Refinamiento de roles detallado.
- Creación de clases de agentes
- Construcción de conversaciones.
- Despliegue final del sistema.

3.9.1. Metodología orientada a agentes (MaSE)

3.9.1.1. Descripciones PAMA

N°	Tipo de agente	Percepciones	Acciones	Meta	Ambiente
A	Agente para análisis de datos del paciente	Datos del paciente(nombre, apellido paterno, apellido materno, fecha de nacimiento, nombre del padre, nombre de la madre)	Comparaciones, selecciones	Evitar la duplicidad de datos de paciente	Paciente
B	Agente para generar un identificador único para cada paciente (historia clínica).	Datos del paciente verificado (nombre, apellido paterno, apellido materno, fecha de nacimiento).	Asignación de identificador único al paciente.	Generar un identificador único de historia clínica.	Paciente

Tabla 3.4. Tabla de la arquitectura del agente

Fuente: Elaboración propia

3.9.1.2. Captura de objetivos

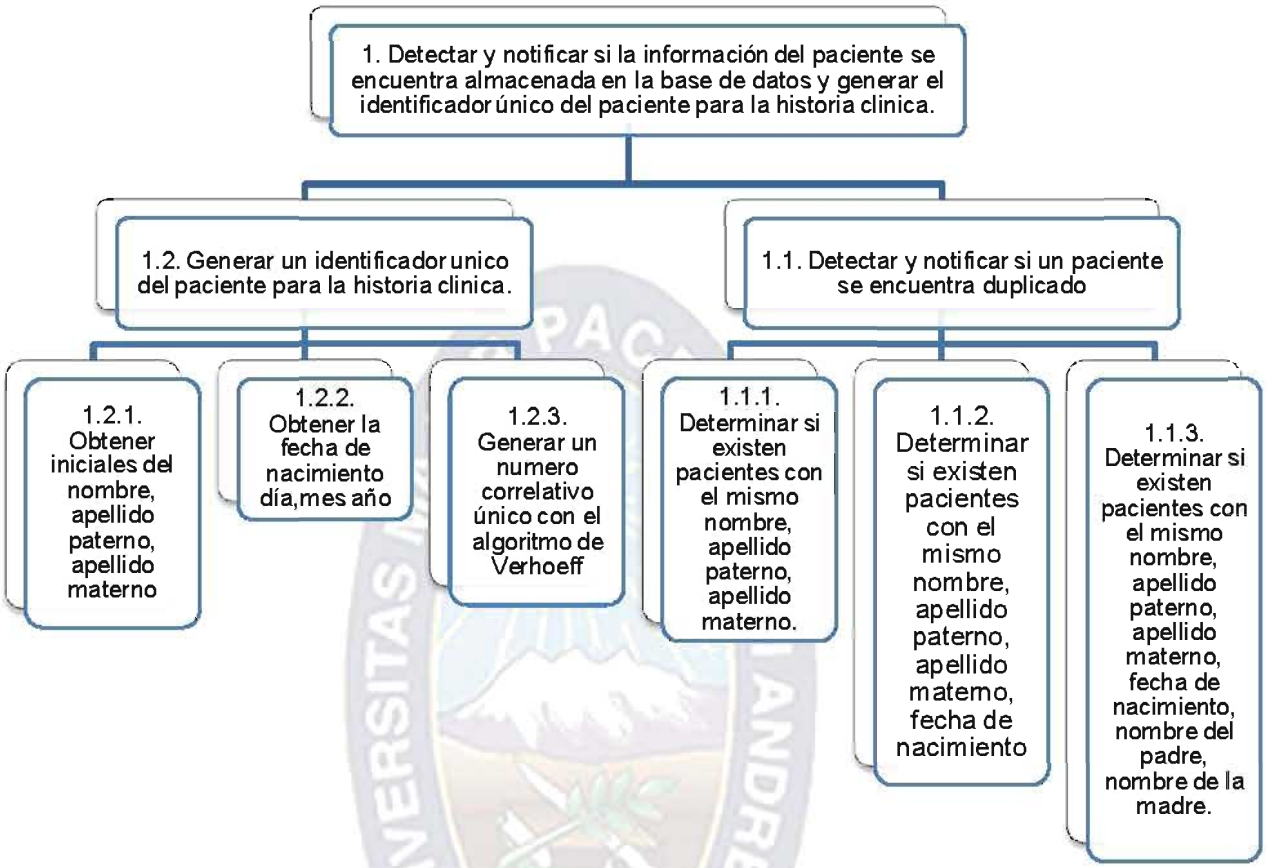


Figura 3.16. Diagrama de jerarquía de objetivos en MaSE

Fuente: Elaboración según [Gallego2004]

3.9.1.3. Aplicación de diagramas UML

Casos de uso del agente

A. Agente para análisis de datos del paciente

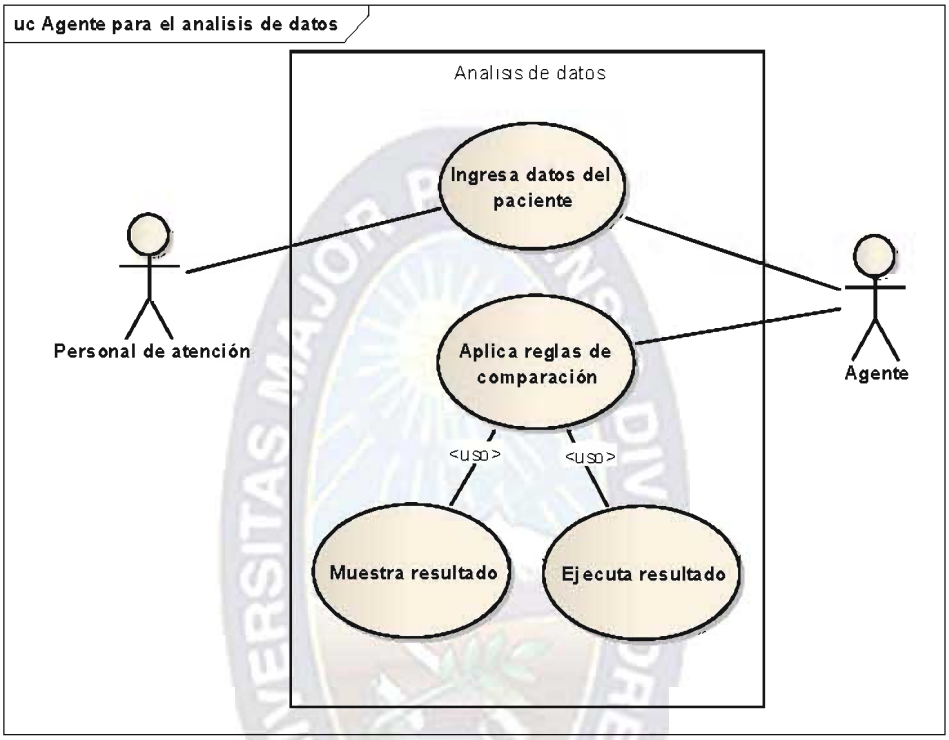


Figura 3.17. Diagrama de Casos de uso del agente de análisis de datos

Fuente: Elaboración según [I. Jacobsob2000]

B. Agente para generar un identificador único (historia clínica)

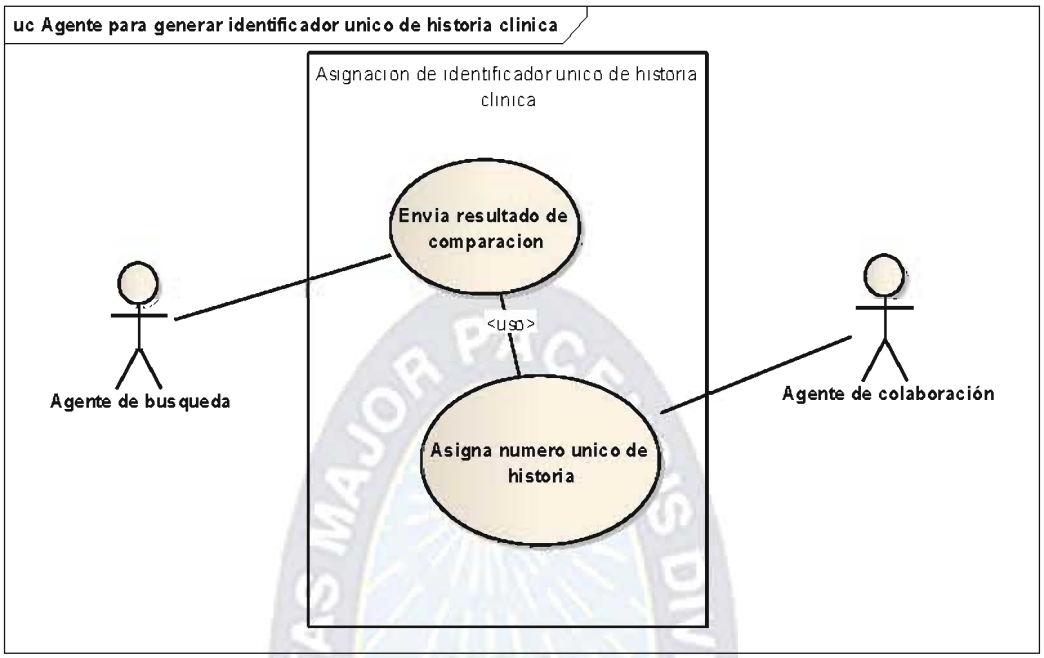


Figura 3.18. Diagrama de Casos de uso del agente de asignación de identificador único de historia clínica

Fuente: Elaboración según [I. Jacobsob2000]

Diagrama de secuencias del agente

A. Agente para análisis de datos del paciente

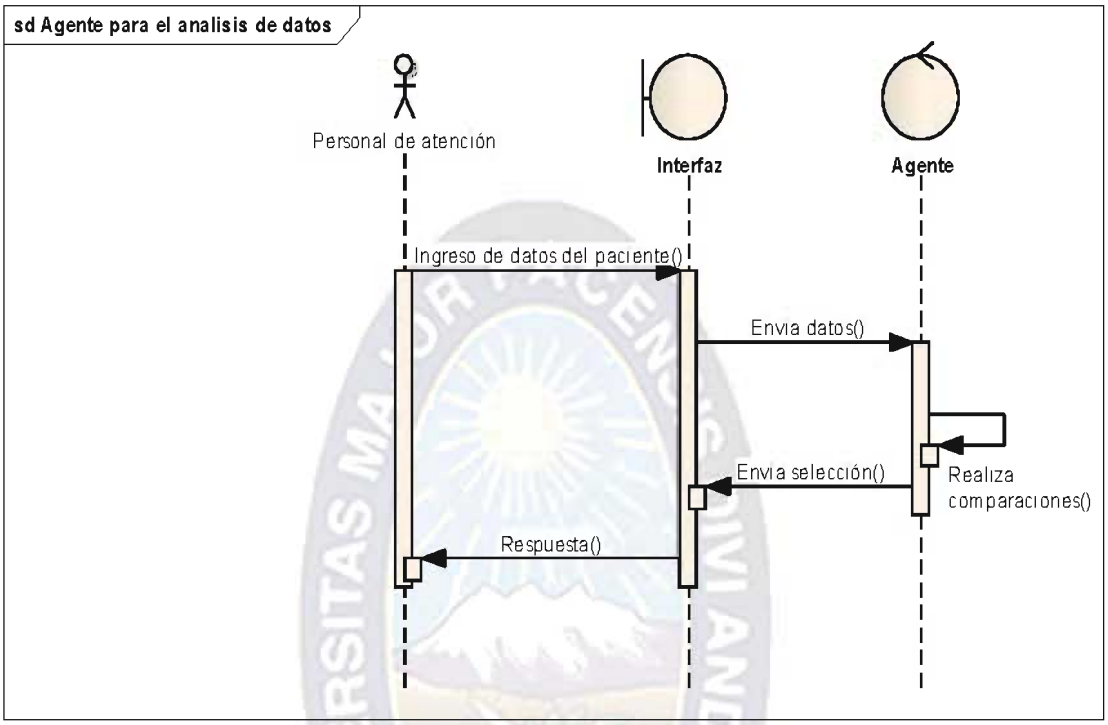


Figura 3.19. Diagrama de secuencias del agente de análisis de datos

Fuente: Elaboración según [I. Jacobsob2000]

B. Agente para generar un identificador único (historia clínica)

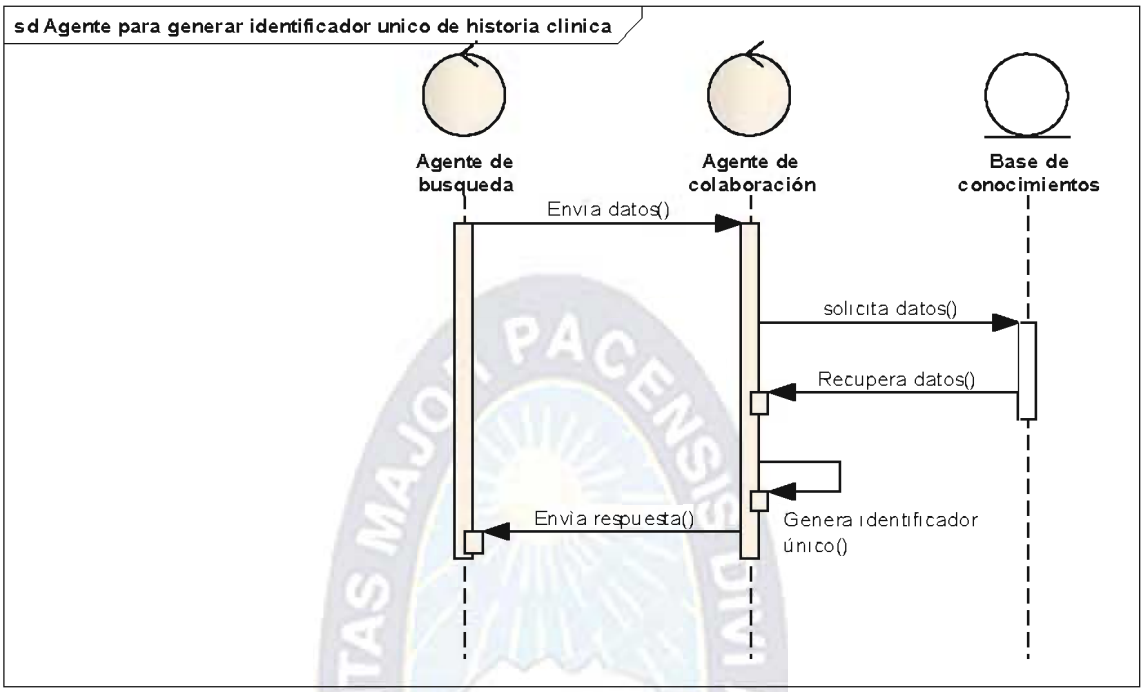


Figura 3.20. Diagrama de secuencia del agente de asignación de identificador único de historia clínica

Fuente: Elaboración según [I. Jacobsob2000]

3.9.1.4. Refinamiento de roles

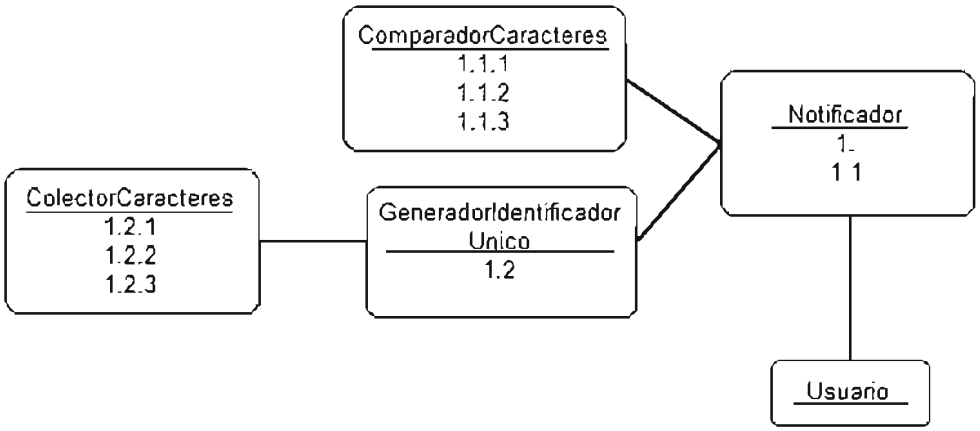


Figura 3.21. Modelo de Roles

Fuente: Elaboración según [Gallego2004]

3.9.1.5. Refinamiento de roles detallado

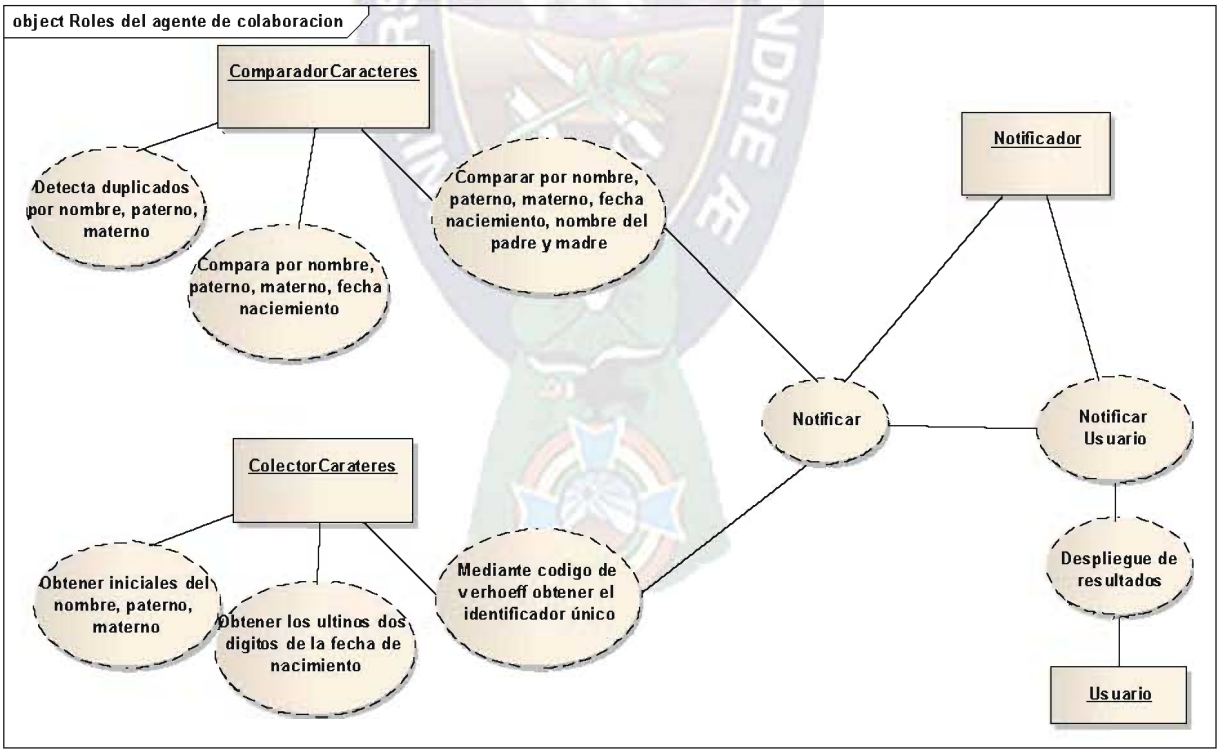


Figura 3.22. Modelo de Roles Detallado

Fuente: Elaboración según [Gallego2004]

3.9.1.6. Creación de las clases de agente

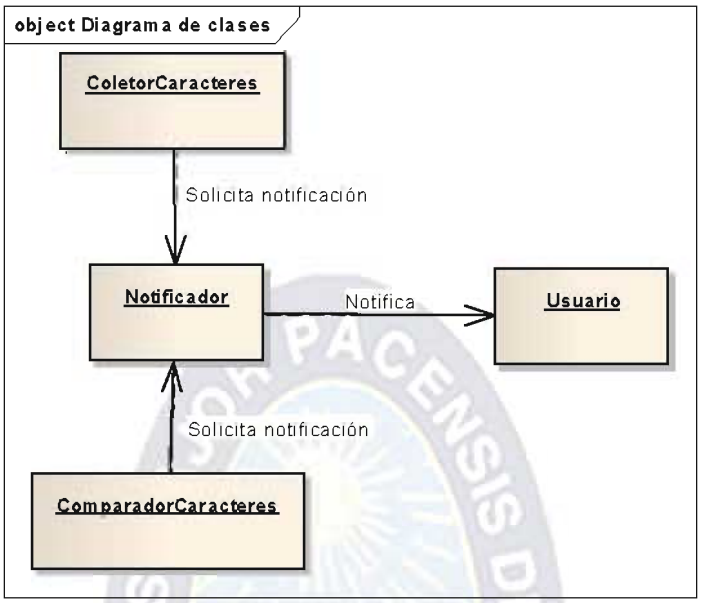


Figura 3.23. Diagrama de Clases en MaSE

Fuente: Elaboración según [Gallego2004]

3.9.1.7. Ensamblaje del agente (arquitectura)

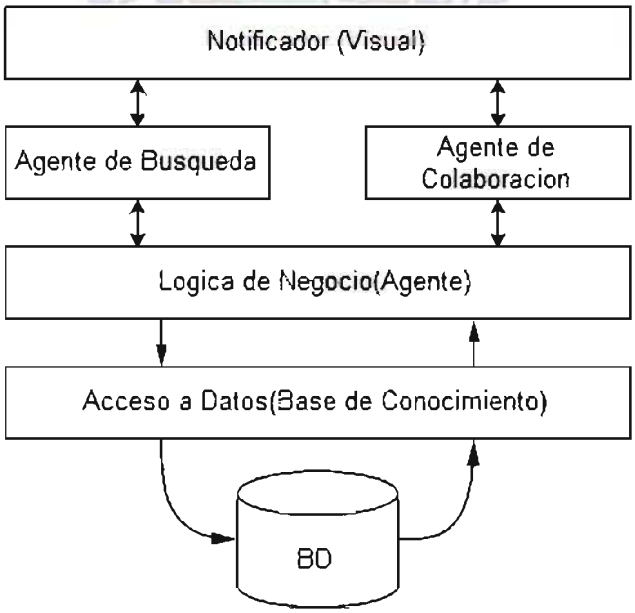


Figura 3.24. Arquitectura de los agentes

Fuente: Elaboración según [Pressman2010]

3.9.1.8. Despliegue final del sistema

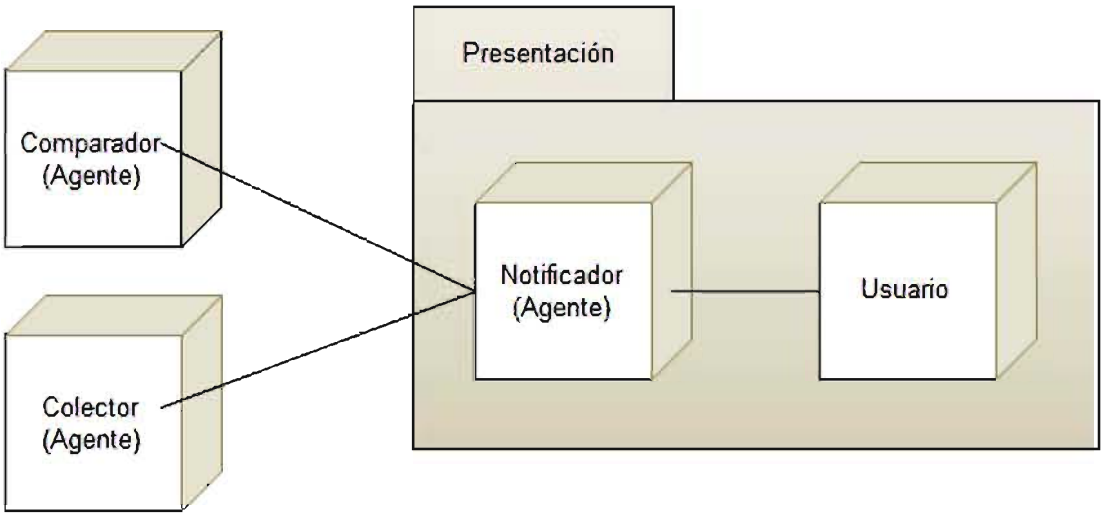


Figura 3.25. Diagrama de despliegue

Fuente: Elaboración según [I. Jacobsob2000]

3.10. SEGUIMIENTO AL PROYECTO CON SCRUM

3.10.1. Historias

Pila de productos (Inicial)		
Código	**Elemento de pila del producto**	**Prioridad**
1	Se requiere múltiples niveles de acceso al sistema, donde se asignen opciones de menú de acuerdo al rol que corresponda	10
2	Se requiere dar de alta usuarios al sistema y clasificarlos en usuarios de administración, generación de diagnósticos y consulta	15

3	Registro de pacientes en el sistema (considerar la existencia de registros anteriores para evitar duplicidad de datos)	25
4	Modificar datos de pacientes existentes (considerar la baja pacientes)	30
5	Asignaciones de consultas a pacientes que lo requieran, por orden de llegada	35
6	Anulación de consultas (considerar reorganización de tiempos)	40
7	Emisión de diagnósticos clínicos por parte de los médicos especialistas	45
8	Emisión de reportes para el personal administrativo de la Asistencia Pública	50
9	Emisión de reportes a demanda para el SNIS	55
10	Arquitectura flexible para posteriores implementaciones (pensar en que posteriormente se trabaje en un escenario Web)	5
11	Migración y consolidación de datos de pacientes e historias clínicas al sistema	20

Tabla 3.5. Tabla de historias de usuario

Fuente: Elaboración propia

3.10.2. Pila de productos(sprint)

Pila de productos (para sprint)				
Código	**Elemento de pila del producto**	**Prioridad**	**Criterios de verificación**	**Observaciones**
10	Arquitectura flexible para posteriores implementaciones (pensar en que posteriormente se trabaje en un escenario Web)	5	Verificar arquitectura del sistema utilizando la herramienta visual que se usó para el desarrollo de la etapa inicial del sistema	Se debe plantear un diagrama de componentes de UML para esquematizar la arquitectura
1	Se requiere múltiples niveles de acceso al sistema, donde se asignen opciones de menú de acuerdo al rol que corresponda	10	Generar roles en el sistema y asignarles las opciones de menú que se desee	
2	Se requiere dar de alta usuarios al sistema y clasificarlos en usuarios de administración, generación de diagnósticos y consulta	15	Ingresar usuarios nuevos al sistema y asignarles roles para el manejo del mismo	
11	Migración y consolidación de datos de pacientes e historias clínicas al sistema	20	Verificar datos consolidados con consultas a la base de datos	
3	Registro de pacientes en el sistema (considerar la existencia de registros anteriores para evitar duplicidad de datos)	25	Registrar pacientes en el sistema, verificar que no se ingresen pacientes duplicados en base a su documentación	Se debe plantear diagramas de casos de uso y actividad para su mejor compresión

4	Modificar datos de pacientes existentes (considerar la baja pacientes)	30	Dar de baja un paciente y verificar que no sea tomado en cuenta para las tareas que realiza el sistema	
5	Asignaciones de consultas a pacientes que lo requieran, por orden de llegada	35	Asignar una consulta a un paciente "vigente"	Se debe plantear diagramas de casos de uso y actividad para su mejor compresión
6	Anulación de consultas (considerar reorganización de tiempos)	40	Anular la consulta de un paciente y verificar que ese espacio sea asignado a sucesor	
7	Emisión de diagnósticos clínicos por parte de los médicos especialistas	45	Emitir diagnósticos clínicos para un paciente seleccionado, verificar que esos datos sean almacenados correctamente	Se debe plantear diagramas de casos de uso y actividad para su mejor compresión
8	Emisión de reportes para el personal administrativo de la Asistencia Pública	50	Emitir reportes con información de prueba	
9	Emisión de reportes a demanda para el SNIS	55	Emitir reportes con información de prueba	

Tabla 3.6. Tabla de la pila de productos

Fuente: Elaboración propia

BACKLOG # 1 (SPRINT 1)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
10	Arquitectura flexible para posteriores implementaciones (pensar en que posteriormente se trabaje en un escenario Web)	5	Plantear un diagrama de componentes para la arquitectura	0.5
			Plantear las capas lógicas para que el sistema sea escalable	1
			Organizar la arquitectura del sistema en base a una herramienta de programación	2.5
			Desarrollar las clases y métodos base para el sistema	4
Tiempo estimado para el desarrollo del backlog				8

BACKLOG # 2 (SPRINT 1)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
1	Se requiere múltiples niveles de acceso al sistema, donde se asignen opciones de menú de acuerdo al rol que corresponda	10	Implementar clases de negocio para la administración de roles y recursos de menú	2
			Implementar test unitarios para las clases de negocio implementadas	1

			Desarrollar interfaces para la administración de roles y recursos de menú	2
			realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

BACKLOG # 3 (SPRINT 1)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
2	Se requiere dar de alta usuarios al sistema y clasificarlos en usuarios de administración, generación de diagnósticos y consulta	15	Implementar clases de negocio para la administración usuarios del sistema	2
			Implementar test unitarios para las clases de negocio implementadas	1
			Desarrollar interfaces para la administración usuarios y asignación de roles	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

BACKLOG # 4 (SPRINT 2)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
11	Migración y consolidación de datos de pacientes e historias clínicas al sistema	20	Diseño del agente para la selección y verificación de los datos del paciente e historias clínicas	2
			Implementar lógica de negocio del agente	2
			Test unitarios de los métodos implementados	1
			Desarrollar interface de usuario para trabajar con el agente	2
			Realizar pruebas funcionales de las pantallas del agente	1
Tiempo estimado para el desarrollo del backlog				8

BACKLOG # 5 (SPRINT 2)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
3	Registro de pacientes en el sistema (considerar la existencia de registros anteriores para evitar duplicidad de datos)	25	Implementar clases de negocio para el registro de nuevos pacientes al sistema de gestión de historias clínicas	2

			Implementar test unitarios para las clases de negocio implementadas	1
			Desarrollar interfaces para el registro de nuevos pacientes al sistema	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

BACKLOG # 6 (SPRINT 2)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
4	Modificar datos de pacientes existentes (considerar la baja pacientes)	30	Implementar clases de negocio para la modificación de pacientes al sistema de gestión de historias clínicas	2
			Implementar métodos que permitan la baja y la habilitación de pacientes	1
			Implementar test unitarios para las clases de negocio implementadas	1

			Desarrollar interfaces para la baja y habilitación de pacientes	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				7

BACKLOG # 7 (SPRINT 3)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
5	Asignaciones de consultas a pacientes que lo requieran, por orden de llegada	35	Implementar clases de negocio para la asignación de consultas (considerar horarios de atención)	2
			Implementar test unitarios para las clases de negocio implementadas	1
			Desarrollar interfaces para la asignación de consultas	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

BACKLOG # 8 (SPRINT 3)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
6	Anulación de consultas (considerar reorganización de tiempos)	40	Implementar clases de negocio para la anulación consultas (considerar reasignación de horarios a pacientes)	2
			Implementar test unitarios para las clases de negocio implementadas	1
			Desarrollar interfaces la anulación de consultas	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

BACKLOG # 9 (SPRINT 3)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
7	Emisión de diagnósticos clínicos por parte de los médicos especialistas	45	Implementar clases de negocio para la emisión de diagnostico médico.	2
			Implementar test unitarios para las clases de negocio	1

			implementadas	
			Desarrollar interfaces la emisión de diagnostico médico.	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

BACKLOG # 10 (SPRINT 3)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
8	Emisión de reportes para el personal administrativo de la Asistencia Pública	50	Implementar clases de negocio para la obtención de datos para los reportes para el personal administrativo.	2
			Implementar test unitarios para las clases de negocio implementadas	1
			Desarrollar interfaces la emisión de reportes.	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

BACKLOG # 11 (SPRINT 3)				
Código	**Elemento de pila del producto**	**Prioridad**	**Tareas**	**Tiempo (días)**
9	Emisión de reportes a demanda para el SNIS	55	Implementar clases de negocio para la obtención de datos para la emisión de reportes para el SNIS.	2
			Implementar test unitarios para las clases de negocio implementadas	1
			Desarrollar interfaces la emisión de reporte estadístico (SNIS).	2
			Realizar pruebas funcionales de las pantallas	1
Tiempo estimado para el desarrollo del backlog				6

Tabla 3.7. Tabla de los backlog

Fuente: Elaboración propia

3.10.3. Sprints del proyecto

3.10.3.1. Sprint 1

SPRINT 1	Código	Elemento de pila del producto	Prioridad	Tareas	Tiempo (días)	SEPTIEMBRE																					
						4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	10	Arquitectura flexible para posteriores implementaciones (pensar en que posteriormente se trabaje en un escenario Web)	5	Plantear un diagrama de componentes para la arquitectura	0.5																						
				Plantear las capas lógicas para que el sistema sea escalable	1																						
				Organizar la arquitectura del sistema en base a una herramienta de programación.	2.5																						

				Desarrollar las clases y métodos base para el sistema	4																						
	1	Se requiere múltiples niveles de acceso al sistema, donde se asignen opciones de menú de acuerdo al rol que corresponda.	10	Implementar clases de negocio para la administración de roles y recursos de menú	2																						
				Implementar test unitarios para las clases de negocio implementadas	1																						
				Desarrollar interfaces para la administración de roles y	2																						

				recursos de menú																							
				Realizar pruebas funcionales de las pantallas	1																							
	2	Se requiere dar de alta usuarios al sistema y clasificarlos en usuarios de administración, generación de diagnósticos y consulta	15	Implementar clases de negocio para la administración usuarios del sistema	2																						
				Implementar test unitarios para las clases de negocio implementadas	1																						
				Desarrollar interfaces para la administración usuarios y	2																						

				asignación de roles																									
				Realizar pruebas funcionales de las pantallas	1																								

3.10.3.2. Sprint 2

SPRINT 2	Código	Elemento de pila del producto	Prioridad	Tareas	Tiempo (días)	SEPTIEMBRE					OCTUBRE																			
						26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		
	11	Migración y consolidación de datos de pacientes e historias clínicas al sistema	20	Diseño del agente para la selección y verificación de los datos del paciente e historias clínicas	2																									
				Implementar lógica de negocio del agente	2																									
				Test unitarios de los métodos implementados	1																									
				Desarrollar interface de usuario para trabajar con el	2																									

				agente																									
				Realizar pruebas funcionales de las pantallas del agente	1																								
	3	Registro de pacientes en el sistema (considerar la existencia de registros anteriores para evitar duplicidad de datos)	25	Implementar clases de negocio para el registro de nuevos pacientes al sistema de gestión de historias clínicas	2																								
				Implementar test unitarios para las clases de negocio implementadas	1																								

				Desarrollar interfaces para el registro de nuevos pacientes al sistema	2																							
				realizar pruebas funcionales de las pantallas	1																							
	4	Modificar datos de pacientes existentes (considerar la baja pacientes)	30	Implementar clases de negocio para la modificación de pacientes al sistema de gestión de historias clínicas	2																							
				Implementar métodos que permitan la baja y la	1																							

				habilitación de pacientes																										
				Implementar test unitarios para las clases de negocio implementadas	1																									
				Desarrollar interfaces para la baja y habilitación de pacientes	2																									
				Realizar pruebas funcionales de las pantallas	1																									

3.10.3.3. Sprint 3

SPRINT 3	Código	Elemento de pila del producto	Prioridad	Tareas	Tiempo (días)	OCTUBRE									NOVIEMBRE																									
						24	25	26	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	5	Asignaciones de consultas a pacientes que lo requieran, por orden de llegada	35	Implementar clases de negocio para la asignación de consultas (considerar horarios de atención)	2																																			
				Implementar test unitarios para las clases de negocio implementadas	1																																			
				Desarrollar interfaces para la asignación de consultas	2																																				
				Realizar pruebas funcionales de las pantallas	1																																				

	6	Anulación de consultas (considerar reorganización de tiempos)	40	Implementar clases de negocio para la anulación consultas (considerar reasignación de horarios a pacientes)	2																																				
				Implementar test unitarios para las clases de negocio implementadas	1																																				
				desarrollar interfaces la anulación de consultas	2																																				
				realizar pruebas funcionales de las pantallas	1																																				
	7	Emisión de diagnósticos clínicos por parte de los médicos especialistas	45	Implementar clases de negocio para la emisión de diagnostico médico	2																																				

				Implementar test unitarios para las clases de negocio implementadas	1
				Desarrollar interfaces la emisión de diagnostico médico	2
				Realizar pruebas funcionales de las pantallas	1
	8	Emisión de reportes para el personal administrativo de la Asistencia Pública	50	Implementar clases de negocio para la obtención de datos para los reportes para el personal administrativo	2
				Implementar test unitarios para las clases de negocio	1

				implementadas	
				Desarrollar interfaces la emisión de reportes	2
				Realizar pruebas funcionales de las pantallas	1
	9	Emisión de reportes a demanda para el SNIS	55	Implementar clases de negocio para la obtención de datos para la emisión de reportes para el SNIS	2
				Implementar test unitarios para las clases de negocio implementadas	1
				Desarrollar interfaces la emisión de reporte estadístico (SNIS)	2

				Realizar pruebas funcionales de las pantallas	1	

3.11. INTERFAZ DE USUARIO

Figura 3.26. Autenticación del usuario

En esta pantalla se tiene dos campos que valida los datos introducidos, en caso de ser errados los datos abrirá un mensaje de error.

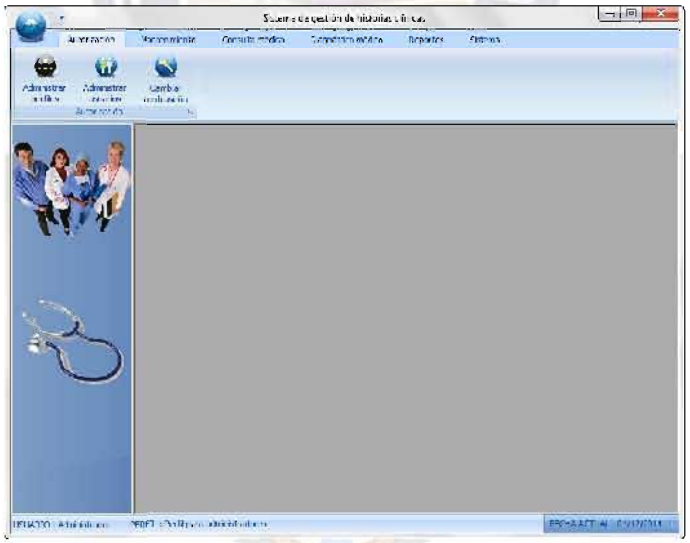


Figura 3.27. Pantalla principal

Una vez autenticado el usuario, ingresa a la pantalla principal, donde se puede encontrar el menú en pestañas según el tipo de usuario.

Figura 3.28. Pestaña de autorización

En la pestaña de autorización se tiene tres opciones, administrar perfiles, administrar usuarios y cambiar la contraseña de usuario. En administrador de perfiles se encuentra la opción de asignar un perfil a los usuarios según sus permisos; en administrador de usuarios se tiene la opción de registrar nuevos usuarios al sistema y en cambiar contraseña la opción de que el usuario cambie su contraseña de ingreso al sistema.

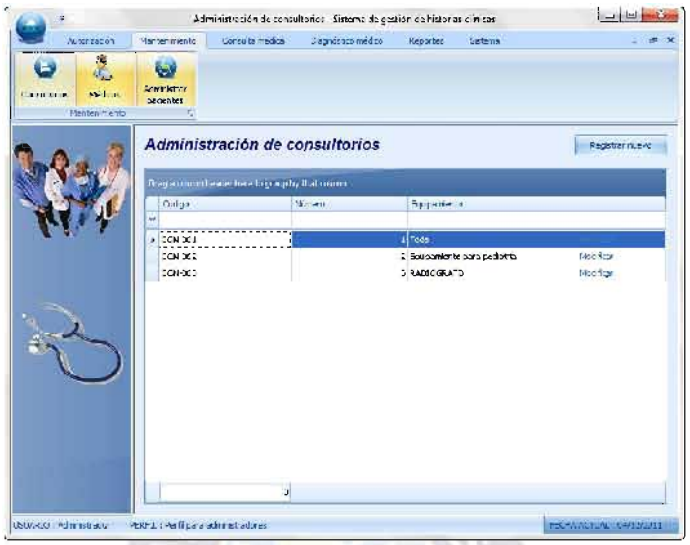


Figura 3.29. Pestaña mantenimiento

Se tiene las opciones de administrar consultorios, médicos y pacientes; con sus respectivas opciones de nuevo y modificación.

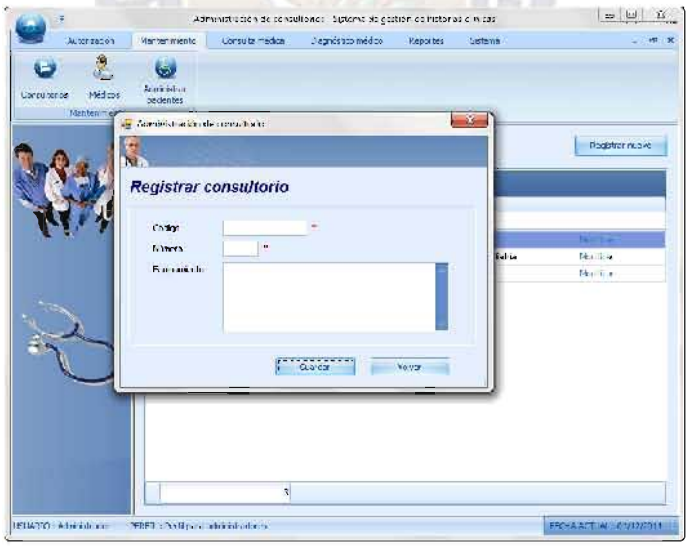


Figura 3.30. Registro de nuevo consultorio

En la pantalla principal de mantenimiento se encuentra el botón de consultorio que muestra la lista de consultorios registrados. En caso de requerir habilitar uno se

selecciona el botón registrar nuevo que nos muestra la pantalla de registro de consultorio.

Figura 3.31. Registro de médicos

Seleccionando el botón de médicos, se muestra la lista de médicos registrados con la opción de modificar, en el caso de habilitar un nuevo medico seleccionamos el botón de registrar nuevo que muestra una nueva pantalla donde se registra al nuevo médico asignándole un consultorio.

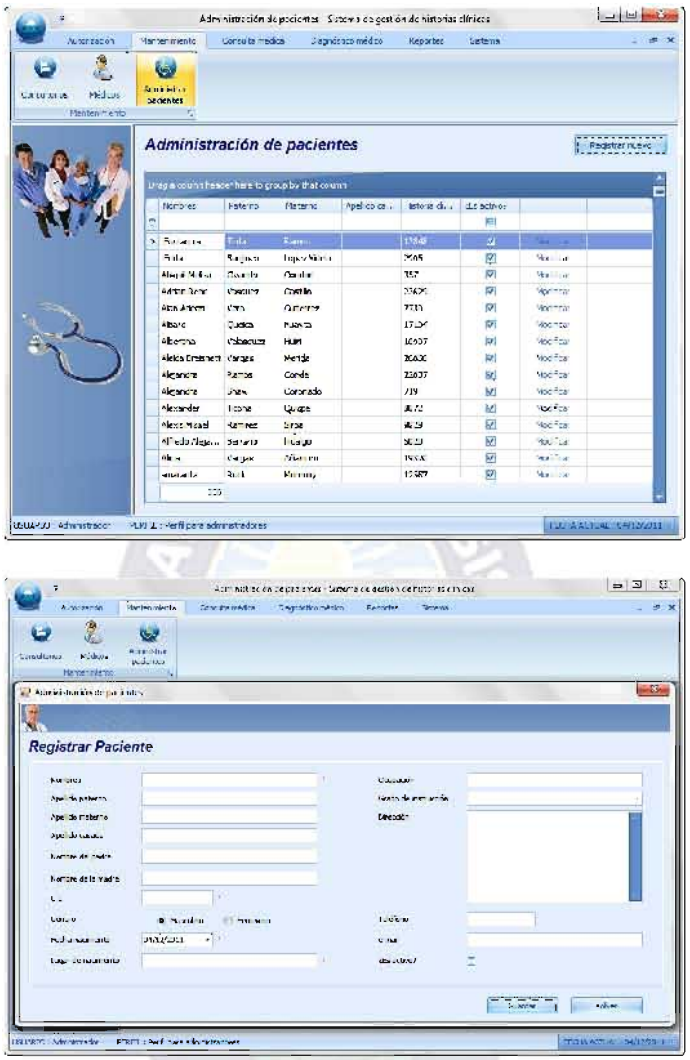


Figura 3.32. Administrar paciente

Al seleccionar el botón de administrar pacientes, el usuario puede ver una lista de todos los pacientes registrados realizar modificaciones o simplemente registro de nuevo paciente. En la pantalla de registrar paciente se introduce los datos y luego se selecciona guardar, volver para regresar al menú principal.

Figura 3.33. Pestaña consulta médica

En esta pestaña se encuentra el botón de asignación de consulta médica donde se muestra la lista de pacientes u el botón de asignar consulta.

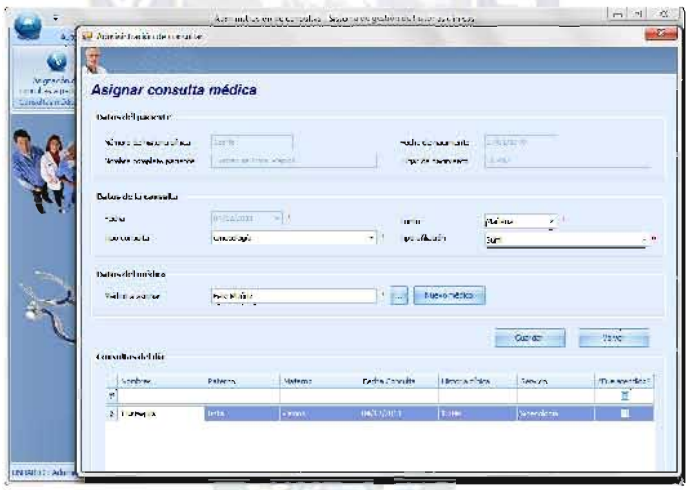


Figura 3.34. Pantalla de asignación de consulta

Esta pantalla aparece luego de seleccionar un paciente se completa los campos necesarios y luego se guarda la consulta.

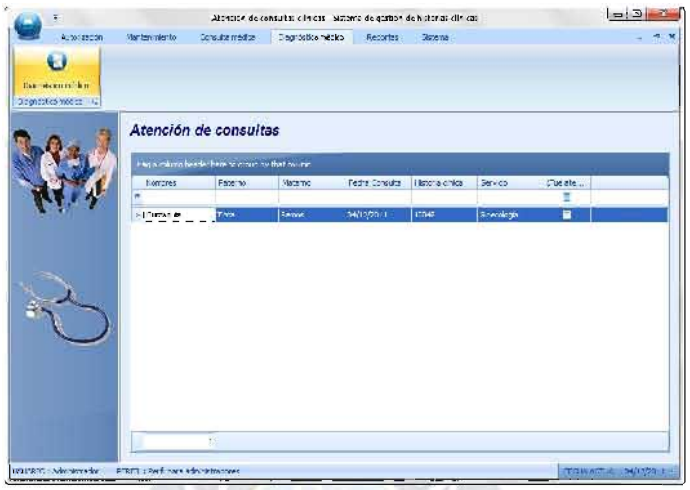


Figura 3.35. Pestaña de diagnostico médico.

En esta pestaña se encuentra el botón de diagnostico médico donde el médico de turno puede ver la lista de pacientes asignado a su consultorio. Seleccionando el botón de atender se puede ver la pantalla donde se registra el diagnostico médico del paciente.

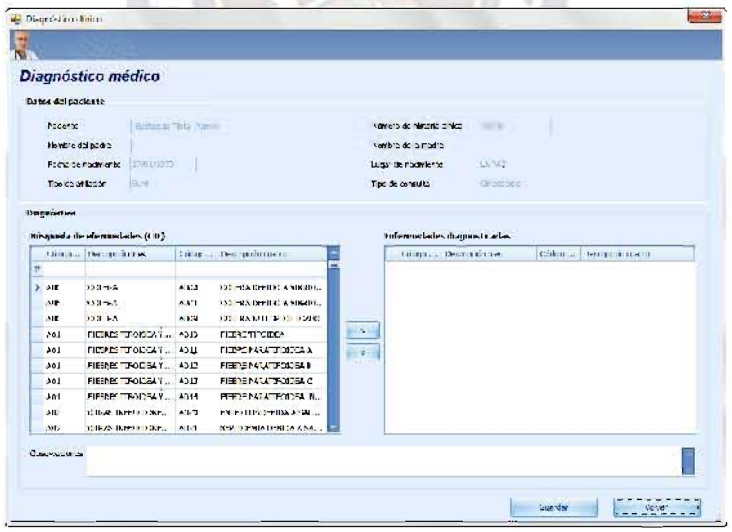


Figura 3.36. Pantalla de diagnostico médico

En esta pantalla se selecciona el diagnostico médico de la lista CIE 10[2].

[2] CIE: código internacional de enfermedades

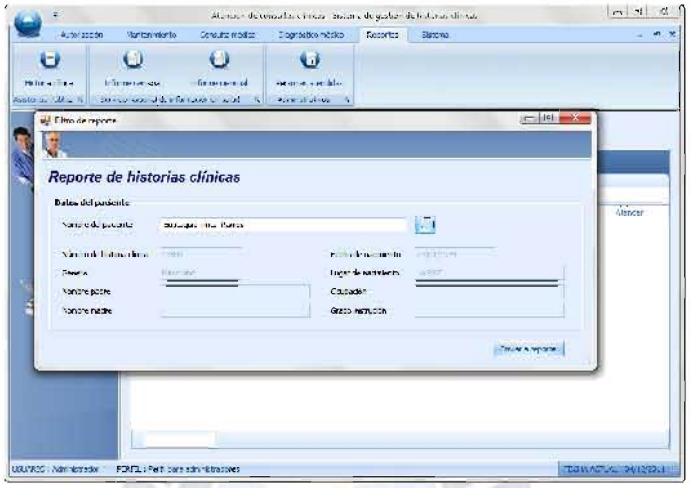


Figura 3.37. Pestaña de reportes

En esta pestaña se encuentra las opciones de reportes para la institución como ser: reporte de historia clínica de un paciente, informes para el SNIS, informes administrativos.

Figura 3.38. Pestaña sistema

En esta pestaña se encuentra la opción de salir del sistema.

3.12. METRICAS DE CALIDAD

3.12.1. Funcionalidad

La funcionalidad de un software se mide según la complejidad del mismo. La funcionalidad se valora evaluando un conjunto de características y capacidades del programa.

Para calcular los puntos función (PF), se utiliza la siguiente relación:

$$PF = \text{conteo total} * [0.65 + 0.01 * \sum(F_i)]$$

Para los parámetros de medida se tiene:

Registro de nuevo paciente

Métricas de punto función

PARAMETROS DE MEDIDA	CUENTA	FACTOR DE PONDERACION				TOTAL
			SIMPLE	MEDIO	COMPLEJO	
Número de entradas externas	22	*	3	4	6	88
Número de entradas externas	15	*	4	5	7	75
Número de consultas externas	14	*	3	4	6	56
Número de archivos lógicos internos	7	*	7	10	155	70
Número de archivos de interfaz externos	3	*	5	7	10	21
Cuenta total						310

FACTOR	SIGNIFICADO	ESCALA F(I)
¿Requiere el sistema copias de seguridad y recuperación fiables?	ESENCIAL	5
¿Requiere comunicación de datos?	ESENCIAL	5
¿Existen funciones de procesos distribuidos?	MEDIO	3
¿Es crítico el rendimiento?	INCIDENCIAL	1
¿Será ejecutado el sistema en un entorno operativo existente y fuertemente utilizado	SIGNIFICATIVO	4
¿Requiere el sistema de entradas interactivas?	SIGNIFICATIVO	4
¿Requiere entrada de datos interactivos sobre múltiples ventanas?	MEDIO	3
¿Se actualizan los archivos maestros de manera iterativa	SIGNIFICATIVO	4
¿Son complejos entradas, salidas, los archivos o peticiones	MODERADO	1

¿Es complejo el procesamiento interno?	MODERADO	2
¿Se ha diseñado el código para ser reutilizable?	SIGNIFICATIVO	4
¿Están incluidas en el diseño la conversión y la instalación?	SIGNIFICATIVO	4
¿Se ha diseñado el sistema para soportar múltiples instalaciones?	ESENCIAL	5
¿Se ha diseñado la aplicación pasa facilitar los cambios y para ser fácilmente utilizada por el usuario?	ESENCIAL	5
$\sum Fi$		50

Con los valores de ponderación de la tabla tenemos los valores de ajuste de complejidad que está dada por $\sum Fi = 50$. Reemplazando en la formula de función se tiene:

$$PF\ real = Cuenta\ total * \left(grado\ de\ confiabilidad + \left(0.01 * \sum Fi\right)\right)$$
$$PF\ real = 310 * \left(0.65 + \left(0.01 * 50\right)\right)$$
$$PF\ real = 356.5$$

Bajo las condiciones de interpolación del valor de punto función la funcionalidad del sistema es óptima.

Pero el resultado esta presentado con un valor de confianza del 65%, así que se puede calcular un valor de confianza del 100% para luego calcular el porcentaje de funcionalidad del sistema

$$PF\ real = Cuenta\ total * \left(grado\ de\ confiabilidad + \left(0.01 * \sum Fi\right)\right)$$
$$PF\ real = 310 * \left(1 + \left(0.01 * 50\right)\right)$$
$$PF\ real = 465$$

Ahora

$$\%PF = \frac{PF\ real}{PF\ esperado} = \frac{356.5}{465} = 0.77$$

Por lo que el sistema desarrollado tiene un 77% de funcionalidad.

Asignación de consulta

Métricas de punto función

PARAMETROS DE MEDIDA	CUENTA	FACTOR DE PONDERACION				TOTAL
			SIMPLE	MEDIO	COMPLEJO	
Número de entradas externas	15	*	3	4	6	60
Número de entradas externas	10	*	4	5	7	50
Número de consultas externas	11	*	3	4	6	44
Número de archivos lógicos internos	6	*	7	10	155	60
Número de archivos de interfaz externos	3	*	5	7	10	21
Cuenta total						235

FACTOR	SIGNIFICADO	ESCALA F(i)
¿Requiere el sistema copias de seguridad y recuperación fiables?	ESENCIAL	5
¿Requiere comunicación de datos?	ESENCIAL	5
¿Existen funciones de procesos distribuidos?	MEDIO	3
¿Es crítico el rendimiento?	INCIDENCIAL	1
¿Será ejecutado el sistema en un entorno operativo existente y fuertemente utilizado	SIGNIFICATIVO	4
¿Requiere el sistema de entradas interactivas?	SIGNIFICATIVO	4
¿Requiere entrada de datos interactivos sobre múltiples ventanas?	MEDIO	3
¿Se actualizan los archivos maestros de manera iterativa	SIGNIFICATIVO	4
¿Son complejos entradas, salidas, los archivos o peticiones	MODERADO	1
¿Es complejo el procesamiento interno?	MODERADO	2
¿Se ha diseñado el código para ser reutilizable?	SIGNIFICATIVO	4
¿Están incluidas en el diseño la conversión y la instalación?	SIGNIFICATIVO	4
¿Se ha diseñado el sistema para soportar múltiples instalaciones?	ESENCIAL	5
¿Se ha diseñado la aplicación pasa facilitar los cambios y para ser fácilmente utilizada por el usuario?	ESENCIAL	5
$\sum Fi$		50

Con los valores de ponderación de la tabla tenemos los valores de ajuste de complejidad que está dada por $\sum Fi = 50$. Reemplazando en la formula de función se tiene:

$$PF\ real = Cuenta\ total * \left(grado\ de\ confiabilidad + \left(0.01 * \sum Fi\right)\right)$$
$$PF\ real = 235 * \left(0.65 + \left(0.01 * 50\right)\right)$$
$$PF\ real = 270$$

Bajo las condiciones de interpolación del valor de punto función la funcionalidad del sistema es buena.

Pero el resultado esta presentado con un valor de confianza del 65%, así que se podrá calcular un valor de confianza del 100% para luego calcular el porcentaje de funcionalidad del sistema

$$PF\ real = Cuenta\ total * \left(grado\ de\ confiabilidad + \left(0.01 * \sum Fi\right)\right)$$
$$PF\ real = 235 * \left(1 + \left(0.01 * 50\right)\right)$$
$$PF\ real = 352.5$$

Ahora:

$$\%PF = \frac{PF\ real}{PF\ esperado} = \frac{270}{352.5} = 0.76$$

Por lo que el sistema desarrollado tiene un 76% de funcionalidad.

Diagnóstico médico

Métricas de punto función

PARAMETROS DE MEDIDA	CUENTA	FACTOR DE PONDERACION				TOTAL
			SIMPLE	MEDIO	COMPLEJO	
Número de entradas externas	15	*	3	4	6	60
Número de entradas externas	10	*	4	5	7	50
Número de consultas externas	11	*	3	4	6	44
Número de archivos lógicos internos	6	*	7	10	155	60
Número de archivos de interfaz externos	3	*	5	7	10	21
Cuenta total						235

FACTOR	SIGNIFICADO	ESCALA F(i)
¿Requiere el sistema copias de seguridad y recuperación fiables?	ESENCIAL	5
¿Requiere comunicación de datos?	ESENCIAL	5
¿Existen funciones de procesos distribuidos?	MEDIO	3
¿Es crítico el rendimiento?	INCIDENCIAL	1

¿Será ejecutado el sistema en un entorno operativo existente y fuertemente utilizado	SIGNIFICATIVO	4
¿Requiere el sistema de entradas interactivas?	SIGNIFICATIVO	4
¿Requiere entrada de datos interactivos sobre múltiples ventanas?	MEDIO	3
¿Se actualizan los archivos maestros de manera iterativa	SIGNIFICATIVO	4
¿Son complejos entradas, salidas, los archivos o peticiones	MODERADO	1
¿Es complejo el procesamiento interno?	MODERADO	2
¿Se ha diseñado el código para ser reutilizable?	SIGNIFICATIVO	4
¿Están incluidas en el diseño la conversión y la instalación?	SIGNIFICATIVO	4
¿Se ha diseñado el sistema para soportar múltiples instalaciones?	ESENCIAL	5
¿Se ha diseñado la aplicación pasa facilitar los cambios y para ser fácilmente utilizada por el usuario?	ESENCIAL	5
$\sum Fi$		50

Con los valores de ponderación de la tabla tenemos los valores de ajuste de complejidad que está dada por $\sum Fi = 50$. Reemplazando en la formula de función tenemos:

$$PF\ real = Cuenta\ total * \left(grado\ de\ confiabilidad + \left(0.01 * \sum Fi\right)\right)$$
$$PF\ real = 235 * \left(0.65 + \left(0.01 * 50\right)\right)$$
$$PF\ real = 270$$

Bajo las condiciones de interpolación del valor de punto función la funcionalidad del sistema es buena.

Pero el resultado esta presentado con un valor de confianza del 65%, así que podremos calcular un valor de confianza del 100% para luego calcular el porcentaje de funcionalidad del sistema

$$PF\ real = Cuenta\ total * \left(grado\ de\ confiabilidad + \left(0.01 * \sum Fi\right)\right)$$
$$PF\ real = 235 * \left(1 + \left(0.01 * 50\right)\right)$$
$$PF\ real = 352.5$$

Ahora

$$\%PF = \frac{PF\ real}{PF\ esperado} = \frac{270}{352.5} = 0.76$$

Por lo que el sistema desarrollado tiene un 76% de funcionalidad.

3.12.2. Confiabilidad

La confiabilidad del sistema está en función de la confiabilidad de cada subsistema.

λ: Tasa de constantes de fallo $\left(\lambda = \frac{nro.\ de\ fallas\ de\ accseso}{nro.\ total\ de\ accesos\ al\ sistema}\right)$

T=periodo de operación en el tiempo.

Después de un determinado tiempo de pruebas en cada módulo del sistema se registro lo siguiente:

	R_1	R_2	R_3
λ	0.03	0.02	0.01
t	5	5	5
R_1	$e^{-\lambda} = 86\%$	$e^{-\lambda} = 91\%$	$e^{-\lambda} = 96\%$

Registro de fallas por modulo en un periodo t=5

El sistema falla si y solo si todos sus componentes fallan. Para la función de transferencia en una combinación de serie de componentes se tiene:

$$R_T(t) = \left[1-(1-R_1)(1-R_2)\right] * R_3$$

$$R_T(t) = \left[1-(1-0.86)(1-0.91)\right] * 0.96$$

$$R_T(t) = 0.947904$$

La confiabilidad del sistema en un periodo de tiempo de 5 horas de operabilidad es del 94%.

3.13. TEORIA DE COSTOS (COCOMO)

El modelo de esfuerzo general aplicable a todos los niveles de aplicación y modos está dado por:

$$E = a(EDSI)b * (EAF)$$

Donde:

E : Es el esfuerzo estimado expresado en hombres-mes

EDSI : Es el número estimado de líneas de código distribuidas en miles para el proyecto

a, b :Son constantes determinadas por el modo del desarrollo, ambos incrementados por la complejidad de la aplicación.

EAF : Es el factor de ajuste de esfuerzo

EAF= 1 (modelo básico)

EAF= producto de 15 factores de costo (modelo intermedia y avanzado)

Coeficientes para el modelo básico que depende de modo de desarrollo:

Modo	Básico			
	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semiencajado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.2	2.5	0.32

El tiempo de desarrollo requerido por el proyecto, en meses es igual a:

$$TDEV = c\,E^d$$

Donde:

E : es el esfuerzo

c,d : son coeficiente, cuyos valores se indicaron anteriormente en una tabla.

El número de programadores es igual a:

$$PG = E/TDEV$$

Usando COCOMO básico para estimar el esfuerzo requerido y el costo del proyecto en el desarrollo de un programa de 3500 líneas en modo orgánico se tiene el siguiente:

$$E = a(EDSI)b * (EAF)$$

E= 2.4(3.5)1.05*1=8.8 mes-hombre

Tiempo de desarrollo del proyecto

$$TDEV = c\,E^{d}$$

TDEV = 2.5 (5.4) ^ [0.38] = 5.71 = 6 meses

Personas necesarias para realizar el proyecto

$$CosteH = E/TDEV$$

CosteH = 8.8 / 5.71 = 1.54 = 1 personas

Costo total del proyecto

$$CosteM = CosteH * Salario\ medio\ entre\ los\ programadores\ y\ analistas$$

CosteM = 1.54 * (4000 Bs) = 6170.36 Bs. En Bolivia

CAPITULO IV

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Con la finalización del presente proyecto y la implementación del sistema se cumplió con el objetivo general propuesto. Dicho sistema optimiza la obtención de la información de los pacientes afiliados al centro de salud Asistencia Pública, mejorando el control y manejo de la información haciendo posible la toma de decisiones a corto plazo.

Así mismo los objetivos específicos planteados también se desarrollaron a cabalidad cumpliendo cada uno de ellos.

- Se implementó agentes inteligentes aplicando la metodología MaSE, el mismo permitió modelar de forma adecuada los agentes de análisis de datos de pacientes y el de generar un número único de identificación del paciente.
- Se realizo el análisis e implementación de la base de datos, que organiza toda la información administrada por la Asistencia Pública.
- Se logro optimizar la obtención de informes y reportes por medio de interfaces diseñadas a partir de la experiencia de los usuarios de la Asistencia Pública.
- La atención de los pacientes es más eficiente gracias a la base de datos centralizada que coadyuva a la obtención de la información del paciente.
- El manejo de la información actualizada optimiza los procesos de diagnostico medico y registro de nuevos pacientes.

Con respecto al desarrollo del sistema de gestión se utilizo la metodología Scrum que permitió el progreso de la misma, con la organización y planificación de las tareas captadas a través de las historias de usuario que fueron útiles para la ingeniería de requerimientos, además se incorporó UML en el modelado del sistema para el mejor entendimiento de los procesos.

Es por ello que la metodología Scrum con el apoyo del UML, permitieron la conclusión del sistema y la obtención de software de calidad, con las funcionalidades requeridas por la institución.

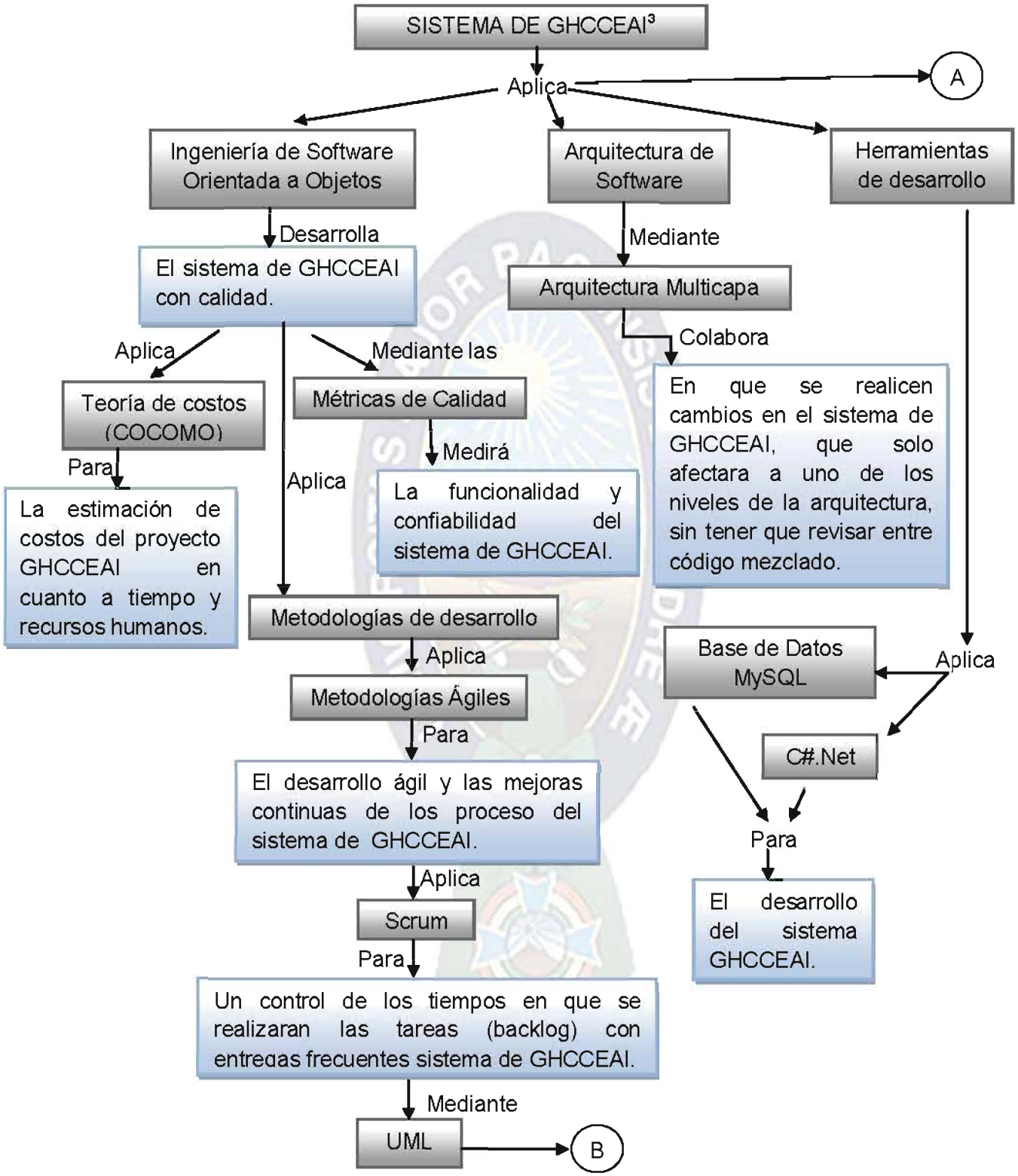
4.2. RECOMENDACIONES

Las recomendaciones para el buen manejo del sistema son:

- Capacitaciones semestrales al personal del centro de salud, con el motivo de formar y verificar el uso adecuado del sistema
- Mantener los documentos impresos de las historias clínicas por seguridad y respaldo legal.
- Realizar el mantenimiento del sistema.
- Realizar copias de seguridad de la información.
- Desarrollar un módulo que realice copias de seguridad automáticamente.
- Automatizar las tareas de otros departamentos como ser farmacia, radiología, emergencias para poder integrar al presente sistema.

ANEXO A

MAPA CONCEPTUAL DEL MARCO TEORICO



[3] GHCCEAI: Gestión de historias Clínicas y Cuadros estadísticos Aplicando Agentes Inteligentes

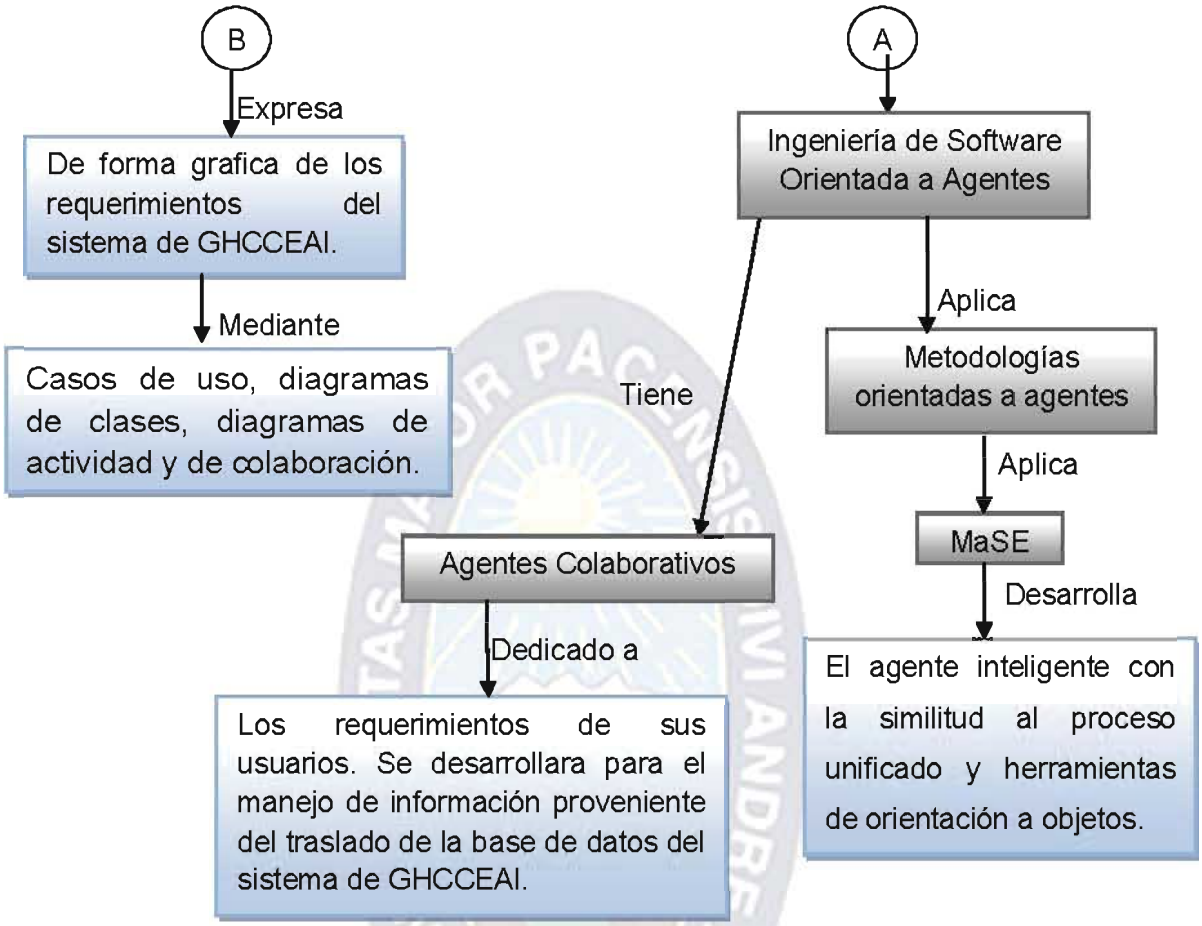


Figura 1 Mapa conceptual del marco teórico

Fuente: Elaboración propia

El sistema "Gestión de Historias Clínicas y Cuadros Estadísticos aplicando Agentes Inteligentes" tiene como base conceptual a la ingeniería de software orientada a objetos que nos permitirá desarrollar y mantener software de calidad, aplicando métricas de calidad que nos permitirá medir la funcionalidad y confiabilidad del sistema y una teoría de costos (COCOMO) para la estimación de costos del proyecto en cuanto a tiempo y recursos humanos.

Así mismo se aplica dentro de esta rama una metodología de desarrollo, en este caso una metodología de desarrollo ágil que debido a la capacidad de cambios durante el desarrollo nos permitirá realizar mejoras continuas de los procesos. Dentro

de esta metodología podemos encontrar varios métodos, sin embargo el método seleccionado para el modelado del sistema es Scrum un método orientado a las personas más que a los procesos, se comienza con una visión general del producto, especificando y dando detalle a las funcionalidades que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en periodos breves (de 8,15 o 30 días), se establece estos periodos de tiempo por consensuo sin imposiciones. El sistema será desarrollado bajo el método Scrum por la organización y la forma de establecer el Product Backlog[4]. Como herramienta de apoyo al modelado se hará uso de UML para expresar de forma gráfica los requerimientos del cliente, aplicado casos de uso, diagramas de clases, diagramas de actividad y de colaboración.

La arquitectura de software que se aplicara es la arquitectura de tres capas con el objetivo de separar la lógica de negocios de la lógica de diseño, llevando a cabo el desarrollo por niveles, que nos será muy útil a la hora de que se realicen cambios en el proyecto, que solo afectara a uno de estos niveles sin tener que revisar entre código mezclado. Para el desarrollo del agente inteligente tenemos como base conceptual a la ingeniería de software orientada a agentes, dentro de ella las metodologías orientadas a agentes como ser: MaSE metodología seleccionada para el modelado del agente inteligente, es un método para desarrolladores acostumbrado al proceso unificado y herramientas de orientación a objetos ya que puede resultar más familiar.

El tipo de agente identificado para el sistema de Historias Clínicas es el Agente colaborativo que constituye un sistema multiagente, es decir, existe más de un agente dedicado a satisfacer los requerimientos de sus usuarios. Se hará el manejo de información proveniente de fuentes distribuidas por lo cual requerirá un sistema multiagente.

Finalmente como base conceptual de las herramientas de desarrollo tenemos como gestor de bases de datos a MySQL que entre sus características tenemos la

[4] Es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar

disponibilidad en gran cantidad de plataformas y sistemas, además de una conectividad segura con diferentes opciones de almacenamiento, según, si se desea velocidad en las operaciones o el mayor número de operaciones disponibles y como lenguaje de programación Visual Studio.Net (en su versión express) administrador de código que coordina toda la operación de los distintos subsistemas del Common Language Runtime, recolector de basura que elimina de memoria objetos no utilizados automáticamente y motor de seguridad que administra la seguridad del código que se ejecuta.

ANEXO B

DIAGRAMAS DE CASOS DE USO

Ingreso al sistema

CASO DE USO: Ingreso al sistema

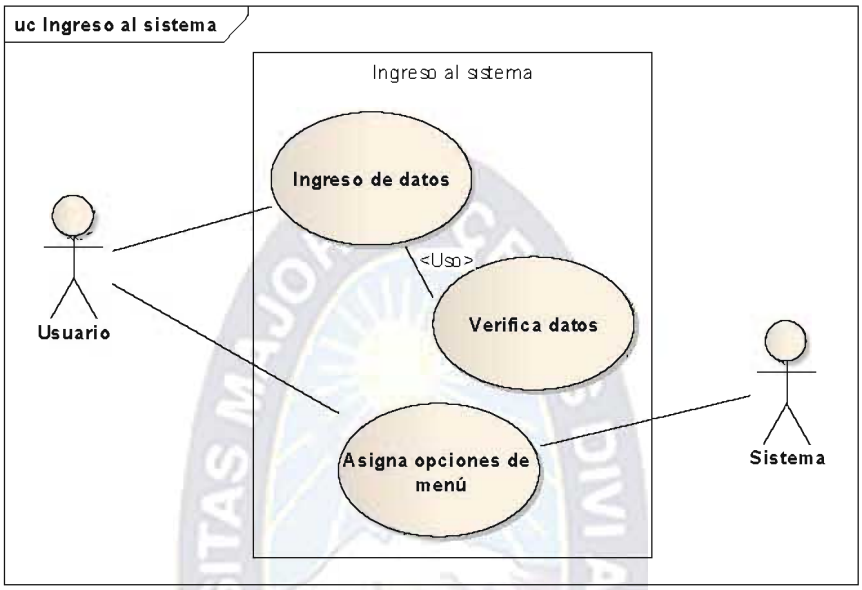


Figura B.1. Casos de Uso de Ingreso al sistema

Fuente: Elaboración según [I. Jacobsob2000]

DESCRIPCIÓN DEL CASO DE USO:

Caso de uso: Ingreso al sistema	
Descripción: El usuario del sistema puede autenticarse e ingresar al menú principal de opciones.	
Actores: Usuario, Sistema	
Requisitos: Ser un usuario del sistema.	
Flujo normal	**Flujo alternativo**
El usuario del sistema ingresa los datos de identificación.	
El sistema verifica los datos del usuario.	Si el usuario no puede ingresar al sistema debe introducir los datos nuevamente.
El sistema le asigna opciones a	

menú según permisos.	
El usuario ingresa al menú de opciones.	
Resultados: Ingreso al sistema y menú de opciones según los permisos dados por el administrador.	

Tabla B.1. Descripción de los Casos de Uso de ingreso al sistema

Fuente: Elaboración según [I. Jacobsob2000]

Generación de consultas y reportes

CASO DE USO: Generación de consultas y reportes

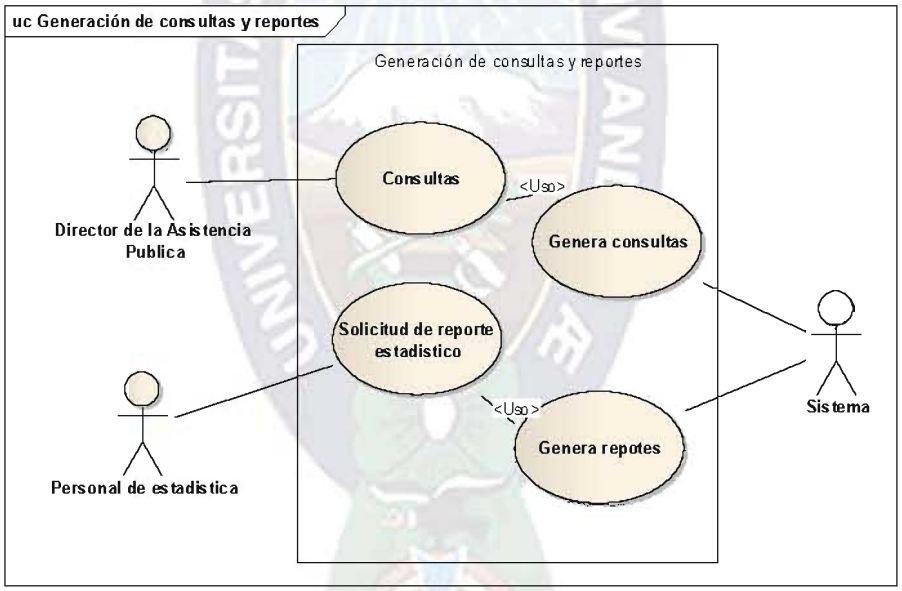


Figura B.2. Casos de Uso de generación de consultas y reportes

Fuente: Elaboración según [I. Jacobsob2000]

DESCRIPCIÓN DEL CASO DE USO:

Caso de uso: Generación de consultas y reportes
Descripción: El usuario del sistema puede generar reportes y realizar consultas

en el momento que se requiere.	
Actores: Usuario (Personal de estadística, director) sistema.	
Requisitos: Ser un usuario del sistema	
Flujo normal	**Flujo alternativo**
1. El usuario del sistema solicita reportes estadísticos.	
2. El sistema genera reportes solicitados.	
3. El usuario realiza consultas al sistema	
4. El sistema genera respuesta a las consultas	
Resultados: Generar reportes estadísticos para el SNIS y para el director de la Asistencia Publica	

Tabla B.2. Descripción de los Casos de Uso de generación de consultas y reportes
Fuente: Elaboración según [I. Jacobsob2000]

DIAGRAMAS DE ACTIVIDAD

Ingreso al sistema

En el siguiente diagrama se describe el conjunto de actividades que se deben realizar para el ingreso del usuario al sistema.

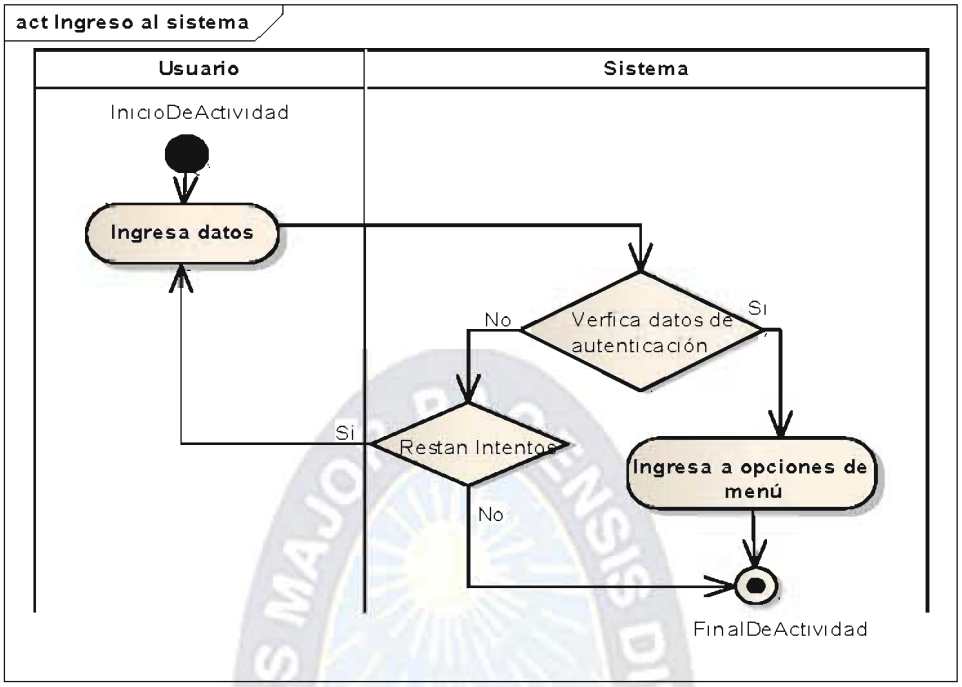
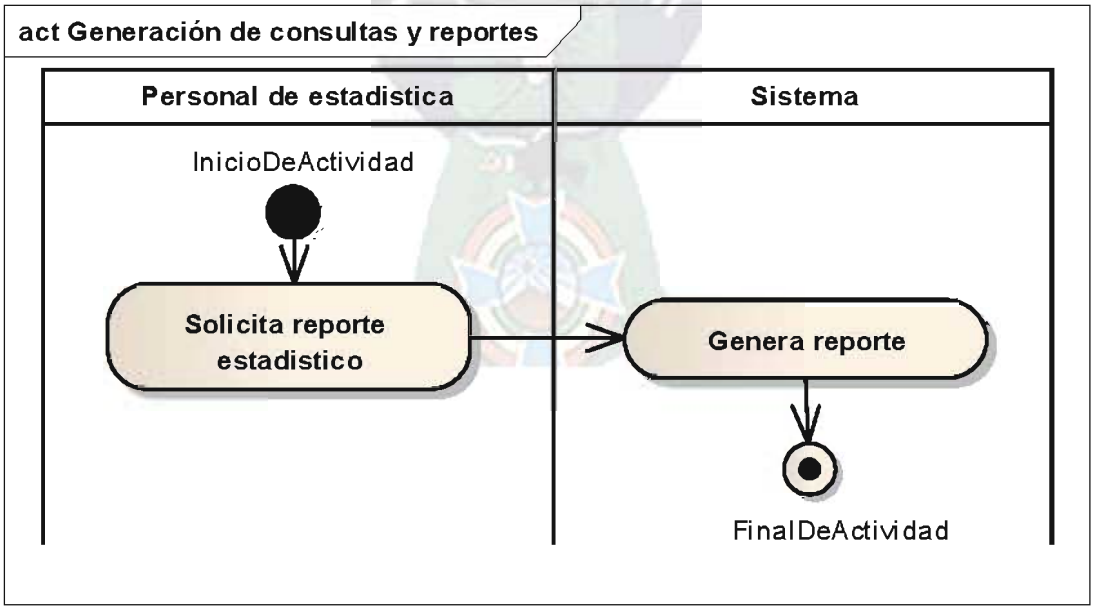


Figura B.3. Diagrama de actividad de ingreso al sistema

Fuente: Elaboración según [I. Jacobsob2000]

Generación de consultas y reportes

El siguiente diagrama describe el conjunto de actividades para solicitar y generar las consultas y reportes estadísticos en el momento que se requieran.



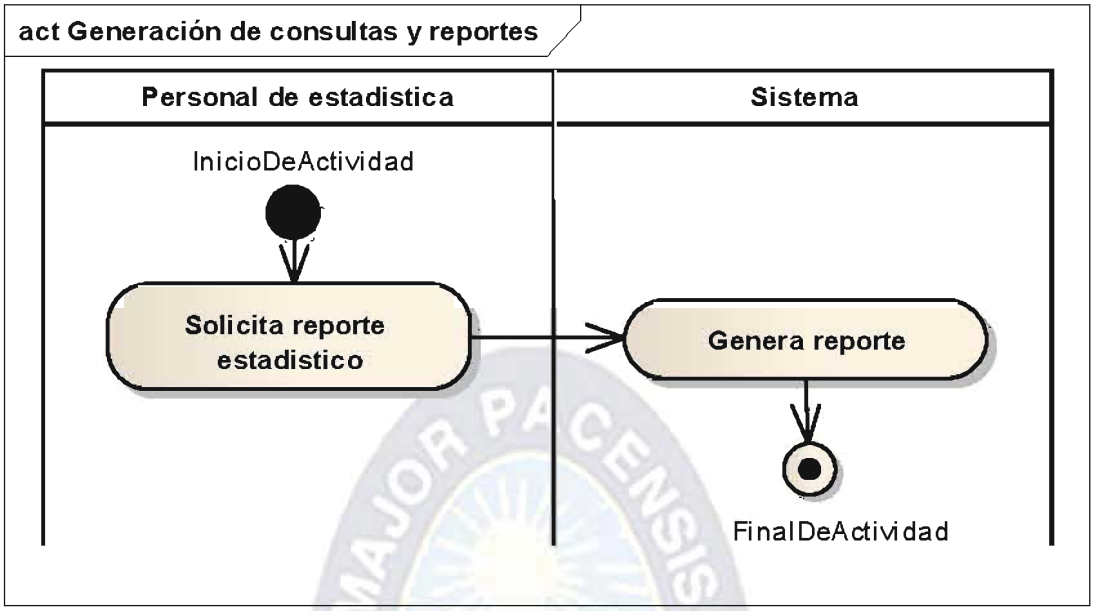


Figura B.4. Diagrama de actividad de generación de reportes y consultas

Fuente: Elaboración según [I. Jacobsob2000]

BIBLIOGRAFÍA

- [Sampieri2002] Sampieri, Fernández y Baptista – 2002, "Metodología de la investigación", tercera edición, Mc Graw Hill, México.
- [I. Jacobsob2000] I. Jacobsob, G. Booch, J. Rumbaugh – 2000, "El proceso unificado de desarrollo de software", Addison Wesley, Mexico.
- [Pressman2010] Pressman – 2010, "Ingeniería de Software: Un enfoque practico", sexta edición, Mc Graw Hill.
- [Senn2001] Senn – 1990, "Ingeniería de Sistemas de Información para la Administración", sexta edición, Mc Graw, Hill/Mexico.
- [Leite1997] Leite, J.C.S.P. – 1997, "Ingeniería de Requisitos", Notas de Cátedra.
- [Dorfman1997] Dorfman, Thayer – 1997, "Ingeniería del Software", IEEE Computer Society Press.
- [Hoare1984] Hoare – 1984, "Programación: ¿Brujería o ciencia?", IEEE Software, vol. 1, N º 2, abril de 1984.
- [Nauer1969] Nauer, Randall, – 1969, "Ingeniería de Software", La OTAN de Asuntos Científicos de la División, Bruselas/Bélgica.
- [Guezzi1991] Guezzi, Jazayeri y Mandrioli – 1991, "Fundamentos de Ingeniería de Software", Prentice Hall.
- [Thayer1997] Thayer, Dorfman, Davis – 1997, "Software Ingeniería de Requerimientos", IEEE Computer Society Press.
- [Nerur2005] Nerur, Mahapatra, Mangalaraj – 2005, "Los desafíos de la migración a las metodologías ágiles", ACM.
- [Tolosa1999] Tolosa, Bordignon – 1999, "La tecnología del software de los agentes".
- [Ramírez2004] Ramírez – 2004, "Editor Inteligente Basado en Agentes", Benemérita Universidad Autónoma de Puebla.
- [Wooldridge1996] Wooldridge, Jennings – 1996, "Agentes N. Software", IEE.
- [Berney1996] Berney – 1996, "Agentes de Software", Universidad Metropolitana de Manchester.

- [Nwana1996] Nwana – 1996, "Agentes de Software: una visión general de Ingeniería del Conocimiento", Cambridge University Press.
- [Weiss1999] Weiss – 1999, "Sistemas multiagente".
- [Geogeroff1995] Geogeroff – 1995, "BDI agentes: De la Teoría a la Práctica", Actas de la Primera Conferencia Internacional sobre sistemas multi-agente (ICMAS-95).
- [Rodriguez2008] Rodríguez – 2008, Tesis de Maestría "Estudio de la Aplicación de Metodologías Ágiles para La Evolución De Productos Software", Facultad de Informática Universidad Politécnica De Madrid.
- [Martín2009] Martín – 2009, "Administración de bases de datos MySQL 5.1", Departamento de Informática y Matemática Aplicada, Universidad de Girona, España.
- [Ramos2011] Ramos – 2011, Tesis Doctoral "Ingeniería de software orientada a agentes en el modelado de sistemas multimedia", Escuela Superior de Ingeniería Informática Universidad de Vigo.
- [Schenone2004] Schenone Marcelo Hernán – 2004, Tesis de Grado "Diseño de una Metodología Ágil de Desarrollo de Software", Facultad de Ingeniería Universidad de Buenos Aires.
- [Scott1999] Scott Wilson – 1999, "El análisis de requisitos y definición de arquitecturas de soluciones". Redmond: Microsoft Press.
- [Boehm1981] Boehm Barry – 1981, "Software Engineering Economics".
- [Gallego2004] Gallego Fernando, Llorens Faraón, Rizo Ramón – 2004, "Breve análisis de algunas metodologías de diseño de SMA", Departamento de Ciencia de la Computación e Inteligencia Artificial Universidad de Alicante, España.
- [Microsoft2011] 2011, "http://msdn.microsoft.com/es-es/library/aa301300(v=VS.71).aspx"