Universidad de las Ciencias Informáticas
Facultad 4

Título: PROPUESTA PARA LOGRAR FIABILIDAD EN LAS APLICACIONES BIOINFORMATICAS VISTA DESDE LA OBTENCION DE REQUISITOS.

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias Informáticas

Autor(es): Kathrin Rodríguez LLanes

Tutor(es): Dra. Aylin Febles Estrada

Co-tutor: Dr. Juan Pedro Febles Rodríguez

Ciudad de la Habana, Junio 2007

"No hay nada en el mundo tan poderoso,
como una idea, cuyo poderoso tiempo ha llegado."

Víctor Hugo

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Kathrin Rodríguez LLanes

_____

Firma del Autor

Aylin Febles Estrada

_____

Firma del Tutor

AGRADECIMIENTOS

Agradezco a todos los que de un modo u otro han influido en mi educación, desde mis abuelos, mis padres que me dieron la vida y me enseñaron los primeros sonidos, mis maestras de los primeros grados de enseñanza, hasta los profesores de la universidad, a quienes nunca olvidaré. A mi hermanita que continúa mis pasos, al resto de mi familia que me ha dado su afecto y comprensión sin límites, a mi novio que me ha apoyado incondicionalmente y ha vivido conmigo los días más tristes brindándome aliento y esperanza, a mis amistades sinceras que me han dado fuerzas hasta llegar aquí…

A todo el que me ha extendido su mano y ha puesto en mí, un granito de tesón, esmero y ahínco proporcionándome esta fuerza diaria para andar con pasos firmes por la avenida del saber, y aprehenderme de ella, doy mis más sinceras gracias.

DEDICATORIA

A mi familia, y en especial a mis padres, que son mi razón de ser y de existir, va dedicado todo el empeño que he puesto en este trabajo, por ser ese su mayor anhelo, a alguien también muy especial para mí, que a pesar de no estar hoy físicamente a mi lado, nunca me ha abandonado porque en mi corazón ocupa un lugar cimero, a mi abuelita Beneda. A ellos porque siempre desearon que llegara este momento y hoy es una realidad…

RESUMEN

En el presente trabajo de diploma quedará confeccionada una guía para el Dpto. de Software Bioinformático de la UCI con los principales elementos o tareas a desarrollar durante el proceso de Ingeniería de Requisitos, específicamente en la etapa de obtención de requisitos para que dichos software cumplan con la fiabilidad. Surge precisamente por la necesidad que existe en la UCI de que haya un modelo único por el cual regirse los proyectos que producen software bioinformático a la hora de realizar el levantamiento de requisitos con el fin de lograr la fiabilidad de estas aplicaciones y por consiguiente la calidad de las mismas. Se realizará además un sitio web para organizar el conocimiento disperso en Internet sobre el tema y poder dar soporte al resultado de numerosas investigaciones que se realizan hoy en el campo de la bioinformática, de la ingeniería de requisitos y de la calidad en las aplicaciones de este tipo; sitio que servirá de apoyo para el trabajo de los equipos de proyecto de desarrollo de software bionformático de la UCI.

PALABRAS CLAVES

Calidad
Requisitos
Bioinformática

INDICE

AGRADECIMIENTOS I
DEDICATORIA II
RESUMEN III
INDICE IV
INTRODUCCION 1
CAPITULO 1: FUNDAMENTACION TEORICA 8
1.1 INTRODUCCION. 8
1.2 CALIDAD DE SOFTWARE. 8
1.2.1 DEFINICIONES DE CALIDAD DE SOFTWARE. 8
1.2.2 PRINCIPIOS PARA OBTENER UN SOFTWARE CON CALIDAD. 9
1.2.3 CONTROL DE LA CALIDAD DEL SOFTWARE. 9
1.2.4 FIABILIDAD 10
1.3 BIOINFORMATICA. 11
1.3.1 SURGIMIENTO DE LA BIOINFORMATICA. 11
1.3.2 LA BIOINFORMATICA EN CUBA. 12
1.3.3 ZONAS DE OPORTUNIDAD PARA LA BIOINFORMATICA EN CUBA. 14
1.3.4 CENTROS DEDICADOS A LA BIOINFORMATICA EN CUBA. PRINCIPALES PROYECTOS. 16
1.3.5 LA BIOINFORMATICA EN LA UCI. 18
1.3.6 ESTADO ACTUAL DE LA BIOINFORMATICA. 19
1.4 INGENIERIA DE REQUISITOS. 20
1.4.1 PROCESOS DE LA INGENIERIA DE REQUISITOS. 22
1.4.2 IMPORTANCIA DE LA INGENIERIA DE REQUISITOS. 24
1.4.3 LOS DEFECTOS MÁS COMUNES EN LA INGENIERÍA DE REQUISITOS Y SUS CONSECUENCIAS. 24
1.4.4 BENEFICIOS DE UNA BUENA INGENIERIA DE REQUISITOS. 29
1.4.5 OBTENCIÓN DE REQUISITOS. TECNICAS DE ELICITACION. 29
1.4.6 PROPUESTA METODOLÓGICA PARA LA OBTENCIÓN DE REQUISITOS. 33
1.4.7 DEFINICION DE REQUISITOS. 35
1.5 CONCLUSIÓN. 37
CAPITULO 2: UNA PROPUESTA DE ORGANIZACIÓN DEL CONOCIMIENTO PARA LOGRAR FIABILIDAD EN LAS APLICACIONES BIOINFORMATICAS. 39
2.1 INTRODUCCIÓN. 39
2.2 CARACTERÍSTICAS ESPECÍFICAS DE LAS APLICACIONES BIOINFORMÁTICAS. 39
2.3 INGENIERÍA DE REQUISITOS EN APLICACIONES BIOINFORMÁTICAS. 41
2.4 APLICACIONES BIOINFORMÁTICAS. FACTORES DE CALIDAD. 43

2.5 ATRIBUTOS DE LA FIABILIDAD..... 47
2.6 LA FIABILIDAD EN LAS APLICACIONES BIOINFORMÁTICAS..... 50
2.7 PROPUESTA PARA LOGRAR LA FIABILIDAD EN LOS SOFTWARE BIOINFORMÁTICOS DESDE EL PROCESO DE OBTENCIÓN DE REQUISITOS. 52
2.8 EVALUACIÓN DE LA FIABILIDAD..... 63
2.8.1 PROPUESTA PARA MEDIR LA FIABILIDAD DE UN SOFTWARE. .. 63
2.8.2 LISTA DE CHEQUEO. 66
2.9 ORGANIZACIÓN DEL CONOCIMIENTO. 69
2.10 CONCLUSIÓN..... 70
CAPITULO 3: DESCRIPCIÓN DEL SITIO. 71
3.1 INTRODUCCIÓN. 71
3.2 ELABORACIÓN DEL SITIO..... 71
3.2.1 CONFECCIÓN DEL ÁRBOL DE NAVEGACIÓN. 71
3.2.2 REQUERIMIENTOS FUNCIONALES (SERVICIOS DEL SITIO) 77
3.2.3 ARQUITECTURA DE INFORMACIÓN..... 77
3.3 VALIDACIÓN DE LA PROPUESTA..... 82
3.3.1 RESULTADOS DE LA VALORACION DE LA PROPUESTA..... 84
3.4 CONCLUSIÓN..... 84
CONCLUSIONES. 85
GLOSARIO DE TERMINOS..... 86
RECOMENDACIONES..... 88
BIBLIOGRAFIA. 89
BIBLIOGRAFIA CITADA. 89
BIBLIOGRAFIA CONSULTADA. 94
ANEXOS. 96

INTRODUCCION

El auge de superordenadores, el desarrollo de la gran autopista de la información y las comunicaciones y la cantidad de datos de las secuencias geonómicas, han permitido aumentar las perspectivas laborales de futuro en la biología y la medicina. El área de la bioinformática es uno de los sectores con grandes potencialidades de crecimiento en todo el mundo para los próximos años, asimismo, el siglo XXI podría ser recordado como el siglo de la Bioinformática, pues abarca áreas tan amplias como la genómica y la proteómica. Debería ser más que una herramienta para procesar datos, una herramienta capaz de sugerir nuevos experimentos e ideas en base a la información recabada y dar sentido a futuros proyectos.

Son muchas las definiciones que autores diversos ofrecen al término *Bioinformática.* Estas constituyen algunas de ellas:

"Es una disciplina científica dedicada a la investigación y desarrollo de herramientas útiles para llegar a entender el flujo de información desde los genes a las estructuras moleculares, a su función bioquímica, a su conducta biológica, y finalmente a su influencia en las enfermedades y en la salud." (Ramírez Domínguez)

"Es una disciplina científica emergente que utiliza tecnología de la información para organizar, analizar y distribuir información biológica con la finalidad de responder preguntas complejas en biología. Bioinformática es un área de investigación multidisciplinaria, la cual puede ser ampliamente definida como la interface entre dos ciencias: Biología y Computación y esta impulsada por la incógnita del genoma humano y la promesa de una nueva era en la cual la investigación genómica puede ayudar dramáticamente a mejorar la condición y calidad de vida humana." (Zepeda García)

"Es un campo de la ciencia en el cual confluyen varias disciplinas tales como: biología, computación y tecnología de la información. El fin último de este campo es facilitar el descubrimiento de nuevas ideas biológicas así como crear perspectivas globales a partir de las cuales se puedan discernir principios unificadores en biología." (National Center for Biotechnology Information (NCBI), 2001)

"La Bioinformática es el uso de las matemáticas y de las técnicas informáticas para resolver problemas biológicos, normalmente creando o usando programas informáticos, modelos matemáticos o ambos." (Proyecto Genoma Humano (PGH), 2001)

Finalmente, todas convergen en la aplicación de las tecnologías de la computación y la información para dar solución a problemas de origen biológico. Es por ello que la bioinformática es un componente clave en el progreso de la medicina. Sin el desarrollo de esta rama no es actualmente posible enfrentar proyectos que aspiren a desarrollar medicamentos y otros productos novedosos para el tratamiento de individuos con padecimientos diversos.

> Bioinformática = Herramientas Computacionales de Alto Nivel + Captura y Almacenamiento de Datos + Procesamiento de los Datos Biológicos.

En la década de los años 80, Cuba comenzó la introducción y el desarrollo de las técnicas modernas de la biotecnología, justo en el momento en que este campo iniciaba su progreso en el resto del mundo. Se crearon varios centros y se integró el Polo Científico del Oeste de la Ciudad de La Habana, que situó el país en una posición competitiva incluso con respecto a los países desarrollados.

En nuestros días, el desarrollo de aplicaciones bioinformáticas no se lleva a cabo siguiendo metodologías de desarrollo, ni se evalúa la calidad de las mismas. Precisamente uno de los retos que enfrenta hoy la Bioinformática en Cuba consiste en producir software con calidad, lo cual constituye un requisito imprescindible para poder colocar los productos nacionales en el mercado mundial.

Debido a la necesidad que tiene el país de producir software bioinformáticos con calidad, en especial nuestra universidad (UCI), como empresa productora de software es que se realizará esta investigación; teniendo como base un estudio realizado por ingenieros informáticos, el cual arrojó que dentro de todos los factores de calidad propuestos por McCall, la fiabilidad es uno de los más importantes en la producción de este tipo de software. (Febles Rodríguez & Febles Estrada, 2005)

Actualmente la UCI no posee una guía donde se definan claramente las técnicas a tener en cuenta durante el proceso de levantamiento de requisitos para que los proyectos bioinformáticos cumplan con la fiabilidad.
La fiabilidad es aquí entendida como la probabilidad de que un programa realice su objetivo satisfactoriamente (sin fallos) en un determinado período de tiempo y en un entorno concreto (denominado perfil operacional)".
En términos estadísticos la fiabilidad de un software se define como: la probabilidad de operación libre de fallos de un programa de computadoras en un entorno determinado en un tiempo específico (MUS, 1987)
En todo caso puede asegurarse que la fiabilidad en la construcción de software es un elemento importante de su calidad general; si un programa falla frecuente y repetidamente en su funcionamiento, no importa si el resto de los factores de calidad son aceptables. Por ello, se ha convertido en una necesidad reconocida, no sólo en el nuevo programa nacional de Tecnologías de la Información y para la Bioinformática, sino también para la Industria de Desarrollo de Software.

Para garantizar esta fiabilidad son necesarias técnicas, métodos y herramientas que soporten de forma adecuada el proceso de desarrollo y, para esto, se necesita una inversión de recursos que promueva avances en la investigación de nuevos métodos de desarrollo de software, nuevas técnicas de verificación y validación, nuevas tecnologías de soporte para componentes de software comercial, etc. La fiabilidad del software, a diferencia de otros factores de calidad, puede ser medida mediante datos históricos o de desarrollo.

Partiendo de la situación problémica existente se ha definido, para este trabajo de diploma, el siguiente **problema científico:**
Para el desarrollo de las aplicaciones bioinformáticas en la UCI no se siguen pasos mínimos en el levantamiento de requisitos para garantizar fiabilidad ni se evalúa su cumplimiento al finalizar el desarrollo.

Se tiene como **objeto de estudio** la Ingeniería de Requisitos, los atributos de calidad y las aplicaciones bioinformáticas.

Del objeto de estudio anterior se deriva como **campo de acción** La Ingeniería de Requisitos en proyectos bioinformáticos.

Para resolver el problema se propone el siguiente **objetivo general:**
Proponer una guía para la Obtención de Requisitos y evaluación de las aplicaciones bioinformáticas que ayude a alcanzar fiabilidad en estas.

El objetivo general puede desglosarse en los siguientes **objetivos específicos**:

- Hacer un estudio del estado del arte relacionado con la fiabilidad en software bioinformáticos.
- Investigar cómo lograr un correcto proceso de Obtención de Requisitos en proyectos bioinformáticos.
- Definir elementos y técnicas a tener en cuenta durante el proceso de Obtención de Requisitos para que un sistema bioinformático cumpla con la fiabilidad.
- Conformar una lista de chequeo para evaluar el cumplimiento de la fiabilidad en las aplicaciones bioinformáticas.
- Realizar un sitio web donde se publique la Propuesta, el resto de la información que sirvió de base en la confección de la misma y el resultado de otras investigaciones vinculadas al tema.

Dentro de las **tareas** propuestas para dar solución a los objetivos planteados están:

- Revisión bibliográfica sobre las tendencias actuales en Cuba y el mundo de cómo lograr sistemas bioinformáticos con calidad.
- Entrevistar a especialistas en calidad de software bioinformáticos.

- Implementar y aplicar una ronda de encuestas a personas experimentadas en el levantamiento de requisitos.
- Investigar cuáles son las principales técnicas a utilizar para lograr un sólido proceso de captura de requisitos.
- Desarrollar una lista de chequeo para comprobar la fiabilidad de un software bioinformático.
- Diseñar e implementar un sitio web que sirva como repositorio de información que relacione la ingeniería y calidad de software con las aplicaciones bioinformáticas.

Hipótesis: Contar con una guía a seguir durante el levantamiento de requisitos y una lista de chequeo para evaluar las aplicaciones ayudará a garantizar la fiabilidad en los software bioinformáticas.

Se utiliza como estrategia de investigación la Investigación explicativa o experimental, porque tiene como objetivo principal, determinar las causas que producen el fenómeno en estudio, es decir, se determinan las razones por las cuales no se obtiene un proceso de Obtención de Requisitos con calidad en los proyectos de desarrollo de software bioinformático y partiendo de ellas, se propone una solución.

Métodos teóricos de la Investigación:

Histórico-lógico: Sirvió para determinar las tendencias actuales en el desarrollo del proceso de Obtención de Requisitos y para determinar los modelos que se siguen en este proceso.

Hipotético-Deductivo: Este método permitió obtener la hipótesis y a partir de ella inferir conclusiones en el transcurso de la investigación y hacer formulaciones particulares, las cuales sí son validadas por expertos y reafirman la validez de las formulaciones y de la hipótesis general que se sustenta.

Analítico-sintético: Mediante este método se procesó la información y sirvió para arribar a conclusiones en la investigación, además para determinar las actividades que se incluirán dentro de la guía a proponer.

Métodos empíricos:

La observación: Planificada y dirigida con el fin de realizar la fundamentación teórica del problema.

Métodos particulares:

La entrevista: Se realiza a especialistas en la producción de aplicaciones bioinformáticas para obtener información del fenómeno tratado.

La encuesta: Se aplica con el fin de lograr información acerca de cómo se lleva a cabo, por el Dpto. de Software Bioinformático de la UCI, el proceso de levantamiento de requisitos a través de un conjunto de preguntas.

Aportes prácticos esperados del trabajo:

El presente trabajo debe dar la posibilidad al Dpto. de Software Bioinformático de la UCI de contar con una guía para que sus productos cumplan con la fiabilidad desde la Obtención de Requisitos; ofreciendo técnicas (preguntas específicas a realizar durante esta etapa), métodos y herramientas que faciliten dicho proceso. También permitirá contar con un sitio elaborado para de facilitar el acceso a investigaciones bioinformáticas donde se publicará la guía anterior.

Estructura de la tesis:

La tesis está estructurada de la siguiente forma: en el capítulo 1 se realiza una fundamentación teórica del tema en cuestión, vista desde el ámbito internacional, nacional y enmarcado en la Universidad de Ciencias Informáticas. En el capítulo 2 se propone la solución mediante la confección de una guía para lograr la fiabilidad de los Software Bioinformáticos desde el proceso de obtención de requisitos y se conforma una lista de chequeo para evaluar ese atributo en una aplicación bioinformática ya

concluida. En el capítulo 3 se desarrolla la validación de dicha propuesta, conjuntamente con una descripción detallada del sitio web elaborado.

CAPITULO 1: FUNDAMENTACION TEORICA.

1.1 INTRODUCCION.

En el presente capítulo se pretende abordar ideas generales y principales conceptos relacionados con la calidad de los software bioinformáticos, para lo cual se ha dividido el tema en tres subtemas o principios básicos: la bioinformática, la ingeniería de requisitos y la calidad del software, que sirven de apoyo en el desarrollo conjunto de la temática. Se incluye además un estado del arte del tema tratado, a nivel internacional, nacional y de la Universidad, y las ideas que en la actualidad han servido de base para la solución del problema que se enfrenta, sobre las que se profundiza más adelante.

1.2 CALIDAD DE SOFTWARE.

1.2.1 DEFINICIONES DE CALIDAD DE SOFTWARE.

Las definiciones de calidad de software son ya comúnmente aceptadas por la comunidad científica internacional, aparecen en los textos y en las clases y conferencias de los profesionales mas avezados en el campo de la informática. Se ha convertido en la actualidad en la meta principal para cualquier empresa productora de software en cualquier parte del mundo.

Resulta suficientemente clara la definición que ofrece (Pressman, 1992).

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente"

Otra definición sencilla y concreta es la que aparece en (Norma ISO 8402 UNE, 1992)

"El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas".

Finalmente puede ser aceptada como una definición general, la siguiente: *"La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad."* (Fernández Carrasco, García León, & Beltrán Benavides, 1995) Pues aquí relacionan

todos los atributos o factores que forman parte de la calidad de un software, y por los cuales podemos medir, si una aplicación, posee o no calidad.

1.2.2 PRINCIPIOS PARA OBTENER UN SOFTWARE CON CALIDAD.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

La política establecida debe estar sustentada sobre tres principios básicos: tecnológico, administrativo y ergonómico.

- El principio tecnológico define las técnicas a utilizar en el proceso de desarrollo del software.
- El principio administrativo contempla las funciones de planificación y control del desarrollo del software, así como la organización del ambiente o centro de ingeniería de software.
- El principio ergonómico define la interfaz entre el usuario y el ambiente automatizado.

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura. Para el aseguramiento de la calidad es necesario su control o evaluación. (Fernández Carrasco, García León, & Beltrán Benavides, 1995)

1.2.3 CONTROL DE LA CALIDAD DEL SOFTWARE.

El control de calidad en la producción de software, garantiza evitar riesgos en su uso. Para controlar la calidad del software es necesario, ante todo, definir los parámetros, indicadores o criterios de medición, ya que, como bien plantea Tom De Marco, "usted no puede controlar lo que no se puede medir".

Las cualidades para medir la calidad del software son definidas por innumerables autores, los cuales las denominan y agrupan de formas diferentes. Por ejemplo, John Wiley define métricas de calidad y criterios, donde cada métrica se obtiene a partir de

combinaciones de los diferentes criterios. La Metodología para la evaluación de la calidad de los medios de programas de la CIC, de Rusia, define indicadores de calidad estructurados en cuatro niveles jerárquicos: factor, criterio, métrica, elemento de evaluación, donde cada nivel inferior contiene los indicadores que conforman el nivel precedente. Otros autores identifican la calidad con el nivel de complejidad del software y definen dos categorías de métricas: de complejidad de programa o código, y de complejidad de sistema o estructura.

Todos los autores coinciden en que el software posee determinados índices medibles que son las bases para la calidad, el control y el perfeccionamiento de la productividad. Una vez seleccionados los índices de calidad, se debe establecer el proceso de control, que requiere los siguientes pasos:

- Definir el software que va a ser controlado: clasificación por tipo, esfera de aplicación, complejidad, etc., de acuerdo con los estándares establecidos para el desarrollo del software.
- Seleccionar una medida que pueda ser aplicada al objeto de control. Para cada clase de software es necesario definir los indicadores y sus magnitudes.
- Crear o determinar los métodos de valoración de los indicadores: métodos manuales como cuestionarios o encuestas estándares para la medición de criterios periciales y herramientas automatizadas para medir los criterios de cálculo.
- Definir las regulaciones organizativas para realizar el control: quiénes participan en el control de la calidad, cuándo se realiza, qué documentos deben ser revisados y elaborados, etc. (Fernández Carrasco, García León, & Beltrán Benavides, 1995)

1.2.4 FIABILIDAD.

Cada vez son más los usuarios que, en todo tipo de contextos y para todo tipo de sistemas, demandan productos que hagan lo que tienen que hacer de manera segura y **fiable**, es decir, que exigen funcionalidad, cumpliendo sus especificaciones; pero sin renunciar a aspectos cada vez más importantes como la fiabilidad, la seguridad, la

disponibilidad o la facilidad de mantenimiento de sus sistemas. De hecho ¿quién está hoy en día dispuesto a adquirir un equipo que no posea las características que se acaban de mencionar?

Para ofrecer productos con posibilidades de éxito en el competitivo mercado de la informática, no es suficiente con desarrollar sistemas que simplemente funcionen. Estos sistemas deben además cumplir con otros requisitos no funcionales, que serán los que finalmente acaben condicionando el nivel de confianza con el que los usuarios podrán explotar las capacidades del sistema. Así pues, una buena solución informática debe, entre otras muchas cosas, ser fácil de utilizar y mantener, capaz de evitar fallos, y en caso de problemas, debe ofrecer un comportamiento seguro y robusto, asegurando la continuidad del servicio ofrecido (es decir, tolerando la ocurrencia del problema) u ofreciendo un procedimiento de parada (o avería) del sistema controlado y, en la medida de lo posible, inocuo tanto para los usuarios, como para sus datos y operaciones. (Calero, Ruíz, & Pianttini, 2006)

Por **fiabilidad** se entiende a la capacidad del producto software para mantener un nivel específico de funcionamiento cuando se está utilizando bajo condiciones especificadas. (Alarcos, 1996)

Las limitaciones en fiabilidad son debidas a fallos en los requerimientos, diseño o implementación. Dependen de la manera en que se utiliza el producto de software y de las opciones del programa seleccionadas, más que del tiempo transcurrido. (Muriel Herrero & Díe Socías, 2007)

1.3 BIOINFORMATICA.

1.3.1 SURGIMIENTO DE LA BIOINFORMATICA.

Además de las políticas de desarrollo e inyecciones económicas, en el avance investigador de las empresas biotecnológicas han contribuido enormemente las nuevas tecnologías de la información, hasta el punto de que hoy en día no se concibe la investigación sin el uso de aplicaciones informáticas. Esta simbiosis entre ambos mundos es lo que ha dado lugar al nacimiento de una nueva disciplina a caballo entre la biología molecular, las matemáticas y la computación: la bioinformática.

Ya en los años sesenta la investigación biológica comenzó a organizar y almacenar la información generada en bases de datos, pero no alcanzó un papel clave hasta la década de los noventa. En estos años, "la bioinformática ha pasado de ser una mera herramienta para la interpretación de datos a convertirse en una nueva aproximación que permite la creación de conocimiento biomédico" (Barbadilla). Antonio Barbadilla, profesor e investigador de Genética y Microbiología en la Universidad Autónoma de Barcelona y fundador de la empresa de bioinformática Ebiointel plantea que fue C.Venter, investigador que secuenció el genoma humano, quien marcó un antes y un después al apoyarse en la bioinformática para concluir su descubrimiento.

Así, la informática demostró que servía para algo más que recopilación y obtención masiva de datos y "se dio el paso del mero acopio de información al entender y conocer lo que quiere decir esa información" (Carazo), profesor de Investigación del CSIC en el Centro Nacional de Biotecnología (CNB) y fundador de Integromics, firma española de software para el análisis de datos en genómica y proteómica.

Hoy en día es inconcebible pensar en acometer un proyecto de innovación biológica sin el uso de potentes herramientas informáticas que, además de facilitar la gestión de la ingente documentación que genera la actividad biológica, recogen y permiten el acceder al conocimiento de los investigadores de todo el mundo.

La bioinformática se ha constituido como una especialidad imprescindible dentro del ámbito de la biotecnología por su capacidad de analizar toda la información que genera este sector. Sin ella sería imposible progresar hoy en día.

1.3.2 LA BIOINFORMATICA EN CUBA.

Cuba se ha introducido en esta disciplina científica con resultados alentadores y en algunos casos sorprendentes. Un somero recuento de los logros alcanzados en los últimos años demuestra un importante abordaje de carácter científico, donde la elaboración o uso de programas computacionales es el factor predominante. He aquí un pequeño historial acerca del desarrollo acelerado de la Bioinformática en Cuba.

Años 80	Cuba entre los países más avanzados en Biotecnología.
Años 90	Productos biotecnológicos establecidos
Año 2001	Validación OMS (Organización Mundial de la Salud).
Años 90	Ejecución del proyecto del genoma humano y desarrollo impresionante en avances tecnológicos y de la computación.
Actualidad	Incorporar la Bioinformática en la estrategia de desarrollo o perder nuestras posibilidades en Biotecnología.

Tabla 1.1: Evolución del desarrollo de la Bioinformática en Cuba.

El listado que a continuación aparece representa un conjunto de títulos de investigaciones realizadas por diferentes instituciones cubanas que matizan el quehacer bioinformático del país y reflejan la madurez alcanzada en una disciplina tan compleja

- Creación, desarrollo y utilización de bases de datos de tipo biológico (incluyendo lo pertinente de la agricultura, el medio ambiente, el sistema de salud pública y la propia medicina).
- Estructura y funciones de las proteínas y otras biomoléculas relevantes. Modelación molecular computacional.
- Relaciones cuantitativas entre la estructura y las propiedades moleculares (QSAR) así como relaciones ligando - receptor para identificar estructuras moleculares bioactivas: diseño de fármacos.
- Genes y genómica.
- Microarreglos (de DNA, proteínas, tejidos, células) y química combinatoria.
- Filogenética y colecciones organizadas de especies biológicas.
- Variaciones del DNA por nucleótidos individuales (SNP o polimorfismos por solo un nucleótido).
- Biología de sistemas.

- Modelos y algoritmos matemáticos y computacionales aplicados a las ciencias de la vida, incluyendo procedimientos de inteligencia artificial.
- Implementación y desarrollo de sistemas de cómputo de alto rendimiento (computación paralela y distribuida). (Febles Rodríguez & Febles Estrada, 2005)

Dos conclusiones evidentes que surgen al analizar esta relación son:
El país se cuenta con un espectro inicial aceptable de trabajo en los tópicos de la bioinformática que se practica hoy en día en el mundo, con disponibilidad del software indispensable (sobre la base de licencias libres, donaciones, generación propia y apropiaciones).
La bioinformática puede ser un catalizador para el desarrollo de las técnicas de cómputo de alto rendimiento en el país, donde solo han trabajado históricamente algunos contados grupos. Estos procedimientos pueden tener un impacto considerable en múltiples esferas de la economía y la sociedad.
Pero la Bioinformática como toda disciplina investigativa no se estanca, ni se niega al desarrollo, sino que mantiene una evolución constante y en espiral, pues siempre existirán problemas a resolver empleando soluciones biológica-computacionales. Dentro de las tantas metas o retos que posee la bioinformática para el año 2010 en Cuba se contemplan las siguientes: asimilar tecnologías de alto flujo, mejorar la conectividad y la capacidad de cómputo, ahondar en la creatividad de los algoritmos de prospección, crear bases de datos propias: (Estándares + Supervisión + Interoperabilidad), ampliar el uso de simulación asistida por computadoras, extraer predicciones verificables de las simulaciones, entre otras. Todo ello contribuirá a crear la "cadena de valor" de la Bioinformática, de gran utilidad en la preservación de la vida humana.

1.3.3 ZONAS DE OPORTUNIDAD PARA LA BIOINFORMATICA EN CUBA.

En Cuba más del 50% de las inversiones se hace en vacunas y biofármacos, con el objetivo de prevenir tanto a niños, jóvenes, como ancianos de padecimientos diversos provocados por cualquier tipo de enfermedad. Investigaciones acerca de la proteómica de microorganismos, entre ellas la proteómica del Meningococo ha marcado una

ventaja competitiva para el país en Meningitis. Cuando en el mundo las empresas biotecnológicas solo han llevado a nivel de registro algunas decenas de productos de uso humano, entre biofármacos y vacunas, Cuba cuenta hoy con la producción de 13 productos biotecnológicos, dos de patente propia y únicos en el mundo (la mencionada vacuna contra la Meningitis B y la estreptoquinasa recombinante, con resultados muy positivos en el tratamiento del infarto del miocardio). También la formulación de algoritmos de predicción de epítopes contribuye de manera directa al descubrimiento de dichas vacunas. Gracias a todo ello, los infantes cubanos reciben hoy, inmunización contra tuberculosis, tifoidea, rubéola, parotiditis, meningitis B-C, hepatitis viral B, mediante 10 vacunas que se suministran gratuitamente, siete de ellas producidas en el país, algunas representativas de aportes a escala mundial. Y realmente se espera que en el futuro se puedan incorporar otras sustancias inmunobiológicas al sistema de inmunización para ampliar la protección contra otras enfermedades. (Miralles, 2007) Las vacunas terapéuticas contra el cáncer son otros de los productos biofarmacéuticos exclusivos de Cuba. También en ensayos para tratamiento de tumores de pulmón, apreciándose un incremento significativo en la sobrevida de pacientes con la enfermedad avanzada, así como de la calidad de vida. (Veloz, 2004)

Otra de las áreas de desarrollo en Cuba donde la bioinformática juega un papel preponderante es en la rama de las Neurociencias. A través de técnicas bioinformáticas se llevan a cabo tareas tales como: el mapeo cerebral, la integración de imágenes cerebrales y prótesis biónicas, importantes para el descubrimiento o tratamiento de enfermedades del sistema nervioso central y periférico.

Otra de las aristas donde el empleo de técnicas bioinformáticas se hace imprescindible es en la creación de los Sistemas de Diagnóstico a escala poblacional, en los cuales se hace una caracterización genética de la población cubana y se trata la Epidemiología Molecular.

.

1.3.4 CENTROS DEDICADOS A LA BIOINFORMATICA EN CUBA. PRINCIPALES PROYECTOS.

La diversidad de centros dedicados hoy a la bioinformática, evidencian la masificación de esta ciencia en todo el país, entre ellos: el Centro de Ingeniería Genética y Biotecnología (CIGB), el Centro de Inmunología Molecular (CIM), el Centro de Neurociencias de Cuba (CNC), el Instituto Finlay, el Centro de Química Farmacéutica (CQF), el Centro Nacional de Sanidad Agropecuaria, el Centro Nacional de Investigaciones Científicas (CNIC), Centro Nacional de Bioinformática (BIOINFO), el Instituto Superior de Tecnologías y Ciencias Aplicadas (InSTEC), el Instituto de Cibernética, Matemática y Física (ICIMAF), la UH, etcétera. A continuación se relacionan los principales proyectos de algunos de los centros anteriormente mencionados.

Centros	**Principales Proyectos**
CIGB	• Asimilación Tecnológica y Formación de Recursos Humanos. • Tamizaje “in-silico” (Dengue y Sida). • Genes asociados con HTA y enfermedades Neuropsiquiátricas. • Varios proyectos en Proteómica.
Instituto FINLAY	• Proteómica de la Neisseria.
CIM	• Simulador del Sistema Inmune. • “Docking” Dinámico. • Evaluación de predicciones sobre combinación entre

	vacunas e inmunosupresión. • Anticuerpos “ANTI-IDIOTIPO” de alta conectividad. • Identificación de redes idiotípicas T-B. • Dinámica de subpoblaciones de linfocitos. • Predicción de interacciones “PROTEINA-LIGANDO”. • Transito rápido a la “Prueba de concepto en la clínica”.
CNC	• Proyecto Neuroinformática. • Mapeo eléctrico cerebral. • Tratamiento de Imágenes.
ICIMAF	• Grupo de reconocimiento de patrones e ingeniería de datos.
Fac. Química UH	• Grupo de Química Computacional.
Fac. Física UH	• Cátedra de Complejidad.
Fac. CIB Matemática, UH	• Estudios de Sistemas Dinámicos no-lineales.

Tabla 1.2: Listado de los centros que se dedican a la Bioinformática en Cuba y los principales proyectos que desarrollan.

1.3.5 LA BIOINFORMATICA EN LA UCI.

En noviembre del 2003 surge el grupo de Bioinformática en la UCI, el cual tiene como misión primordial desarrollar software con alto valor agregado para satisfacer las necesidades de los Centros del Polo Científico del Oeste de La Habana, con el propósito de llegar a convertirse en un grupo líder en el desarrollo de software para bioinformática en Cuba.

La UCI cuenta hoy, en el área de la Bioinformática, con varias líneas de investigación dentro de las que se destacan las siguientes:

- Biología de sistemas
- Diseño de fármacos asistido por computadoras
- Computación de alto rendimiento aplicada a la bioinformática (CAR)
- Evolución Molecular

Proyectos de la universidad con Centro del Polo Científico

- BioSyS: Software para la simulación de sistemas biológicos (CIM)
- J-IMMSIM: Simulador del sistema inmune (CIM)
- GRAph-TOol: Software para la predicción de actividad biológica (CQF)
- Screaning and Docking (CIM)
- Software para el alineamiento de proteínas en 3D (CIGB)
- Silenciamiento de genes (CIGB)
- BioGrid: Plataforma de propósito general para la realización de cálculos distribuidos
- Algoritmo paralelo para el análisis de selección positiva
- Detección de selección positiva en genomas de patógenos bacterianos

- Predicción del potencial de surgimiento de mutantes de escape a partir de perfiles estructurales, tasas de evolución molecular, y selección positiva en candidatos vacúnales.

Otros Proyectos

- BioNOVA: Distribución de linux para bioinformática (Facultad 10)
- MicroImage: Software para el tratamiento de imágenes de MicroArray (CIGB)
- Algoritmo de programación genética para tareas de minería de datos

Proyectos Actuales

- Proyecto de Cromatografía(CIGB)
- Proyecto Base Datos (CQF)
- Proyecto Simulación de Sistemas Biológicos (CIM)
- Proyecto Planificación y Aprendizaje. WebService (Neurociencias)

Todos estos proyectos se han ido desarrollando gracias a la colaboración de:
Centro de Inmunología Molecular (CIM)
Centro de Química Farmacéutica (CQF)
Centro de Ingeniería Genética y Biotecnología (CIGB)
Centro Nacional de Bioinformática (BIOINFO)

1.3.6 ESTADO ACTUAL DE LA BIOINFORMATICA.

La Bioinformática, una de las áreas de investigación de mayor dinamismo, aborda el tratamiento y análisis de información en biología mediante el desarrollo y uso de procedimientos, métodos y sistemas basados en las tecnologías de la información.

El diseño de estructuras y procesos a escala molecular utiliza las teorías de la física moderna y la instrumentación de eficientes ordenadores electrónicos. Se trata de una de las disciplinas científicas en más rápido desarrollo y extensión de nuestros días. Esto se evidencia con el hecho de que las revistas especializadas de más alto impacto publican cada vez más frecuentemente artículos donde los hallazgos experimentales

son respaldados por cálculos teóricos de lo obtenido. La popularidad de estos métodos se ha incrementado por el reducido costo actual de las instalaciones para llevar a cabo cálculos de muy alto nivel. El diseño molecular es también uno de los aspectos más relevantes de la naciente bioinformática.

El año 2003 marcó un hito en la historia de la bioinformática con la finalización de la secuencia del genoma humano. El Proyecto Genoma Humano (PGH) consiste en determinar las posiciones relativas de todos los nucleótidos (o pares de bases) e identificar los 20.000 a 25.000 genes presentes en él.

El proyecto, dotado con 3.000 millones de dólares, fue fundado en 1990 por el Departamento de Energía y los Institutos de la Salud de los Estados Unidos, con un plazo de realización de 15 años. Debido a la amplia colaboración internacional, a los avances en el campo de la genómica (especialmente, en el análisis de secuenciación), así como los avances en la tecnología informática, un borrador inicial del genoma fue terminado en el año 2000. (Revistas Nature y Science, 2003)

Los beneficios de este proyecto se esperan fructíferos en los campos de la medicina y de la biotecnología, eventualmente conduciendo a tratamientos o curas de cáncer, Enfermedad de Alzheimer y otras enfermedades. (Bethesda, 2003)

1.4 INGENIERIA DE REQUISITOS.

A pesar de que las herramientas para construir software han evolucionado enormemente, en la actualidad se sigue produciendo software que no es satisfactorio para los clientes y usuarios. Esto indica que los principales problemas que han dado origen a la crisis del software residen en las primeras etapas del desarrollo, cuando hay que decidir las características del producto software a desarrollar. Es necesario comprender las necesidades reales de los usuarios con tanto detalle como sea posible (requisitos).

La Ingeniería de Requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional (THA, 1997).

De ahí la importancia concedida a la ingeniería de requisitos dentro de todo el ciclo de desarrollo del software. Como dijera (Pressman, 1997) (Davis, 1993), "el coste de un cambio en los requisitos, una vez entregado el producto, es entre 60 y 100 veces superior al coste que hubiera representado el mismo cambio durante las fases iniciales de desarrollo"

La ingeniería de requisitos es todavía una disciplina inmadura. De hecho, no hay una definición universalmente aceptada.

En (Hsia, 1993) aparece la siguiente:

"*Todas las actividades relacionadas con la identificación y documentación de las necesidades de clientes y usuarios, la creación de un documento que describe la conducta externa y las restricciones asociadas de un sistema que satisfará dichas necesidades, el análisis y validación del documento de requisitos para asegurar consistencia, compleción y viabilidad y la evolución de las necesidades."*

En (Christel & Kang, 1992) se pueden encontrar las siguientes:

"*Aplicación disciplinada de principios científicos y técnicas para desarrollar, comunicar y gestionar requisitos."*

"*Es el proceso sistemático de desarrollar requisitos mediante un proceso iterativo y cooperativo de analizar el problema, documentar las observaciones resultantes en varios formatos de representación y comprobar la precisión del conocimiento obtenido."*

En (IEEE , 1990) aparece la siguiente definición:

"*Comprende el proceso de estudiar las necesidades del usuario para llegar a una definición de requisitos de sistema, y el de estudiar y refinar los requisitos de sistema, hardware o software."*

En síntesis, la ingeniería de requisitos puede considerarse como un proceso de intercambio mutuo entre desarrolladores y clientes, con el propósito de que los desarrolladores de software se aprehendan de las necesidades reales de los usuarios para quienes realizarán la aplicación y profundicen en ellas. Por eso es que se plantea que una buena Ingeniería de Requisitos, depende en su mayor parte, de que exista una

estrecha comunicación entre las partes involucradas (cliente-desarrollador) (Goguen, 1994)

Una posible visión de la ingeniería de requisitos es considerarla como un proceso de construcción de una especificación de requisitos en el que se avanza desde especificaciones iniciales, que no poseen las propiedades oportunas, hasta especificaciones finales completas, formales y acordadas entre todos los participantes (Pohl, 1997).

1.4.1 PROCESOS DE LA INGENIERIA DE REQUISITOS.

El proceso de Ingeniería de Requisitos se divide en administración de requisitos y desarrollo de requisitos, y este a su vez se subdivide en obtención, análisis, especificación y validación.

Figura 1.1: Procesos de la Ingeniería de Requisitos.

Obtención

El proceso de obtención de requisitos es el primer paso de búsqueda de información que incluye el conjunto de acciones que debemos desarrollar para entender a nuestros usuarios, conociendo, comprendiendo y descubriendo sus necesidades.

En la obtención se identifican todas las fuentes de requisitos implicadas en el sistema y, en función de las características del entorno y del proyecto se emplean las técnicas más apropiadas para la identificación de las necesidades que deben satisfacerse.

Análisis

Una vez obtenida la información necesaria del entorno, es necesario sintetizarla, darle prioridades, analizar posibles contradicciones o conflictos, descomponer el sistema y distribuir las necesidades de cada parte, delimitar los límites del sistema y definir su interacción con el entorno.

Especificación

Cuando ya se conoce el entorno del cliente y sus necesidades, es necesario plasmarlas en forma de requisitos en los documentos que sirven de base de entendimiento y acuerdo entre cliente y desarrollador, y que establecerán tanto la guía desarrollo como los criterios de validación del producto final.

Documentar los requisitos es la condición más importante para gestionarlos correctamente.

Verificación y validación

Los requisitos deben ser formal y técnicamente correctos (verificación), y satisfacer las necesidades del sistema, sin omitir ninguna ni incluir funcionalidades innecesarias (validación).

El significado de estos dos términos genera confusiones habitualmente. El criterio básico que los diferencia es que verificación se refiere a la calidad formal, en este caso de los documentos de requisitos (no son ambiguos, no son incompletos, son posibles, verificables, etc.) y validación comprende la adecuación en el entorno de producción, en el caso de la documentación de requisitos, la conformidad por parte del cliente de que reflejan lo que él quiere.

1.4.2 IMPORTANCIA DE LA INGENIERIA DE REQUISITOS.

La parte más difícil en la construcción de sistemas software es decidir precisamente qué construir. Es esencial comprender a la perfección los requisitos del software.

Ninguna otra parte del trabajo conceptual es tan ardua como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas software. Ninguna otra parte del trabajo puede perjudicar tanto el resultado final si se realiza de forma errónea. Ninguna otra parte es tan difícil de rectificar posteriormente.

Para que un esfuerzo de desarrollo de software tenga éxito es imprescindible que se haya llevado a cabo una correcta Ingeniería de Requisitos. Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, decepcionará al usuario y desprestigiará al que lo ha desarrollado. De ahí la importancia que posee esta fase en el ciclo de desarrollo de un producto software.

1.4.3 LOS DEFECTOS MÁS COMUNES EN LA INGENIERÍA DE REQUISITOS Y SUS CONSECUENCIAS.

Los defectos que con más frecuencia suelen ocurrir durante la Ingeniería de Requisitos son:

Implicación insuficiente del cliente: Algunos clientes no comprenden la importancia de trabajar con rigor en la obtención de los requisitos, para garantizar la calidad de los resultados. También es frecuente que los desarrolladores prefieran comenzar a trabajar en la construcción del sistema, porque les resulta más atractivo que reunirse con el cliente.

Hay situaciones en las que resulta difícil encontrar representantes del cliente que conozcan a fondo el problema, o que por tratarse de un sistema o negocio nuevo, nadie en el entorno del cliente tenga claras todas las funcionalidades que se necesitan.

Requisitos crecientes y cambiantes: Independientemente del punto del ciclo de vida en que nos encontremos, el sistema cambiará y la tendencia al cambio persistirá a lo largo de todo el ciclo de vida. (Prentice-Hall, 1980)

Es normal que los requisitos evolucionen durante el desarrollo del sistema, pero los cambios deben partir de una descripción inicial correcta, y gestionarse convenientemente, midiendo su impacto en la planificación del proyecto, y consensuándolo con todos los participantes.

La evolución de los requisitos durante el desarrollo de los proyectos puede incrementar o modificar funcionalidades ya implementadas, desbordando los costes y agendas planificados.

Partir de una especificación de requisitos incompleta incrementará el número de modificaciones que sufrirá el proyecto durante el desarrollo. Si los desarrolladores han diseñado un sistema que no corresponde con las expectativas del cliente, la introducción sistemática (generalmente con agendas apretadas, o sin modificar las agendas iniciales), generará parches de programación, con inserción de código adicional que puede trastocar principios básicos de diseño y degradar la arquitectura del sistema obteniendo finalmente un producto con serias deficiencias técnicas.

Requisitos ambiguos: La ambigüedad es un defecto habitual de las descripciones de requisitos. Si lectores diferentes obtienen interpretaciones diferentes, o si un lector puede interpretar los requisitos de formas diferentes, éstos son ambiguos. La ambigüedad crea expectativas diferentes entre las partes del proyecto, y hace que los desarrolladores programen funcionalidades que no se ajustan a lo que los usuarios necesitan. La consecuencia inevitable de este problema es la reprogramación. La reprogramación puede consumir más del 40% del coste total de un desarrollo y se ha estimado que hasta un 85% de las revisiones pueden deberse a errores en los requisitos. Para evitarla hay que confirmar que se comparte la visión obtenida con la que tienen las diferentes fuentes de requisitos, y que los distintos participantes obtienen la misma interpretación de la documentación de requisitos.

Requisitos innecesarios: Es frecuente la tendencia de algunos desarrolladores a incluir funcionalidades que no figuran en la especificación de requisitos, con la suposición de que los usuarios lo agradecerán. Muchas veces los usuarios no les encuentran utilidad y quedan finalmente programadas pero sin uso, suponiendo un coste de desarrollo innecesario.

Las sugerencias y posibilidades aportadas por el equipo de desarrollo pueden descubrir mejoras importantes para el cliente o los usuarios, pero no deben implementarse sin consultarlas y validarlas previamente.

Desde el punto de vista del equipo de desarrollo la mejor perspectiva es respetar la sencillez y funcionalidad, y no ir más allá de los requisitos, sin la aprobación del cliente.

También es frecuente que el cliente pida funcionalidades que a primera vista pueden parecer necesarias, pero que en realidad no añaden funcionalidad al producto, y que sin embargo suponen un esfuerzo importante de desarrollo. Identificar estas funcionalidades, y pensar dos veces si realmente se necesitan puede ahorrar trabajo innecesario.

Requisitos insuficientes (mínimos): Muchas veces el cliente tiene tan sólo el concepto general del producto que desea, y no comprende por qué es tan importante detallar los requisitos. La tentación en estos casos es partir de una descripción mínima, o incluso de una explicación verbal, e ir preguntando y revisando a los programadores conforme el desarrollo avanza.

Esta forma de trabajo puede resultar apropiada sólo para la construcción de sistemas experimentales o prototipos, pero en general suele terminar con la frustración de los desarrolladores y el desconcierto y desesperación del cliente.

Este planteamiento también genera la situación muy frecuente de contar a los desarrolladores la idea general de un nuevo producto, para pedirles una estimación del tiempo necesario para desarrollarlo.

Normalmente la visión general, sin descender a los detalles que implica, genera previsiones optimistas que terminarán desbordadas al descubrir durante el desarrollo las implicaciones que pasan inadvertidas en la concepción inicial.

Las estimaciones prematuras, basadas en información limitada pueden fácilmente desbordarse en más del doble. Siempre que sea preciso ofrecer valoraciones previas es conveniente ofrecer varias posibilidades (mejor caso, caso probable, peor caso), o incluir un porcentaje posible de error probable.

Omisión de las necesidades de algunos grupos de usuarios:
Entre los usuarios de un sistema es frecuente que se incluyan grupos de personas con necesidades diferentes, que empleen funcionalidades distintas, e incluso que presenten diversos perfiles de experiencia y conocimientos.
Al identificar las posibles fuentes de requisitos hay que localizar todos los posibles usuarios y obtener información de sus características, necesidades y expectativas.

Figura 1.2: Los defectos más comunes en los requisitos y sus consecuencias.

Todos estos errores en los requisitos se comportan como una enfermedad contagiosa que siempre repercute en todas las fases del proyecto.
Estimación: No es posible estimar con rigor costes y recursos necesarios para desarrollar algo que no se conoce.

Planificación: No se puede confiar en la planificación para el desarrollo de algo que no se sabe bien como es.

Diseño: Los errores en requisitos, las modificaciones frecuentes, las deficiencias en restricciones o futuras evoluciones, producen arquitecturas que más tarde se confirmarán como erróneas y serán modificadas.

Construcción: Las deficiencias en los requisitos obligan a programar en ciclos de prueba y error que derrochan horas y paciencia de programación sobre patrones de recodificación continua y programación heroica.

Validación y verificación: Terminado el desarrollo del sistema, si las especificaciones tienen errores de bulto, o peor aún, no están reflejadas en una especificación de requisitos, no ser posible validar el producto con el cliente.

Figura 1.3: Causas del fracaso de los proyectos.

1.4.4 BENEFICIOS DE UNA BUENA INGENIERIA DE REQUISITOS.

Acuerdo entre desarrolladores, clientes y usuarios sobre el trabajo que debe realizarse. Unos requisitos bien elaborados y validados con el cliente evitan descubrir al terminar el proyecto que el sistema no era lo que se pedía.

Acuerdo entre desarrolladores, clientes y usuarios sobre los criterios que se emplearán para su validación. Resulta muy difícil demostrar al cliente que el producto desarrollado hace lo que el pidió si su petición no está documentada y validada por él.

Base objetiva para la estimación de recursos (costo, personal en número y competencias, equipos y tiempo) Si los requisitos no comprenden necesidades reales, las estimaciones no dejan de ser meras apuestas. Las estimaciones en el fondo son cálculos de probabilidad que siempre implican un margen de error; por esta razón disponer de la mayor información posible reduce el error.

Concreción de los atributos de calidad (ergonomía, mantenibilidad, etc.) Más allá de funcionalidades precisas, los requisitos recogen atributos de calidad necesarios que en ocasiones son desconocidos por los desarrolladores, produciendo finalmente sistemas sobredimensionados o con serias deficiencias de rendimiento.

Eficiencia en el consumo de recursos: reducción de la recodificación, reducción de omisiones y malentendidos. Tener un conocimiento preciso de lo que hay que hacer evita la prueba y error, repetición de partes ya desarrolladas, etc.

1.4.5 OBTENCIÓN DE REQUISITOS. TECNICAS DE ELICITACION.

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema (Díez, 2001). El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de

requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa. A continuación se presentan un grupo de técnicas que de forma clásica han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software.

Entrevistas: resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización (Durán, Bernáldez, Ruíz, & Toro, 1999), (Pan, Zhu, & Jonhson, 2001)
Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados. A pesar de que las entrevistas son esenciales en el proceso de la captura de requisitos y con su aplicación el equipo de desarrollo puede obtener una amplia visión del trabajo y las necesidades del usuario, es necesario destacar que no es una técnica sencilla de aplicar (Pan, Zhu, & Jonhson, 2001). Requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado. Aquí desempeña un papel fundamental la preparación de la entrevista.

JAD (Joint Application Development/Desarrollo conjunto de aplicaciones): esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes (IBM, 1997). Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación WYSIWYG (What You See Is What You Get, lo que ve es lo que obtiene), es decir, durante la aplicación de la técnica se trabajará sobre lo que se generará. Tras una fase de preparación del JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de

(Raghavan, Zelesnik, & Ford, 1994) ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación.
Durante la sesión se discute en grupo sobre estos temas, llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema. Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados.

Brainstorming (Tormenta de ideas): es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre (Raghavan, Zelesnik, & Ford, 1994). Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador.
Como técnica de captura de requisitos es sencilla de usar y de aplicar, contrariamente al JAD, puesto que no requiere tanto trabajo en grupo como éste. Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

Concept Mapping: Los concept maps (Pan, Zhu, & Jonhson, 2001) son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy subjetivos y pueden llegar a ser ambiguos en casos complejos, si no se acompaña de una descripción textual.

Sketches y Storyboards: Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario (sketches). Estos sketches pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (storyboard).

Casos de Uso: Aunque inicialmente se desarrollaron como técnica para la definición de requisitos (Jacobson, 1995), algunos autores proponen casos de uso como técnica para la captura de requisitos (Pan, Zhu, & Jonhson, 2001) y (Liu & Yu, 2001). Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas, etc.) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.
La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria (Vilain, Schwabe, & Sieckenius, 2002) y (Insfrán, Pastor, & Wieringa, 2002) si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades (UML, 2001)

Cuestionarios y Checklists: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (Checklist). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.

Comparación de terminología: Uno de los problemas que surge durante la obtención de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de

terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos, misma terminología para diferentes conceptos o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos. (Pan, Zhu, & Jonhson, 2001)

Existen más técnicas para la captura de requisitos (análisis de otros sistemas, estudio de la documentación, etc.), incluso también es común encontrar alternativas que combinen varias de estas técnicas (Pan, Zhu, & Jonhson, 2001). Sin embargo, las presentadas ofrecen un conjunto representativo de las más utilizadas en el mundo.

1.4.6 PROPUESTA METODOLÓGICA PARA LA OBTENCIÓN DE REQUISITOS.

La propuesta metodológica para la obtención de requisitos que propone Amador Durán Toro, Doctor en Informática de la Universidad de Sevilla plantea una serie de tareas a realizar y productos a obtener (tanto internos como externos o entregables). Esta metodología es una evolución de la presentada inicialmente en (Durán, 1998).

En la propuesta se contemplan las siete tareas que se relacionan a continuación, con un posible orden de realización orientativo.

Tarea 1: Obtener información sobre el dominio del problema y el sistema actual

El objetivo principal de esta tarea es conocer el dominio del problema lo mejor posible. Las razones para ello son varias. En primer lugar, el ingeniero de requisitos debe conocer el lenguaje de clientes y usuarios para poder comunicarse con ellos. Cada dominio de problemas posee un vocabulario propio que es necesario conocer (Rombach, 1990) y (Brackett, 1990). En segundo lugar, el ingeniero de requisitos debe evitar utilizar sus propios esquemas y categorías mentales a la hora de obtener la información, ya que esto puede dificultar la comunicación (Goguen, 1994). Debe aprender a pensar en los términos en los que lo hacen clientes y usuarios. En tercer lugar, mantener sesiones de obtención de requisitos sin conocer el dominio del problema puede provocar que los malentendidos o las preguntas continuas sobre el significado de los términos empleados por clientes y usuarios hagan que la confianza

hacia el equipo de ingeniería de requisitos se vea deteriorada enormemente, provocando cierta reticencia a participar e involucrarse en el proyecto, que es la primera causa de fracaso de los proyectos software (TSG , 1995) y (Blaha & Premerlani, 1998).

Tarea 2: Preparar y realizar las sesiones de elicitación/negociación

El objetivo de esta tarea es conocer las necesidades de clientes y usuarios y resolver los conflictos identificados en las actividades de análisis de iteraciones previas. Es la tarea más crítica, ya que en ella es donde existe una mayor interacción personal entre el equipo de ingeniería de requisitos y los clientes y usuarios, por lo que una adecuada selección de los participantes es crucial (TSG , 1995).

Tarea 3: Identificar/revisar los objetivos del sistema

El objetivo de esta tarea es conocer por qué se acomete el desarrollo, y por lo tanto conocer qué objetivos se esperan alcanzar mediante el sistema software a desarrollar. En la primera iteración se realizará una primera identificación de los objetivos. En las iteraciones posteriores puede que sea necesario revisarlos si se han identificado conflictos que les afecten. La idea básica es ir obteniendo los requisitos como un refinamiento de los objetivos, de forma que la existencia de un requisito esté siempre (Blaha & Premerlani, 1998).

Tarea 4: Identificar/revisar los requisitos de información

El objetivo de las tareas 4, 5 y 6 es identificar, o revisar en función de posibles conflictos, los requisitos a partir de la información obtenida en las tareas anteriores. La división en tres tareas se ha realizado por simplicidad, ya que habitualmente se realizan en paralelo.

En esta tarea en concreto se deben identificar o revisar los requisitos de almacenamiento de información que deberá satisfacer el sistema. Normalmente estos requisitos son la respuesta a la pregunta ¿qué información, relevante para los objetivos de su negocio, deberá almacenar el sistema?

Tarea 5: Identificar/revisar los requisitos funcionales

En esta tarea se deben identificar o revisar los requisitos funcionales que deberá satisfacer el sistema, para lo que se ha optado por la utilización de los casos de uso (Jacobson, 1993). Estos requisitos suelen obtenerse como respuesta a la pregunta ¿qué debe hacer el sistema con la información almacenada para alcanzar los objetivos de su negocio? o ¿qué debe permitir el sistema hacer a los usuarios con la información almacenada?

Además de los casos de uso, en esta tarea es necesario identificar y describir a los actores del sistema (Jacobson, 1993). Otro aspecto importante es la determinación del ámbito del sistema (Lilly, 2000), es decir qué aspectos serán responsabilidad del sistema y qué aspectos se gestionarán manualmente o por otro procedimiento. La utilización de los diagramas de casos de uso (Booch, 1999), permite de forma sencilla especificar claramente qué queda dentro y qué queda fuera del ámbito del sistema.

Tarea 6: Identificar/revisar los requisitos no funcionales

En esta tarea se deben identificar o revisar los requisitos no funcionales del sistema, normalmente relacionados con aspectos técnicos o legales: comunicaciones, interfaces con otros sistemas, fiabilidad, entorno de desarrollo, portabilidad, etc.

Tarea 7: Priorizar objetivos y requisitos

En esta última tarea se deben asignar prioridades a los objetivos y requisitos estableciendo su importancia y su urgencia, de forma que en el caso de que se desarrolle incrementalmente se tengan los criterios suficientes para saber qué requisitos deben implementarse en cada versión que se vaya entregando. (Toval & Nicolás, 1999)

1.4.7 DEFINICION DE REQUISITOS.

También para la actividad de definición de requisitos en el proceso de ingeniería de requisitos hay un gran número de técnicas propuestas. A continuación se describen brevemente las más relevantes para este trabajo.

Lenguaje natural: Resulta una técnica muy ambigua para la definición de los requisitos. Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello. A pesar de que son muchos los trabajos que critican su uso, es cierto que a nivel práctico se sigue utilizando.

Glosario y ontologías: La diversidad de personas que forman parte de un proyecto software hace que sea necesario establecer un marco de terminología común. Esta necesidad se vuelve más patente en los sistemas de información web puesto que el equipo de desarrollo en ellas suele ser más interdisciplinario (Koch, 2001). Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema. En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.

Plantillas o patrones: Esta técnica, recomendada por varios autores (Durán, Bernáldez, Ruíz, & Toro, 1999) y (Escalona, Torres, & Mejias, 2002)), tiene por objetivo describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso.

Escenarios: La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos (Liu & Yu, 2001). La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo (Weidenhaupt, Pohl, Jarke, & Haumer, 1999). El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales de sistema (Lowe & Hall, 1999).

Casos de uso: Como técnica de definición de requisitos es como más ampliamente han sido aceptados los casos de uso. Actualmente se ha propuesto como técnica básica del proceso RUP (Kruchten, 1998). Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos (Díez, 2001) y (Vilain, Schwabe, & Sieckenius, 2002) y (Insfrán, Pastor, & Wieringa, 2002), por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.

Lenguajes Formales: Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años (Peña, 1998). Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre el cliente y el analista. Por el contrario, es la representación menos ambigua de los requisitos y la que más se presta a técnicas de verificación automatizadas. (Escalona & Koch, 2002)

Para concluir se puede enfatizar que la técnica más usada en la actualidad por la mayor parte de los equipos desarrolladores de software es Plantillas o Patrones, ya que a además de ofrecer características importantes sobre las necesidades funcionales del sistema y de eliminar ambigüedades del lenguaje natural, favorece el intercambio entre el cliente y el analista.

1.5 CONCLUSIÓN.

Existen normas formalmente establecidas a nivel mundial y abundante literatura científica que definen procedimientos, técnicas, lenguajes y herramientas para la Ingeniería de Requisitos, sin embargo subsisten problemas con la calidad final de los proyectos, la ejecución en tiempo y presupuesto y la satisfacción de los clientes en muchos casos provocados por deficiencias en el desarrollo y gestión de requisitos.

Existe además numerosos e importantes artículos dedicados a abordar temas relacionados con las aplicaciones bioinformáticas que se desarrollan hoy en todo el

mundo para dar solución a enfermedades, para producir fármacos, para la predicción de epítopes, entre otros aspectos de interés, pero ninguno aborda el tema de cómo lograr la calidad en este tipo de software.

En la literatura se considera la Ingeniería de Requisitos como un proceso vital en la producción de software del cual depende en gran medida el éxito de los proyectos; sin embargo, no se encuentran referencias de aplicación y adaptación de este proceso en proyectos de Bioinformática.

De lo anterior se infiere la necesidad de organizar y unificar todo el conocimiento que se encuentra disperso en la web para ponerlo en manos de aquellos que se dedican a la producción de aplicaciones bioinformáticas y de esta forma contribuir a su fácil consulta. También se infiere la necesidad de investigar acerca de cómo se llevan a cabo en las aplicaciones bioinformáticas, etapas tan importantes dentro del ciclo de desarrollo de cualquier software, como es la Ingeniería de Requisitos de forma tal que se obtengan productos de calidad.

CAPITULO 2: UNA PROPUESTA DE ORGANIZACIÓN DEL CONOCIMIENTO PARA LOGRAR FIABILIDAD EN LAS APLICACIONES BIOINFORMATICAS.

2.1 INTRODUCCIÓN.

En el presente capítulo se tocan a fondo temas inherentes al logro de la calidad en las aplicaciones bioinformáticas, así como las características que distinguen a estas aplicaciones del resto de los software y las principales causas que provocan el fracaso de proyectos relacionados con esta rama tan novedosa aún, de la ciencia. También se maneja en este capitulo cuál es el factor de calidad determinante en estos software y como evaluarlo en una aplicación después de concluida. Se plantea además, una propuesta de organización del conocimiento con el objetivo de lograr la fiabilidad en estas aplicaciones.

2.2 CARACTERÍSTICAS ESPECÍFICAS DE LAS APLICACIONES BIOINFORMÁTICAS.

A partir del conocimiento obtenido de la consulta de múltiples bibliografías, de entrevistas realizadas a conocedores del tema fundamentalmente del Polo Científico, y el estudio del funcionamiento de numerosas aplicaciones bioinformáticas se pueden plantear una serie de características comunes a este tipo de proyectos y que tienen impacto en los procesos inherentes a la obtención de requisitos. Son señaladas y argumentadas a continuación:

1. Gran complejidad algorítmica:

   Los proyectos hacen uso de técnicas heurísticas, de leyes y teorías de la física estadística y cuántica y de la programación y procesamiento paralelo y distribuido. También son usados métodos matemáticos como el tratamiento de secuencias a modo de Vectores Unidimensionales y Diagramas de Recurrencia aplicados por ejemplo al estudio de proteínas, mapas iterativos lineales y no lineales para el estudio de sistemas en tiempo discreto y ecuaciones diferenciales para sistemas en tiempo continuo. Además es frecuente la utilización de modelos matemáticos en la investigación biológica: modelo de Michaelis-Menten para la catálisis enzimática,

modelo de Lotcka-Voltera (Presa-depredador en Ecología), modelos de Luria-Delbruck para el proceso de selección mutación en bacterias, modelo de Hodgkin-Huxley para la neurona y modelo de Geoge Bell para la respuesta Inmune Humoral entre otros (Febles, 2006) (León, 2006).

2. Trabajo con ingentes cantidades de datos:

Se generan y manejan cantidades masivas de datos biológicos. Por solo citar algunos ejemplos, en la actualidad se conoce la secuencia de más de un millón y medio de proteínas, más cien genomas y la estructura tridimensional de más de 20 mil proteínas. Bases de datos de secuencias de ADN almacenan datos en el orden de los millones y las cifras aumentan considerablemente cada año. GenBank por ejemplo, almacenaba en diciembre de 1997, $1.26 * 10^9$ bases, en abril de 1999, $2.57 * 10^9$, en septiembre del 2000 el número aumentó a $1.0 * 10^{10}$ y ya en agosto de 2005 almacenaba más de $8.0 * 10^{10}$ bases. El conocimiento científico acumulado a lo largo de las últimas décadas se encuentra disperso en más de 12 millones de artículos (Febles, 2007) (Mount, 2004) (NCBI, 2006).

3. Diferencias en cuanto a lenguaje, formación y visión entre clientes y equipo de desarrollo:

Los proyectos de desarrollo de software bioinformático son generalmente caracterizados además por la diferencias en cuanto a lenguaje, formación y visión entre clientes y equipo de desarrollo. Los profesionales de las ciencias biológicas y las informáticas han recibido años de preparación en ramas diferentes y han adoptado un lenguaje científico propio de la ciencia respectiva. Cada profesional ha hecho suyos conceptos con un nivel elevado de conocimientos que los respaldan y en ocasiones presumen como trivial su comprensión por parte del interlocutor. Este hecho supone una gran barrera para el entendimiento mutuo. Además de este conocimiento explícito, dichos especialistas han obtenido a lo largo de su desempeño profesional e investigativo considerables cantidades de conocimiento tácito que generalmente los ingenieros de software deben obtener y modelar en forma de requisitos.

2.3 INGENIERÍA DE REQUISITOS EN APLICACIONES BIOINFORMÁTICAS.

Una encuesta realizada en (tesis de maestría de Karina Pérez Teruel, 2007) a 25 desarrolladores de software BI arrojó que en un 76% de los casos, los proyectos en los que han participado han excedido el tiempo y presupuesto asignados (Figura 2.1).

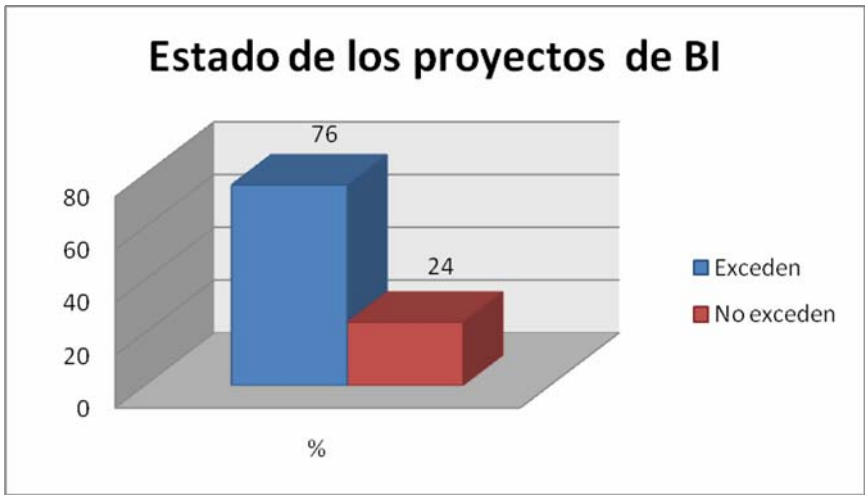


Figura 2.1: Estado de los proyectos de BI en relación con el tiempo y presupuesto asignados.

La prioridad asignada por parte de los encuestados a los requisitos dentro del proceso de desarrollo de software BI fue alta en un 88%, media en un 8% y baja en un 4% de los casos (Figura 2.2).

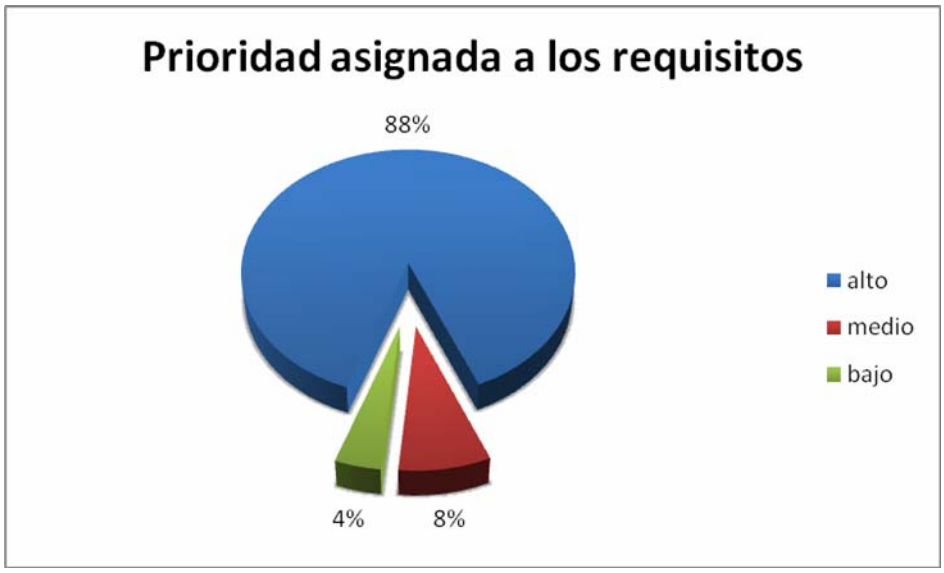


Figura 2.2: Nivel de prioridad asignada a los requisitos dentro del proceso de desarrollo de software BI.

En el mismo diagnóstico se midió el nivel de impacto de factores que podrían provocar un exceso en el tiempo y presupuesto en proyectos de BI (Figura 2.1). Los cinco factores con mayor incidencia son provocados por ineficiencias en los procesos de IR. Ellos son:

- Falta de comunicación con usuarios y/o clientes - 64%.
- Requisitos y especificaciones incompletas - 40%.
- Requisitos y especificaciones cambiantes - 64%.
- Falta de claridad en los objetivos - 52%.
- Estimaciones irreales - 36%.

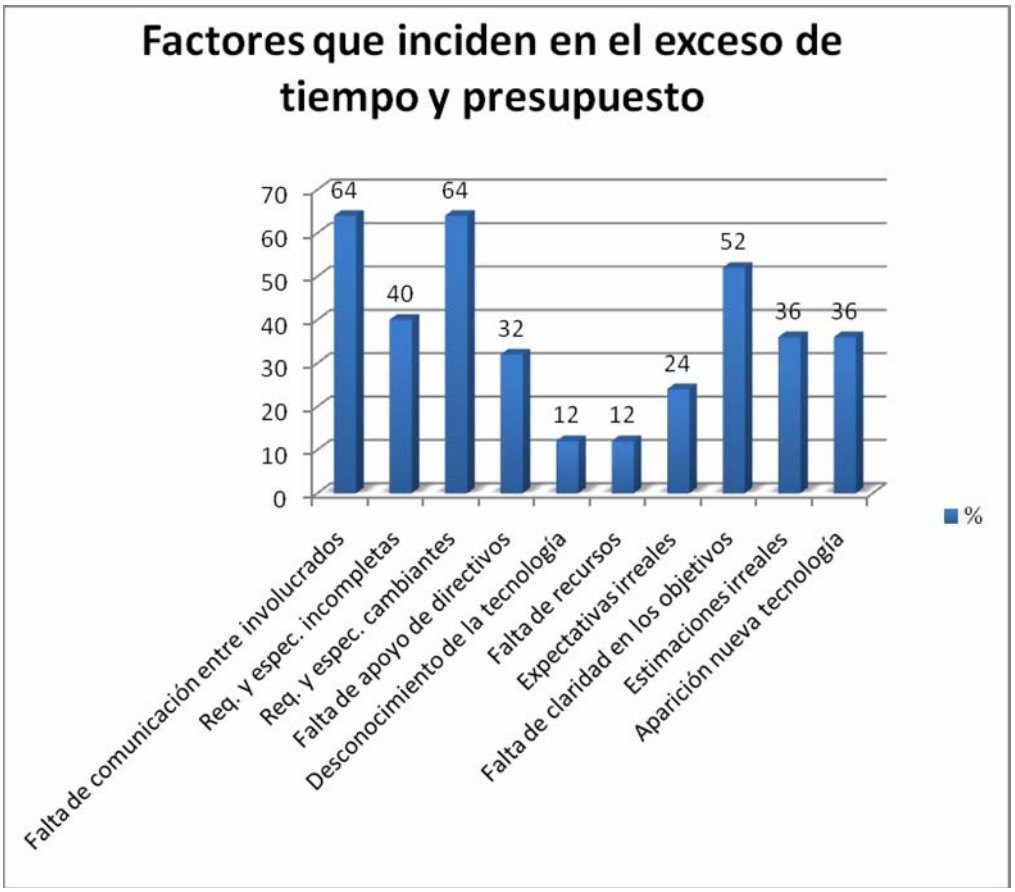


Figura 2.3: Factores que influyen en el exceso de tiempo y presupuesto de las aplicaciones BI.

El contrastar el dato de que el 88% de los encuestados asigna un nivel alto de prioridad a los requisitos, con el hecho de que los factores que con mayor impacto inciden en que los proyectos de BI excedan el tiempo y presupuesto asignados, están fuertemente relacionados a la IR, permite concluir que los procedimientos o técnicas usados por los encuestados no son los idóneos para este tipo de proyecto.

2.4 APLICACIONES BIOINFORMÁTICAS. FACTORES DE CALIDAD.

Los avances de la bioinformática han sido mayores a medida que se han ido acumulando grandes volúmenes de información obtenidos por nuevas tecnologías, pero, paralelamente, también ha crecido en complejidad la tarea de procesar, analizar y

almacenar toda esta información, lo cual supone un cuello de botella en las tareas de investigación, ya que el ritmo de producción de datos es mucho mayor que la funcionalidad de las técnicas de análisis existentes. Esto ha creado la necesidad inmediata de desarrollar herramientas eficientes especializadas para el análisis y comprensión de datos biológicos.

Para el desarrollo y el uso de estas aplicaciones Bioinformáticas la plataforma preferida es Linux, este ambiente es idóneo por las ventajas de seguridad y gratuidad que lo caracterizan, así como otros sistemas operativos emparentados con Unix. Perl es un lenguaje de programación frecuentemente usado para el desarrollo de pequeñas aplicaciones bioinformáticas y las bases de datos relacionales son sistemas estándar de almacenamiento, análisis y consulta de datos.

Los tipos de herramientas usadas más frecuentemente en aplicaciones Bioinformática es un objetivo muy amplio que abarca técnicas y métodos enfocados a analizar y comparar secuencias de ácidos nucleicos, analizar y anotar genomas, predecir y comparar la estructura y la función de proteínas, diseñar fármacos optimizados para su interacción con centros activos de enzimas o receptores o con ácidos nucleicos, etc.

Actualmente en el mundo no son muchas las empresas que se dedican específicamente a la construcción de software bioinformático. En realidad los proyectos bioinformáticos surgen como respuesta a determinadas problemáticas o trabas que se le presentan a grupos científicos, grupos universitarios en sus investigaciones cotidianas; es decir, vienen siendo como tareas secundarias dentro de sus proyectos principales.

Algunos ejemplos de aplicaciones bioinformáticas son las siguientes:

- Desarrollo de una plataforma de análisis de datos en bioinformática basada en Matlab. Análisis y visualización de datos de expresión génica. (Pascual Montano, 2005)
- El paquete de herramientas bioinformáticas del Centro de Investigación Príncipe Felipe de Valencia (CIPF), conocido como Gepas, permite analizar diariamente las características de los genes del genoma presente en más de

300 experimentos realizados en laboratorios de todo el mundo. (Martínez, 2007)

- Desarrollo de una aplicación para la Extracción de conocimiento en Bases de Datos moleculares, Oswaldo Trelles, Universidad de Málaga, España.
- Bioinformatics for bacterial genome projects Joao Setúbal, UNICAMP, Brasil.
- Desarrollo de aplicaciones bioinformáticas para el análisis de la reactividad enzimática, Xavier Messeguer, Universidad Politécnica de Cataluña (UPC), España.
- Proyecto para solucionar algunos problemas de interferencia en biología computacional, Elizabeth Tapia, Universidad Nacional Rosario, Argentina.
- Proyecto de Anotación del Genoma Aspergillus Nidulans, Miguel Angel Peñalva, Instituto Nacional de Bioinformática (INB), España.
- Proyecto de Simulación de Dinámica Molecular, Instituto Nacional de Bioinformática (INB), España.
- Bioinformática aplicada a la exportación frutícola, Ricardo Baeza, Universidad de Chile, Chile.
- Bioinformática en el CNB: investigación, servicios, y coordinación José Ramón Valverde, UAM, España.
- Bioinformática y Genómica Funcional de la Vida, Mauricio Canales, Universidad Técnica Federico Sta. María, Chile

Un aspecto que debiera ser analizado también es la “calidad” de los productos de software que son utilizados o elaborados para el campo de la bioinformática. El término calidad del software se interpreta de diferentes maneras. McCall especifica una serie de factores de calidad (ISO 9126), cada uno de esos factores los subdivide en criterios, teniendo asociado cada uno de ellos una métrica (McCall, 1977). En tabla siguiente se

muestran los factores generales de calidad que fueron analizados por un grupo de expertos bioinformáticos para determinar un criterio de prioridad a la hora de establecer una estrategia de calidad para la disciplina. La valoración de las opiniones de los expertos indicó que uno de los factores claves para software bioinformáticos es la fiabilidad.

FACTOR	**DEFINICIÓN**
Corrección	Grado en el que un programa satisface las especificaciones y cumple los objetivos del usuario.
Fiabilidad	Grado en el que un programa se espera que realice su función con una precisión requerida.
Eficiencia	Cantidad de recursos y código requeridos por un programa para realizar una función.
Integridad	Grado en el que se controla el acceso al programa o los datos por usuarios no autorizados.
Usabilidad	Esfuerzo necesario para aprender, operar, preparar entradas e interpretar la salida de un programa.
Mantenibilidad	Esfuerzo requerido para localizar y corregir un error en un programa en funcionamiento.
Facilidad de prueba	Esfuerzo requerido para probar un programa (para garantizar que realiza la función deseada).
Flexibilidad	Esfuerzo requerido para modificar un programa en funcionamiento.
Portabilidad	Esfuerzo requerido para transferir un programa de una configuración hardware o entorno software a otro.
Reusabilidad	Grado en el que un programa se puede utilizar en otras aplicaciones
Interoperatividad	Esfuerzo requerido para acoplar un sistema con otro.

Tabla 2.1: Factores generales de calidad propuestos por McCall.

2.5 ATRIBUTOS DE LA FIABILIDAD.

La **fiabilidad** se subdivide en cuatro subcaracterísticas o atributos:

- **Madurez:** la capacidad del producto software para evitar fallos provocados por errores en el software.

- **Tolerancia a fallos:** la capacidad del producto software para mantener un nivel de rendimiento determinado en caso de defectos en el software o incumplimiento de su interfaz.

- **Recuperabilidad:** la capacidad del producto software para restablecer un determinado nivel de rendimiento y recuperar los datos afectados directamente en caso de ocurrir un fallo.

- **Conformidad:** la capacidad del producto software para adaptarse a estándares, convenciones y regulaciones referidas a la fiabilidad. (Alarcos, 1996)

Madurez

Atributo del software que tiene que ver con la frecuencia de fracaso por fallos en el software.

En países como la India, donde la industria del software es una de las más desarrolladas del mundo y sus exportaciones superan los cinco mil millones de dólares anuales, los productores de software plantean: que la clave está en la calidad y sobre todo en la madurez de sus productos software. (Ventura Miranda, 2002)

A medida que los sistemas informáticos crecen en complejidad, las pérdidas causadas por fallos en dichos sistemas son cada vez mayores. Liberar a un sistema de dichos fallos es una forma de controlar la calidad del mismo y colocarlo en el mercado con la completa seguridad de que son programas que se pueden usar sin riesgo alguno. (Bedini G., 2000)

Son indicadores de madurez: el tiempo medio entre fracasos, la densidad de fallos del producto, la estabilidad del producto, la densidad de fallo, la densidad de prueba, la cobertura de prueba.

Atributos asociados a Madurez
La madurez se mide en función del número de versiones comerciales que han salido al mercado y en función del tiempo que una versión permanece en activo, es decir, vamos a medir un tiempo medio entre versiones comerciales del componente.

− *Volatilidad:* Este atributo indica el tiempo que dura la versión en el mercado. Mide el promedio de tiempo entre cada nueva versión comercializada y su versión predecesora. Nos referiremos a él como "Tiempo medio entre versiones". Se utiliza una variable de tipo Tiempo, que indique el tiempo medio entre versiones.
− *Evolucionabilidad:* El número de versiones puede dar una indicación de la madurez de un producto, y de como ha ido evolucionando desde que estaba disponible su primera versión. Puede ser interesante en conjunción con la volatilidad para indicarnos si debemos esperar posibles cambios en un periodo corto de tiempo. Este atributo se mide mediante un entero el número absoluto de versiones del componente que han sido distribuidas comercialmente.
− *Fallos Eliminados:* Una medida de madurez es el número de errores corregidos en cada versión. Aunque en principio un alto número de errores corregidos puede parecer que estabiliza la nueva versión, la experiencia de los autores en el desarrollo de productos software nos lleva a afirmar que el número de errores corregidos es muchas veces un buen indicador del número de errores que quedan por descubrir.
Se utiliza una métrica de tipo Entero para indicar el número medio de errores eliminados en las últimas versiones. (Bertoa & Vallecillo, 2002)

Tolerancia a fallos
Atributo del software que refiere su capacidad de mantener un nivel especificado de funcionamiento en los casos de fallo del software o de las infracciones de su interfaz.
Son indicadores de tolerancia a fallos: el número de perturbaciones, vulnerabilidad, el

valor de integridad, el porcentaje de interrupción, porcentaje de detección de errores operacionales o de entrada.

Grados de tolerancia a fallos

− *Tolerancia completa:* El sistema sigue funcionando, al menos durante un tiempo, sin perder funcionalidad ni prestaciones.

− *Degradación aceptable:* El sistema sigue funcionando con una pérdida parcial de funcionalidad o prestaciones hasta la reparación del fallo.

− *Parada segura*: El sistema se detiene en un estado que asegura la integridad del entorno hasta que se repare el fallo.

El grado de tolerancia de fallos necesario depende de la aplicación. (de la Puente, 1997)

Recuperabilidad

La subcaracterística de recuperabilidad tiene que ver con la capacidad de reestablecer su nivel de funcionamiento y recuperar los datos directamente afectados en caso de un fallo y durante el tiempo y el esfuerzo necesario para ello.

Son indicadores de recuperabilidad: el tiempo medio para reparar, el porcentaje de recuperación automática, el tiempo medio de recuperación, el tiempo medio de interrupción, el tiempo medio de reinicio, el tiempo medio de fallo.

La Recuperabilidad (datos, procesos, tecnología), depende de una serie de atributos que pueden estar presentes o no en su diseño, indicando los métodos que se utilizan para implementarlos.

Atributos asociados a Recuperabilidad

− *Secuencialidad:* Indica si el componente es capaz de secuenciar su código y su estado, para transferirlo entre diversos agentes y recuperarlos con posterioridad. Utilizaremos una métrica de tipo presencial.

− *Persistente:* Este atributo señalará si el componente puede almacenar su estado de forma persistente. Utilizaremos una métrica de tipo Presencial.

− *Transaccional:* Este atributo debe indicar si el componente suministra alguna interfaz que permita que sus operaciones estén sujetas a transacciones. Utilizaremos una métrica de tipo Presencial.

− *Tratamiento de Errores:* Este atributo debe indicar si se realiza algún tipo de tratamiento de errores y en su caso el tipo de tratamiento de errores que lleva a cabo el componente. Se utiliza una variable Presencial que deberá indicar el nivel de tratamiento de los errores, por ejemplo, siguiendo un modelo clásico como el siguiente: detectar (los detecta pero no realiza ninguna acción), detectar y avisar (los detecta y avisa de ello) y tratar (los detecta e implementa un mecanismo de excepciones). (Vallecillo, Bertoa, & Troya, 2002)

Conformidad:

Atributos asociados a Conformidad

−*Conformidad con los requisitos: Este atributo indica cuan cerca de las necesidades reales del cliente esta el producto software.*

− *Conformidad con estándares:* Este atributo indica si el componente cumple algún estándar internacional. Utilizaremos una métrica de tipo Presencial, para indicar si cumple algún estándar y, de ser cierto, qué estándares cumple el componente.

− *Certificaciones:* De forma similar al anterior atributo, si la componente puede acreditar algún tipo de certificación, tanto por organizaciones externas o de forma interna, puede indicarlo con este atributo. Se utiliza una métrica de tipo Presencial para indicar si tiene algún tipo de certificación. (Bertoa & Vallecillo, 2002)

2.6 LA FIABILIDAD EN LAS APLICACIONES BIOINFORMÁTICAS.

Como la mayor parte de las aplicaciones bioinformáticas responden a la solución de proyectos de investigación, las bases de datos que necesitan tienen que ser lo suficientemente robustas para sustentar todo el cúmulo de información que ellas manejan con garantías de fiabilidad. Por ello crear y mantener una gran base de datos accesible a través de Internet o de cualquier red, se vuelve un objetivo común en este tipo de software. Para realizar bases de datos robustas se necesitan potentes gestores de bases de datos, por ellos durante el proceso de obtención de requisitos,

dependiendo del tipo de información que manejará la aplicación que se vaya a realizar se tiene que dejar claro en los requerimientos no funcionales cuál será el gestor de bases de datos a utilizar.

Debido a que las aplicaciones bioinformáticas se caracterizan por la complejidad de los algoritmos usados y por el manejo de grandes cantidades de datos inherente a la investigación biológica, las pérdidas causadas por fallos en dichos sistemas son mucho mayores comparadas con otro tipo de aplicaciones. Determinar la **madurez** de éstos e identificar los puntos críticos que impiden obtener un determinado nivel de madurez son acciones claves en la obtención de requisitos de este tipo de software. Cuando se habla de madurez de un software se está enfocando al mejoramiento contínuo de la solución buscada antes de implementarse y a la prevención de defectos que pudieran causar riesgos en la operación que realiza.

Precisamente desarrollar productos lógicos que, cumpliendo las normas, satisfagan las necesidades del usuario, y que tiendan a cero errores es la prioridad número uno en las aplicaciones bioinformáticas. Las causas mecánicas o algorítmicas de los errores se llaman fallos. Se hace evidente que si las aplicaciones bioinformáticas usan algoritmos tan complejos, va a existir una tendencia elevada a que se cometan fallos. Los fallos de funcionamiento de un sistema pueden tener su origen en una especificación inadecuada, errores de diseño del software, averías en el hardware e interferencias transitorias o permanentes en las comunicaciones, pero causa principal radica en inconsistencias en la obtención de los requisitos. Es por ello la necesidad de realizar un proceso de elicitación de requisitos encaminado a evitar la ocurrencia de fallos, es decir a aumentar la fiabilidad del mismo. Hay dos formas de aumentar la fiabilidad de un sistema, una es la prevención de fallos (se trata de evitar que se introduzcan fallos en el sistema antes de que entre en funcionamiento) y la segunda es la **tolerancia de fallos** (se trata de conseguir que el sistema continúe funcionando con un nivel de rendimiento adecuado aunque se produzcan fallos). En ambos casos el objetivo es desarrollar sistemas con modos de fallo bien definidos. (Mejías Álvarez, 2002)

Independientemente de las diferencias que puedan existir en cuanto a lenguaje, formación y visión entre clientes y equipo de desarrollo de aplicaciones bioinformáticas, estos sistemas tendrán que adaptarse a determinados estándares y regulaciones. Es por ello que desde la obtención de los requisitos hay que solucionar cualquier tipo de incongruencia en cuanto a terminología que pueda surgir. Esta es otra de las formas de garantizar la fiabilidad de un sistema.

La fiabilidad de un sistema es una medida de su **conformidad** con una especificación autorizada de su comportamiento. (Mejías Álvarez, 2002)

Precisamente por las características que distinguen a las aplicaciones bioinformáticas del resto de los software es que como técnica de elicitación de requisitos en el desarrollo de estos proyectos para el logro de la fiabilidad se recomienda el JAD (Joint Application Development/Desarrollo conjunto de aplicaciones) por el nivel de precisión tan elevado que se va alcanzando desde un inicio. Según la opinión de la autora esta es una técnica superior a la entrevista, a la tormenta de ideas, a los mapas de concepto, entre otras, porque durante la aplicación de la técnica se trabajará sobre lo que se generará, requiriéndose para esto un grupo de participantes bien integrados y organizados.

2.7 PROPUESTA PARA LOGRAR LA FIABILIDAD EN LOS SOFTWARE BIOINFORMÁTICOS DESDE EL PROCESO DE OBTENCIÓN DE REQUISITOS.

Un atributo de los requisitos que está muy relacionado con la fiabilidad y por supuesto es muy importante en aplicaciones bioinformáticas es el nivel de riesgo: posibilidad de que la información digital se vea afectada, pudiendo causar daños al sistema, o provocar perjuicios económicos. Esto típicamente es el resultado de que el software no se ha realizado como es debido.

El nivel de riesgo tiene cuatro categorías:

- Despreciable: No resulta peligroso porque no causa daños significativos al sistema.
- Marginal: Puede controlarse sin llegar a provocar daños severos al sistema.

- Crítico: Pueda causar daños mayores a la aplicación, pero si se actúa de manera correctiva inmediata puede lograrse la supervivencia del sistema.
- Catastrófico: Puede causar daños fatales de información, o la pérdida del sistema completo.

Es fundamental para los productos bioinformáticos no fallar muy a menudo. Aquí los errores tienen que ser mínimos porque a diferencia de otros software, estos manipulan información humana o para humanos. Por eso durante la fase de obtención de requisitos se debe explorar bien en la posibilidad de fracaso y especificar los niveles reales de riesgo.

La pérdida de información evidentemente es una de las principales cosas que hay que lograr prevenir durante esta primera etapa de desarrollo, pues esto alteraría considerablemente cualquier cálculo u operación que realice el software, por eso el principal objetivo de la obtención de requisitos es evitar a toda costa fallos.

En realidad, como la fase que ocupa a esta investigación es la etapa de obtención de requisitos, a continuación se plantea una guía para llevar a cabo este proceso de forma tal que garantice la fiabilidad de un sistema bioinformático.

Propuesta:

Para llevar a cabo el proceso de obtención de los requisitos en las aplicaciones bioinformáticas debe usarse la técnica del Desarrollo Conjunto de Aplicaciones (JAD), puesto que presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados, durante la aplicación de la técnica se trabajará sobre lo que se generará. Durante la sesión se discute en grupo sobre estos temas, llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema.

1. Analizar y definir el ámbito y alcance del problema a groso modo. Esta tarea consiste en tener una idea general del área específica de la bioinformática que se desea informatizar y de los objetivos que se persiguen con el software.

   ¿Qué tipo de software bioinformático desea realizar?

   I) ____ Procesamiento de Imágenes

   II)____ Búsqueda de información biológica

   III)____ Análisis de información biológica

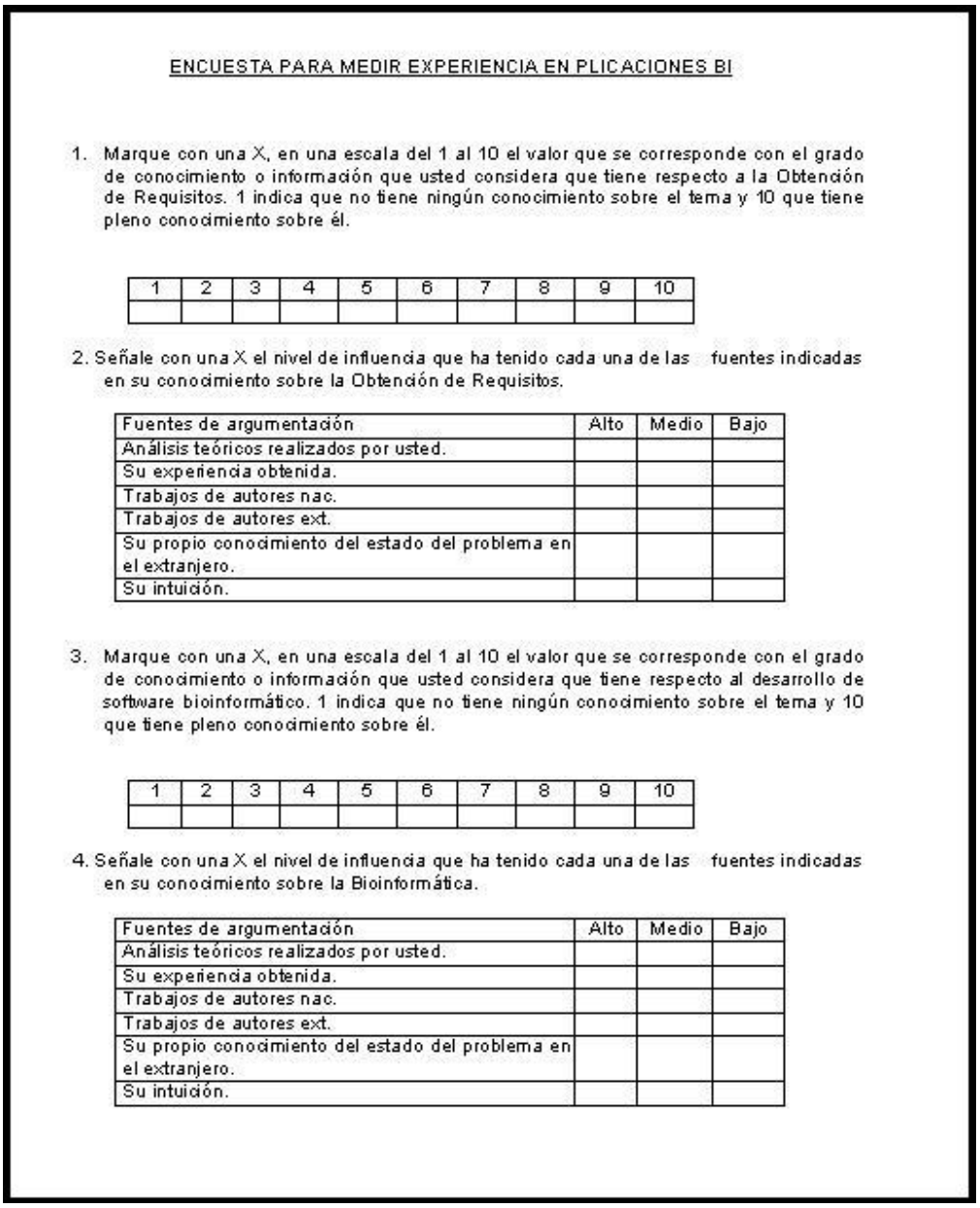
   IV)____ Simulación del comportamiento de sistemas biológicos

   V)____ Cálculos moleculares y químico-teóricos

   VI)____ Tratamientos de algoritmos

   En este primer paso va obteniendo una parte del objetivo principal del proyecto, el atributo que mayormente esta vinculado a esto es Conformidad con los requisitos.

2. En dependencia del tipo de software bioinformático que necesite el usuario se prosigue a seleccionar los analistas que llevarán a cabo el proceso de obtención de requisitos y a elegir correctamente a los clientes que participarán en el intercambio. Se aplica una encuesta a los desarrolladores con el objetivo de medir los conocimientos que poseen en la realización de aplicaciones bioinformáticas.



ENCUESTA PARA MEDIR EXPERIENCIA EN PLICACIONES BI

1. Marque con una X, en una escala del 1 al 10 el valor que se corresponde con el grado de conocimiento o información que usted considera que tiene respecto a la Obtención de Requisitos. 1 indica que no tiene ningún conocimiento sobre el tema y 10 que tiene pleno conocimiento sobre él.

1	2	3	4	5	6	7	8	9	10

2. Señale con una X el nivel de influencia que ha tenido cada una de las fuentes indicadas en su conocimiento sobre la Obtención de Requisitos.

Fuentes de argumentación	Alto	Medio	Bajo
Análisis teóricos realizados por usted.			
Su experiencia obtenida.			
Trabajos de autores nac.			
Trabajos de autores ext.			
Su propio conocimiento del estado del problema en el extranjero.			
Su intuición.			

3. Marque con una X, en una escala del 1 al 10 el valor que se corresponde con el grado de conocimiento o información que usted considera que tiene respecto al desarrollo de software bioinformático. 1 indica que no tiene ningún conocimiento sobre el tema y 10 que tiene pleno conocimiento sobre él.

1	2	3	4	5	6	7	8	9	10

4. Señale con una X el nivel de influencia que ha tenido cada una de las fuentes indicadas en su conocimiento sobre la Bioinformática.

Fuentes de argumentación	Alto	Medio	Bajo
Análisis teóricos realizados por usted.			
Su experiencia obtenida.			
Trabajos de autores nac.			
Trabajos de autores ext.			
Su propio conocimiento del estado del problema en el extranjero.			
Su intuición.			

Figura 2.4: Encuesta para realizar la selección de los desarrolladores que trabajarán en el proyecto bioinformático.

Posteriormente se eligen aquellos analistas que hayan realizado anteriormente otras aplicaciones o investigaciones de este tipo, pues evidentemente contarán con mayor preparación que otros que se dediquen a otra esfera de la Bioinformática. En la medida que la preparación de los especialistas involucrados sea mayor, pues menor será el margen de ocurrencia de fallos y mayor el grado de fiabilidad del producto software. Este paso se ve estrechamente vinculado con la madurez que alcanzará el producto final, pues como los desarrolladores tienen experiencia en esta línea contribuirán al constante mejoramiento del producto y a la prevención de defectos que pudieran causar riesgos en la operación que se realiza.

3. Preparar un proceso organizado y racional de varias iteraciones de intercambio con el cliente. El objetivo principal de esta tarea es formular una serie de preguntas para llevar a cabo las sesiones de obtención de requisitos.

   Aquí primeramente se relacionan las técnicas informáticas a utilizar con el tipo de software bioinformático que se pretende desarrollar.

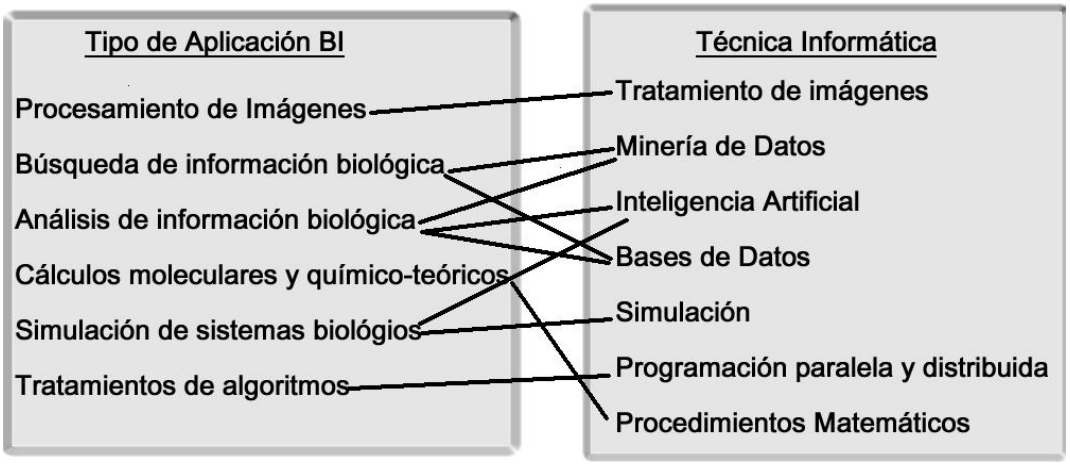


Figura 2.5: Relación entre el tipo de aplicación bioinformática a realizar y la técnica que debe usarse para darle solución.

Como se cuenta únicamente con una visión muy general del sistema, le va a permitir al desarrollador ir valorando cuales podrían ser los puntos críticos que quebrantarían la fiabilidad del producto en un futuro.

4. Se realizan los primeros intercambios formales con el cliente donde se trata de obtener una visión un poco más detallada de las necesidades del sistema y del porqué se hace el software.

Preguntas al clientes	Atributos	Justificación
¿Con qué tipo de ente biológico trabajará la aplicación?	eliminación de errores	Especificar cuál de los sistemas biológicos se va a analizar, permite aplicar las técnicas correspondientes a ese ente y no a otro, evitándose que existan confusiones que provoquen futuros errores
	volatilidad	Al enmarcar el problema se podrá profundizar en las funcionalidades del software, logrando una mayor calidad que garantizará un mayor tiempo de uso de la versión.
	conformidad con los requisitos	Al enmarcarme en el problema se garantizará el cumplimiento de los requisitos.
¿Que tipo de variables de entrada tendrá el software para procesar la información que desea?	tratamiento de errores	Para preparar al sistema o al componente para cuando hagan entradas erróneas.
¿Cuáles son los posibles resultados esperados? Definir rango en cuanto a valores y características.	eliminación de errores	Permite descartar los valores obtenidos que estén fuera del rango delimitado y no cumplan con las características definidas.
	madurez	Delimitando los resultados que se obtendrán, se podrá prevenir la aplicación de riesgos en la operación que realice.
¿Dónde se va a hostear (desplegar) la aplicación?	conformidad con los requisitos	En dependencia de esto se seleccionará la plataforma y tecnología adecuada a utilizar para el

¿Sobre qué sistema correrá la aplicación? Windows, Linux, etc.		desarrollo del software.
¿Conoce algún software que se haya desarrollado con fines similares? Caso(I) Si__ Caso (II) No__		
Si selecciona el Caso (I) ¿Cuál es? ¿Por qué no lo compra?	evolucionabilidad	Teniendo en cuenta aplicaciones similares ya existentes, puedo crear un software con mucha más calidad y funcionalidades.
¿Prefiere algún lenguaje específico para realizar la aplicación? ¿Cuál?	conformidad con estándares	Los desarrolladores verifican si tienen los permisos necesarios para usar todas las herramientas del lenguaje de programación.
¿En dependencia de la investigación que esta llevando a cabo, cuales son los posibles cambios que se podrían dar en el software?	evolucionabilidad	Aquí los analistas deben contemplar cuales serian los posibles cambios en pos de mejorar la funcionalidad de la aplicación.
¿Hacia donde se dirigen las nuevas investigaciones?		

Tabla 2.2: Preguntas que se le realizan al cliente para garantizar fiabilidad desde las primeras sesiones de obtención de requisitos.

Se propone un primer modelo no funcional. En muchas ocasiones el modelo no funcional guía al cliente y hasta le sugiere ideas de cómo hacer mejor lo que desea.

5. Estudiar el lenguaje natural (terminología) usado en aplicaciones bioinformáticas. El objetivo principal de esta tarea es preparar a los analistas en los términos bioinformáticos relacionados con el negocio que se esta llevando a cabo para evitar incomprensiones. Con ello se pretende liberar el software de errores y prevenirlo contra fallos, garantizando así, la fiabilidad desde un inicio.

6. Acercarse al problema de una forma natural e indagar sobre él, sobre lo que se ha hecho en el mundo relacionado con ese tipo de aplicación bioinformática, etc. Es decir documentarse bien, en un período de tiempo limitado. Esto permitirá tener mayor conocimiento del tema y por tanto guiar las sesiones de obtención hacia la búsqueda de unos requisitos para un software más eficiente con menor probabilidad de fallos y en caso de que existan, poder contar con mayor número de opciones para solucionarlo. Lo que garantiza la tolerancia a fallos y la recuperabilidad.

   Que los desarrolladores interactúen directamente con los especialistas (biólogos, químicos, especialistas en medicamentos) en los procesos que desean informatizar. Por ejemplo si se trata del procesamiento de determinado experimento, que los desarrolladores se adentren en los laboratorios donde se desarrollan estos experimentos, vean como es que se realizan y ver pregunten hasta los pasos que los especialistas consideran obvios e insignificantes, para que les facilite un trabajo posterior (variables a calcular, procedimientos algoritmos a utilizar, etc).

   En dependencia de las acciones que realizará el software se verifica si es factible la Reutilización de aplicaciones ya existentes: Otro objetivo importante de los proyectos BI es el estudio de las ventajas que un mecanismo de reutilización aportaría a la solución actual. En concreto, dentro del grupo de desarrolladores se indaga en la búsqueda de una generalización de los requisitos funcionales que permita la reutilización de un código ya implementado (el punto de partida para el diseño un nuevo sistema puede ser un sistema previamente diseñado o una generalización del mismo), así como la reutilización de su información de verificación (esta reutilización permitirá ahorrarse numerosas ejecuciones del algoritmo de verificación, de uso muy frecuente en aplicaciones bioinformáticas). En este ámbito, se calcula la distancia funcional que permita decidir cuándo es ventajoso reutilizar y cuándo la reutilización no aporta ninguna ventaja por estar la funcionalidad de los sistemas disponibles muy alejada del sistema que se pretende diseñar.
   Si ya se ha trabajado en una solución parecida a la que se desea implementar, aquí se podían hacer numerosas preguntas relacionadas con los fallos que se

presentaron en la aplicación anterior, tratando de eliminarlos en esta nueva versión. Al tratar de lograr el perfeccionamiento del software se evidencia la evolucionabilidad de la presente aplicación.

7. Realización de otras sesiones de obtención de requisitos, donde a través de un mecanismo de preguntas y respuestas, el cliente deja claro que es lo que quiere que haga el sistema. La meta principal que se persigue con esta tarea es profundizar en las necesidades reales de los clientes y comprender los objetivos de la solución buscada. Se va refinando el modelo no funcional y con ello va disminuyendo el riesgo a fallos. Para garantizar los atributos de madurez, tolerancia a fallos y recuperabilidad en la aplicación y en dependencia de las operaciones que desarrollará la misma, se realizan las siguientes preguntas a los especialistas (biólogos, químicos, bioquímicos, especialistas en medicamentos, etc.)

Preguntas al clientes
Procesamiento de Imágenes
¿Qué tipo de análisis se le pretende hacer al ente biológico?
¿Se pretende hacer una visualización, edición o un reconocimiento de patrones?
Minería
¿Se utilizará la minería de datos o de textos?
¿Cuáles son los criterios de búsqueda?
Inteligencia Artificial
¿Por qué técnica se va a desarrollar, tiene preferencia por alguna? ¿Cuál?
¿Cuáles son las variables independientes y dependientes?
Bases de Datos
¿En qué formato está la BD que tengo que leer?
¿Qué es lo que se pretende buscar en la BD biológica?
Programación paralela y distribuida
¿Cuál es el algoritmo a utilizar?
¿Cuánto tiempo invierte normalmente en ejecutar ese algoritmo?

¿Posee un closter donde ejecutar el algoritmo? Caso(I) Si__ Caso (II) No__ Si selecciona el Caso (II) ¿Tiene posibilidad de montar uno?

Tabla 2.3: Preguntas que se le realizan al cliente en dependencia del tipo de aplicación bioinformática a desarrollar.

El objetivo fundamental de la fiabilidad es prevenir al software de errores que puedan provocar, pero los desarrolladores deben dejarle claro a los clientes que a pesar de que se trabajará por mitigar o evitar los errores, no significa que la aplicación vaya a estar exenta de estos, por tanto hay que contar con su existencia para pronosticar una posible solución. Para ello se le realizan al cliente las siguientes preguntas.

Preguntas al clientes
Tolerancia a fallos.
¿Si ocurre un fallo, como prefiere que se comporte el sistema? __Sigue funcionando, al menos durante un tiempo, sin perder funcionalidad ni prestaciones. __Sigue funcionando, con una pérdida parcial de funcionalidades, hasta la reparación del fallo. __ El sistema se detiene en un estado que asegura la integridad del entorno, hasta que se repare el fallo.
Recuperabilidad
En caso de ocurrir un error durante la ejecución de la aplicación ¿cómo desea que actúe el sistema? __ Detectar (detecta el error, pero no realiza ninguna acción) __ Detectar y avisar (detecta el error y avisa de ello) __ Tratar (detecta el error e implementa un mecanismo de excepciones)
En caso de ocurrir un fallo ¿Qué información no debe perder bajo ningún concepto?

Tabla 2.4: Cuestionario que se le aplica al cliente para el trabajo en caso de errores.

8. Documentar las observaciones resultantes en varios formatos de representación y comprobar la precisión del conocimiento obtenido durante todas las sesiones de elicitación, creando un documento que describe la conducta externa y las restricciones asociadas a la aplicación bioinformática que se desarrolla. Es decir un documento que recoge los requisitos funcionales, no funcionales y de información del sistema. Lo que permitirá prevenir defectos que pudieran causar riesgos en la operación que realizará el producto y de esta forma evitar fallos en su funcionamiento futuro; con el objetivo de garantizar la fiabilidad del software y por ende la calidad del mismo.

De forma general existen 4 grandes tareas para lograr una calidad óptima en el proceso de obtención de requisitos en aplicaciones bioinformáticas: observación, inmersión/aprendizaje, intercambio y conclusión mediante un diseño gráfico que proponen los desarrolladores a los clientes; todas ellas con el fin de que los analistas se aprehendan de las necesidades reales de los clientes para los cuales realizan la aplicación.

A nadie le gustan claro, los errores, defectos, fallos del sistema, o los datos perdidos, y en la ausencia de cualquier referencia a los tales fenómenos en los requisitos, el usuario asumirá naturalmente que ninguno existirá. Pero incluso en el mundo computarizado de hoy, el usuario más optimista es consciente que las cosas salen mal. Así, los requisitos deben describir el grado a que el sistema debe comportarse en un modo usuario-aceptable.

En algunos casos, los requisitos pueden especificar alguna métrica del "predictor" para la fiabilidad. Un ejemplo típico de esto es el uso de una complejidad métrico, como la complejidad del cyclomatic métrico que puede usarse para evaluar la complejidad y por consiguiente el potencial de un programa software.
También le da la oportunidad de poner las expectativas del cliente y usuarios sobre la cantidad de tiempo que el producto estará disponible para el uso.

Una vez incorporada la fiabilidad dentro del proceso de ingeniería, el siguiente paso consiste en determinar qué cambios o nuevas características del software han de introducirse en este proceso. Para ello se utilizan herramientas de monitorización y bucles de realimentación que proporcionan un nivel adecuado de comprensión del comportamiento y de fiabilidad del software.

2.8 EVALUACIÓN DE LA FIABILIDAD.

No existen en la literatura mecanismos explícitos para garantizar y evaluar este atributo en las aplicaciones bioinformáticas. Entre los elementos básicos que se deben tener en cuenta para esto, es contar con una gestión de requisitos adecuada y dirigida a cumplir con este atributo, evaluar en la aplicación obtenida este factor y contar en los equipos de proyectos con un grupo de herramientas, técnicas, modelos organizados y de fácil acceso.

2.8.1 PROPUESTA PARA MEDIR LA FIABILIDAD DE UN SOFTWARE.

Esto es una propuesta (Castillo M., Lizana Z. & Muñoz C., 2003) que permitirá evaluar un criterio de calidad de software llamado fiabilidad. Mediante la cual se logra analizar y mejorar la fiabilidad del producto de software una vez desarrollado, lo que traerá beneficios tanto para la empresa donde se utilizará como para la empresa desarrolladora.

La propuesta se basa principalmente en el estudio de técnicas estadísticas, estableciendo una metodología que permite comparar la situación ideal con la real, y con esto es posible descubrir que tan alejado está el comportamiento del sistema del funcionamiento ideal, es decir, permite estimar la distancia que hay entre ellas, definiendo así el nivel de fiabilidad que posee el producto desarrollado. La propuesta se aplica una vez finalizado el producto software, para así poder validarla y lograr determinar el nivel de fiabilidad del producto real, lo que permitirá aplicar mejoras en caso de ser necesario.

Propuesta:

El atributo que interesa medir es la Fiabilidad. Como se menciona anteriormente, la fiabilidad nos permite evaluar que tan libre de fallos se encuentra un sistema. Por tanto

el problema consiste en determinar el número de errores que se generan en el sistema en un lapso de tiempo determinado.

Evaluación de la Fiabilidad en el Sistema Ideal	**Evaluación de la Fiabilidad en el Sistema Real**
Tipos de métricas. Las métricas que son posibles utilizar para la fiabilidad son: • **densidad de defectos**. La densidad de defectos es una métrica de calidad que se define como el número de errores que ocurren durante un lapso de tiempo determinado. • **media de ocurrencia de fallos**. Se refiere al promedio del tiempo que tarda en producirse un fallo durante la operación de un producto de software.	**Tipos de métricas.** Al igual que en caso ideal, el atributo a evaluar es la Fiabilidad y la métrica para dicho atributo será **densidad de defectos.**
Tiempo de evaluación. Las pruebas se realizarán simulando nuestra situación con tiempos representativos, repitiendo la simulación, de este modo obtendríamos unas 500 evaluaciones que debieran ser más que representativas.	**Tiempo de evaluación.** El tiempo para la evaluación de cada experimento será de unidades de tiempo constante. El número de experimentos para el caso ideal debiera ser de la misma cantidad que del teórico.
Proceso de medición. • **seleccionar los componentes a medir.** La simulación se diseñó para medir todos los componentes del sistema propuesto. • **medir las características de los**	**Proceso de medición.** • **seleccionar los componentes a medir.** Una vez realizado sistema real se medirán todos los componentes que

componentes con las métricas de software. En este punto como la métrica fue la densidad de defectos, la simulación se prepara para que en un marco de tiempo se contabilicen el número de defectos que aparecerán en la simulación. • **identificar las mediciones anómalas.** Por el hecho de tratarse de una simulación que trabaja con condiciones ideales, las mediciones anómalas podrían ser pocas. • **identificar los componentes anómalos.** Por medio de las mediciones se podrían estimar los posibles componentes que frecuentemente han sido mal codificados.	forman al sistema. • **medir las características de los componentes con las métricas de software.** En este punto la métrica en utilizar será la densidad de defectos. Durante cada experimento se contabilizará el número de defectos ocurridos.

Tabla 2.5: Cómo evaluar la fiabilidad de un sistema.

Resultados Esperados: Para obtener los resultados esperados, tanto para el caso real como para el caso ideal se debe proceder de la misma forma. Primero se debe construir una tabla donde se resuman los datos obtenidos durante la evaluación, esta tabla debe contener los siguientes campos:

Densidad de Defectos	**Frecuencia**
0	
1	
...	
N	

Tabla 2.6: Resultados obtenidos durante la evaluación.

Posteriormente se debe construir un histograma donde se grafique en el eje X la Densidad de Defectos y en el eje Y las respectivas frecuencias. A continuación se debe escoger la ley de probabilidad que mejor represente el comportamiento de la población según el histograma obtenido. Como la población está representada por la densidad de defectos, la ley escogida debiera ser una que represente el tiempo de ocurrencia de fallos.

Debemos recordar que cada ley de probabilidad tiene una función y parámetros asociados. Según la ley escogida se debe estimar los parámetros mediante alguna técnica de métodos numéricos, se proponen estimadores de máxima verosimilitud.

Luego se reemplazan los valores de los parámetros obtenidos en la función de probabilidad escogida, obteniendo una función a la cual llamaremos fi para el caso ideal y fr para el caso real.

Como resultado de la propuesta se espera obtener un gráfico que represente el comportamiento de la población para el caso ideal y otro para el caso real, para luego superponerlos y así poder compararlos. Si las gráficas se aproximan, el software desarrollado cuenta con un alto nivel de fiabilidad. Si se observa que el comportamiento obtenido para el caso real está alejado del comportamiento del caso ideal, se requiere implementar mejoras en el software desarrollado.

2.8.2 LISTA DE CHEQUEO.

Subcaracterística	Pregunta	Evaluación
Madurez (Evaluar la capacidad de la aplicación para evitar fallos)	**1. ¿Cómo fue el nivel de respuesta de la aplicación después que se hizo fallar intencionalmente?**	**1 = Inaceptable** **2 = Debajo del Promedio** **3 = Promedio** **4 = Bueno** **5 = Excelente**
	2. ¿Durante la obtención de requisitos se identificaron todos los posibles fallos que podía tener la aplicación?	**1 = No** **5 = Si**
Tolerancia a Fallos (Verificar la capacidad de la aplicación de seguir procesando en	**1. ¿Cuál fue el nivel de rendimiento de la aplicación después que se hizo fallar intencionalmente?**	**1 = Inaceptable** **2 = Debajo del Promedio** **3 = Promedio** **4 = Bueno**

caso de fallo)		**5 = Excelente**
	2. ¿Cual fue el comportamiento de la aplicación cuando se evalúo con valores fronteras?	**1 = Inaceptable** **2 = Debajo del Promedio** **3 = Promedio** **4 = Bueno** **5 = Excelente**
	3. ¿Cuál fue el comportamiento de la aplicación cuando se ejecutó demandando recursos en cantidad, frecuencia o volúmenes anormales?	**1 = Inaceptable** **2 = Debajo del Promedio** **3 = Promedio** **4 = Bueno** **5 = Excelente**
	4. ¿La aplicación continuó su ejecución después de haberse hecho fallar intencionalmente?	**1 = No** **5 = Si**
	5. Se sometió la aplicación a una gran cantidad de datos para verificar si los límites alcanzados la hacían fallar. ¿Falló?	**1 = No** **5 = Si**
Recuperabilidad (Verificar que el proceso de recuperación restaura apropiadamente la base de datos, la aplicación y el sistema a un estado conocido o deseado)	**1. ¿Cómo se realizó la recuperación después de haber sometido a la aplicación a condiciones extremas?**	**1 = Inaceptable** **2 = Debajo del Promedio** **3 = Promedio** **4 = Bueno** **5 = Excelente**
Conformidad (Verificar que la aplicación se adapta a alguna regulación, convención o estándar referido a fiabilidad)	**1. ¿Se adapta el software a alguna regulación o estándar referido a fiabilidad?**	**1 = No** **5 = Si**

Tabla 2.7: Lista de Chequeo para evaluar la Característica Fiabilidad.

Característica de calidad: Fiabilidad					
Sub-característica	**Peso Asignado**				
	1	**2**	**3**	**4**	**5**
Madurez					
Tolerancia a Fallos					
Recuperabilidad					
Conformidad					

Tabla 2.8: Plantilla de asignación de peso a las subcaracterísticas de fiabilidad.

Después de obtenida la aplicación. Cómo evaluar si la aplicación es fiable o no?

1. Se calcula el promedio, por subcaracterística, de las respuestas a la lista de chequeo. (SC)
2. Se pondera cada subcaracterística, según la importancia que tiene en la aplicación. (P).
3. Se multiplican los valores obtenidos de cada subcaracterística (SC) por su peso correspondiente (P).
4. Se suman los valores obtenidos de la multiplicación y se divide este valor entre la suma de todos los pesos.

Este cálculo se representa a través de la siguiente fórmula:

$$R = \frac{\Sigma SC * P}{\Sigma P}$$

Esta evaluación será representada en un rango del 1 al 5, al igual que los resultados dados para las subcaracterísticas. Después con ese valor se obtiene el porcentaje

Es importante señalar que se tomó, como nivel de satisfacción de la característica de calidad un valor del 85**%.** Un valor por debajo del 85**%** se considera deficiente en el Software que se está evaluando, es decir la aplicación no estaría cumpliendo con el atributo de fiabilidad que se esta evaluando.

2.9 ORGANIZACIÓN DEL CONOCIMIENTO.

En la literatura se considera la Ingeniería de Requisitos como un proceso vital en la producción de software del cual depende en gran medida el éxito de los proyectos; sin embargo, no se encuentran referencias de aplicación y adaptación de este proceso en proyectos de Bioinformática.

Resultados de una encuesta sobre el conocimiento de los que hacen aplicaciones bioinformáticas en la UCI sobre calidad.

La autora aplicó una encuesta a los líderes de los proyectos bioinformáticos de la UCI sobre el conocimiento de estos acerca de la calidad en las aplicaciones de este tipo y se obtuvieron los siguientes resultados.

Figura 2.6: Nivel de conocimiento sobre la calidad de las aplicaciones BI en la UCI.

Los resultados de la encuesta realizada dan una idea del desconocimiento que existe en los equipos de desarrollo de software bioinformático en la UCI sobre cómo lograr la calidad de los sistemas que realizan, de ahí la necesidad que existe de poner a disposición de estos equipos de proyecto materiales de consulta donde el conocimiento

que se ha obtenido como resultado de disímiles investigaciones este organizado y bien localizable.

Precisamente con ese fin se ha diseñado y elaborado en una primera iteración un sitio que se describe en el Capítulo 3, con el objetivo de simplificar y acelerar el trabajo de los equipos de desarrollo de software bioinformático en la UCI para contribuir con la investigación biomédica, proporcionando resultados de investigaciones, plataformas y servicios bioinformáticas. Esto permitirá la interoperabilidad entre fuentes heterogéneas de datos y aplicaciones, ajustándose a las necesidades específicas de cada proyecto de investigación y actualizándose con un sistema automatizado de mantenimiento. Ofrece además un servicio de noticias relacionadas con el tema de carácter internacional.

2.10 CONCLUSIÓN.

Se hace evidente que la calidad de cualquier solución de software bioinformático comienza en una ingeniería de requisitos fiable. Para garantizar que un producto cumpla esta condición, primero se valida proactivamente su fiabilidad en todas las fases principales del proceso de ingeniería de requisitos (obtención, análisis, especificación y validación). Muchas de las herramientas que se utilizan para el desarrollo de estas actividades se emplean indistintamente en dependencia de los fabricantes, pero lo que si es común para todos, es que garantizando la fiabilidad en cada una de ellas, estoy garantizando un porcentaje elevado de la fiabilidad de toda la aplicación.

CAPITULO 3: DESCRIPCIÓN DEL SITIO.

3.1 INTRODUCCIÓN.

El sitio que a continuación se presenta se ha realizado con el objetivo de concentrar un cúmulo de información importante para los proyectos que se dedican a la bioinformática en nuestra universidad, para que los integrantes de estos equipos no pierdan tiempo haciendo búsquedas en Internet y tengan disponible en dicho sitio las principales técnicas, métodos y herramientas útiles en el desarrollo de software bioinformático, así como resultados de investigaciones relacionadas con el tema, todo ello, para garantizar la calidad en estas aplicaciones, aspecto que hoy en la universidad, constituye una meta importante.

En el presente capítulo se realiza una descripción exhaustiva de dicho sitio. Con tal objetivo se ha realizado la arquitectura de información del mismo, ayudado de un árbol de navegación que representa cómo está distribuida la información en el sitio Web. Se ha utilizado para ello elementos basados en texto que permiten mostrar la navegación en la pantalla en cada uno de sus niveles. La redacción de los textos tiene asociados vínculos a direcciones URL que llevan al usuario a la página indicada.

3.2 ELABORACIÓN DEL SITIO.

3.2.1 CONFECCIÓN DEL ÁRBOL DE NAVEGACIÓN.

Una vez que se han definido los contenidos que tendrá el sitio, así como su estructura y diseño, puede procederse a la realización del árbol de navegación. Ello implicará confeccionar un grafo con la forma que tendrá el sitio web y sus principales partes, que se ofrecerá a los usuarios para que conozcan acerca de la distribución de los contenidos en el sitio y avancen hacia ellos por las páginas.

Cuando se usa la idea de crear un árbol, se refiere exactamente a generar un diagrama que cuente con un tronco, ramas y hojas, para mostrar las zonas principales, secundarias y contenidos finales que se irán incorporando. Un árbol de navegación muestra de manera práctica cuántas secciones tendrá el sitio en desarrollo y cuántos niveles habrá dentro de cada una. Básicamente refleja cuál será la experiencia que

tendrá un usuario cuando accede al sitio. De esta manera podremos determinar dónde estará ubicada la información por temas.

Árbol de Navegación:

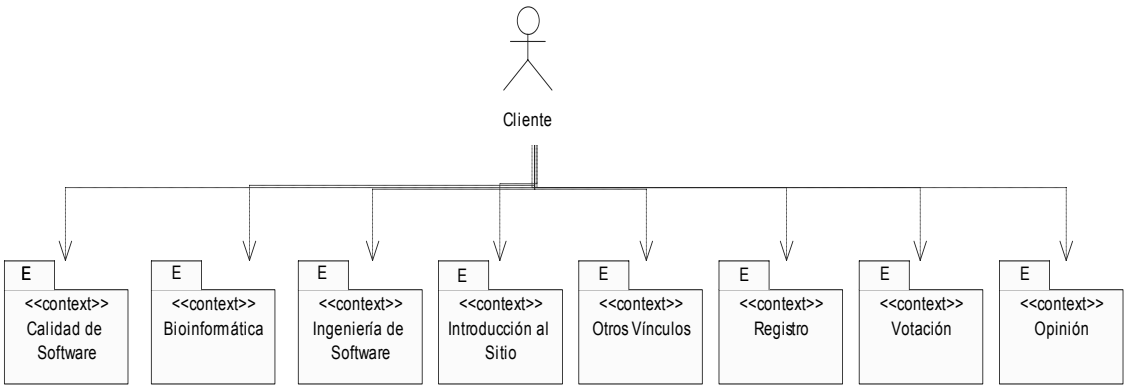


Figura 3.1: Árbol de Navegación. Sección General.

Página de Inicio (Ver Anexo #1)

Guía para lograr la fiabilidad de los software bioinformáticos desde la Obtención de Requisitos. **(Ver Anexo #2)**

Tesis de maestría Ing. Karina Pérez

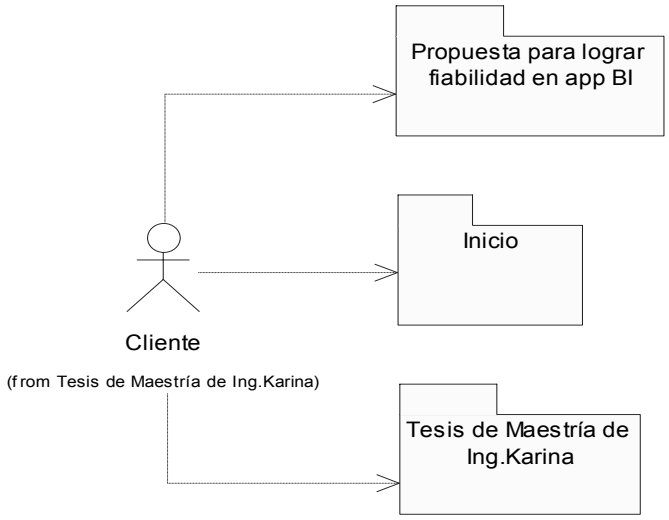


Figura 3.2: Árbol de Navegación para el Paquete Resultado de Investigaciones.

Calidad del Software

Definición

Principios de un Software con calidad

Control de la calidad

Factores de calidad **(Ver Anexo #3)**

Fiabilidad

Atributos de la Fiabilidad **(Ver Anexo #4)**

Cómo medir fiabilidad

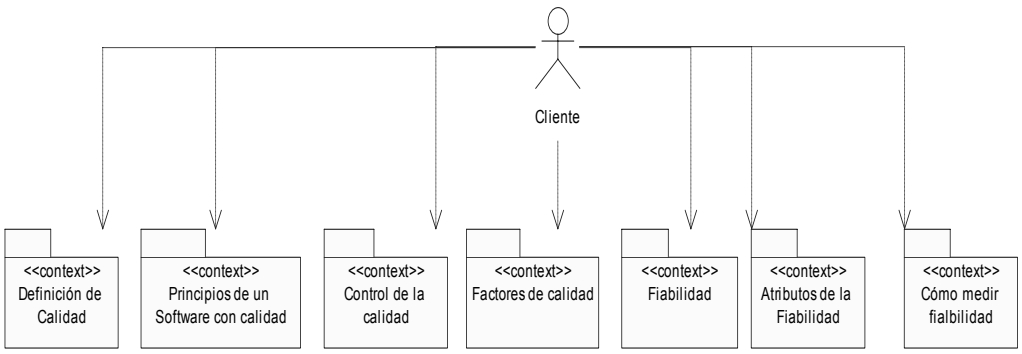


Figura 3.3: Árbol de Navegación para el Paquete Calidad de Software.

Bioinformática

Definición

Surgimiento de la Bioinformática

La Bioinformática en Cuba

Zonas de oportunidad de la Bioinformática en Cuba

Centros dedicados a la Bioinformática en Cuba. Principales Proyectos **(Ver Anexo #5)**

La Bioinformática en la UCI **(Ver Anexo #6)**

Estado Actual

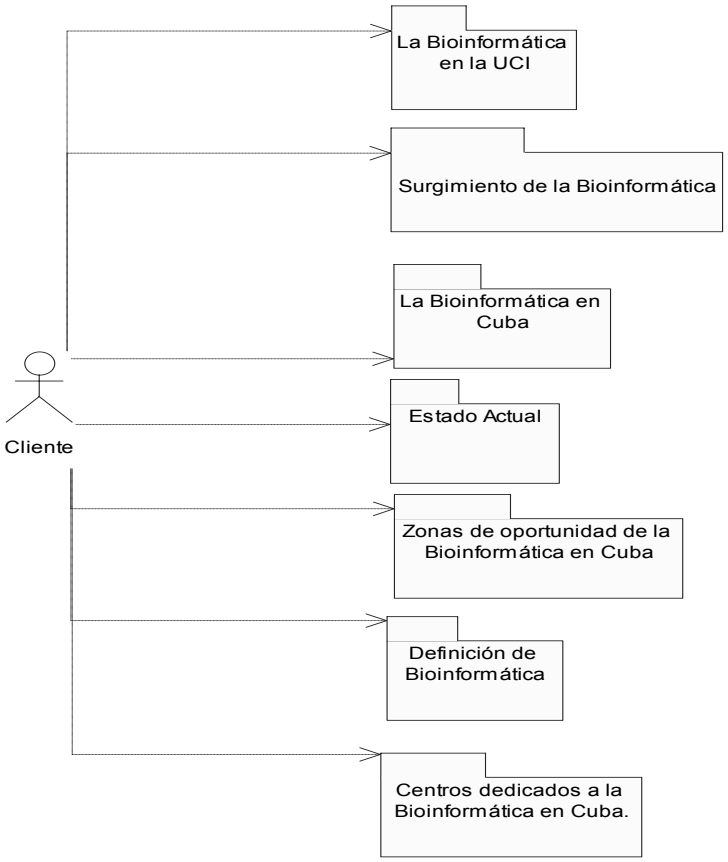


Figura 3.4: Árbol de Navegación para el Paquete Bioinformática.

Ingeniería de Requisitos

Definición

Procesos de la Ingeniería de Requisitos **(Ver Anexo #7)**

Obtención de Requisitos **(Ver Anexo #8)**

Definición de Requisitos

Importancia de la Ingeniería de Requisitos

Los defectos más comunes en la Ingeniería de Requisitos y sus consecuencias **(Ver Anexo #9)**

Beneficios de una buena Ingeniería de Requisitos

Propuesta Metodológica para la obtención de Requisitos (Dr. Amador Durán)

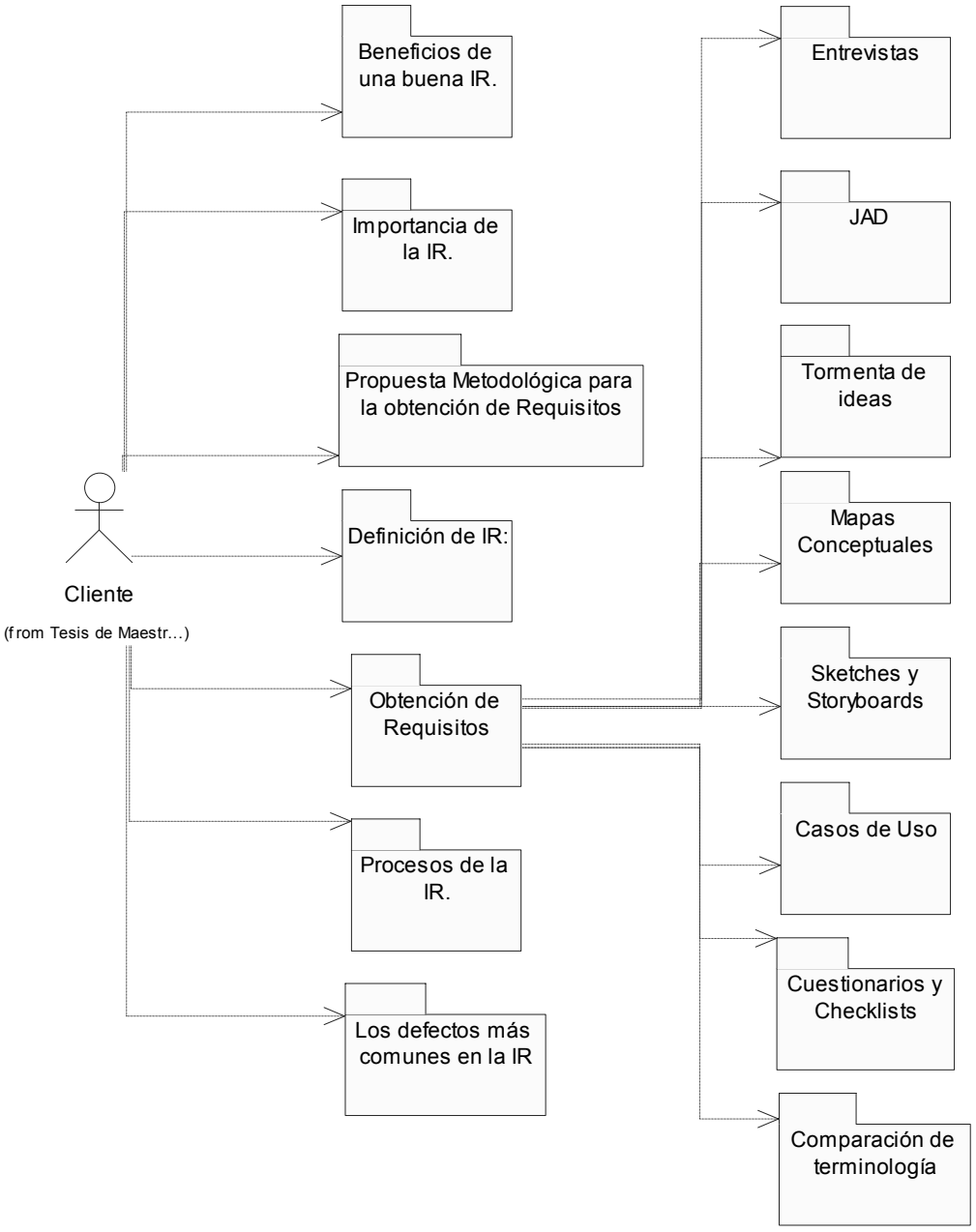


Figura 3.5: Árbol de Navegación para el Paquete Ingeniería de Requisitos.

Otros Vínculos

CIM

CIGB

CNIC

Intranet

Proyectos UCI

Biblioteca UCI

Investigaciones UCI

Grupo de Bioinformática

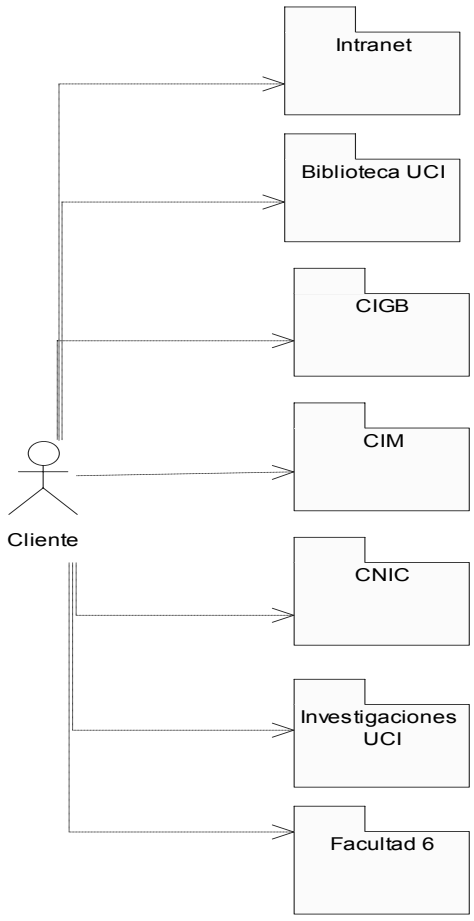


Figura 3.6: Árbol de Navegación para el Paquete Otros Vínculos.

3.2.2 REQUERIMIENTOS FUNCIONALES (SERVICIOS DEL SITIO)

R1- Permitir a los usuarios autenticarse en el sistema con su usuario del dominio.

R2- Permitir a los usuarios navegar por sitios de nuestra red interna, como investigaciones, facultad 6, intranet.

R3- Permitir a los usuarios una vez autenticados enviar sus opiniones sobre lo que creen del sitio.

R4- Permitir a los usuarios contar con un cúmulo de conocimiento organizado acerca de la calidad de los software BI.

R5- Publicar las informaciones relacionadas con el contenido de la investigación.

R6- Publicar la Propuesta para lograr la fiabilidad del software bioinformático durante el proceso de obtención de requisitos.

R7- Permitir a los usuarios votar en dependencia de lo que consideren del sitio.

R8- Mostrar vínculos a sitios web del polo científico para los cuales se producen los software bioinformáticos en la UCI.

http://www.cigb.edu.cu/pages/

http://www.cnic.edu.cu/

http://www.cim.sld.cu/

3.2.3 ARQUITECTURA DE INFORMACIÓN.

La arquitectura de la información se ha trabajado de una forma sencilla, tratando de mostrar toda la información a un clic del usuario, por lo cual se ha podido soportar el sitio en tres niveles de navegación.

1er nivel de navegación (Usuario)

1. Introducción al Sitio.
2. Propuesta para lograr la fiabilidad de los Software Bioinformáticos en la Obtención de Requisitos.
3. Definición de Calidad del Software.
4. Principios de un software con calidad.
5. Control de la calidad.
6. Factores de calidad.

7. Fiabilidad.
8. Atributos de la fiabilidad.
9. Cómo medir fiabilidad
10. Definición de Bioinformática.
11. Surgimiento de la Bioinformática.
12. La Bioinformática en Cuba.
13. Centros dedicados a la Bioinformática en Cuba. Principales Proyectos.
14. La Bioinformática en la UCI.
15. Definición de Ingeniería de Requisitos.
16. Procesos de la Ingeniería de Requisitos.
17. Obtención de Requisitos.
18. Definición de Requisitos.
19. Importancia de la Ingeniería de Requisitos.
20. Los defectos más comunes en la Ingeniería de Requisitos y sus consecuencias.
21. Beneficios de una buena Ingeniería de Requisitos.
22. Propuesta Metodológica para la Obtención de Requisitos (Dr. Amador Durán Toro)
23. Autenticación. (Entrar)
24. Intranet.
25. Investigación UCI.
26. Biblioteca UCI.
27. Proyectos UCI.
28. Centros del Polo. (CIM, CIGB, CNIC)
29. Grupo de Bioinformática.

2do y 3er nivel de navegación (Usuario)

1 Introducción al Sitio.

2 Propuesta para lograr la fiabilidad de los Software Bioinformáticos en la Obtención de Requisitos.

Capítulo 3

3 Definición de Calidad del Software.
4 Principios de un software con calidad.
5 Control de la calidad.
6 Factores de calidad.
7 Fiabilidad.
8 Atributos de la fiabilidad.
9 Cómo medir fiabilidad
10 Definición de Bioinformática.
11 Surgimiento de la Bioinformática.
12 La Bioinformática en Cuba.
13 Centros dedicados a la Bioinformática en Cuba. Principales Proyectos.
14 La Bioinformática en la UCI.
15 Definición de Ingeniería de Requisitos.
16 Procesos de la Ingeniería de Requisitos.
17 Obtención de Requisitos.
- Muestra las técnicas de obtención de requisitos.
    - Muestra en lo que consisten cada una de las técnicas.

18 Definición de Requisitos.
- Muestra las técnicas de definición de requisitos.
    - Muestra en lo que consisten cada una de las técnicas.

19 Importancia de la Ingeniería de Requisitos.
20 Los defectos más comunes en la Ingeniería de Requisitos y sus consecuencias.
21 Beneficios de una buena Ingeniería de Requisitos.
22 Propuesta Metodológica para la Obtención de Requisitos (Dr. Amador Durán Toro)
23 Autenticación. (Entrar)
24 Intranet.
25 Investigación UCI.
26 Biblioteca UCI.
27 Proyectos UCI.
28 Centros del Polo. (CIM, CIGB, CNIC)
29 Grupo de Bioinformática.

1er nivel de navegación (administrador)

1. Introducción al Sitio.
2. Propuesta para lograr la fiabilidad de los Software Bioinformáticos en la Obtención de Requisitos.
3. Definición de Calidad del Software.
4. Principios de un software con calidad.
5. Control de la calidad.
6. Factores de calidad.
7. Fiabilidad.
8. Atributos de la fiabilidad.
9. Cómo medir fiabilidad
10. Definición de Bioinformática.
11. Surgimiento de la Bioinformática.
12. La Bioinformática en Cuba.
13. Centros dedicados a la Bioinformática en Cuba. Principales Proyectos.
14. La Bioinformática en la UCI.
15. Definición de Ingeniería de Requisitos.
16. Procesos de la Ingeniería de Requisitos.
17. Obtención de Requisitos.
18. Definición de Requisitos.
19. Importancia de la Ingeniería de Requisitos.
20. Los defectos más comunes en la Ingeniería de Requisitos y sus consecuencias.
21. Beneficios de una buena Ingeniería de Requisitos.
22. Propuesta Metodológica para la Obtención de Requisitos (Dr. Amador Durán Toro)
23. Autenticación. (Entrar)
24. Intranet.
25. Investigación UCI.
26. Biblioteca UCI.
27. Proyectos UCI.

28. Centros del Polo. (CIM, CIGB, CNIC)
29. Grupo de Bioinformática.
30. Panel de Administración.

2do y 3er nivel de navegación (administrador)

1. Introducción al Sitio.
2. Propuesta para lograr la fiabilidad de los Software Bioinformáticos en la Obtención de Requisitos.
3. Definición de Calidad del Software.
4. Principios de un software con calidad.
5. Control de la calidad.
6. Factores de calidad.
7. Fiabilidad.
8. Atributos de la fiabilidad.
9. Cómo medir fiabilidad
10. Definición de Bioinformática.
11. Surgimiento de la Bioinformática.
12. La Bioinformática en Cuba.
13. Centros dedicados a la Bioinformática en Cuba. Principales Proyectos.
14. La Bioinformática en la UCI.
15. Definición de Ingeniería de Requisitos.
16. Procesos de la Ingeniería de Requisitos.
17. Obtención de Requisitos.

- Muestra las técnicas de obtención de requisitos.
    - Muestra en lo que consisten cada una de las técnicas.

18. Definición de Requisitos.

- Muestra las técnicas de definición de requisitos.
    - Muestra en lo que consisten cada una de las técnicas.

19. Importancia de la Ingeniería de Requisitos.
20. Los defectos más comunes en la Ingeniería de Requisitos y sus consecuencias.
21. Beneficios de una buena Ingeniería de Requisitos.

22. Propuesta Metodológica para la Obtención de Requisitos (Dr. Amador Durán Toro)
23. Autenticación. (Entrar)
24. Intranet.
25. Investigación UCI.
26. Biblioteca UCI.
27. Proyectos UCI.
28. Centros del Polo. (CIM, CIGB, CNIC)
29. Grupo de Bioinformática.
30. Panel de Administración.

3.3 VALIDACIÓN DE LA PROPUESTA.

Para dar una valoración a la propuesta anterior se ha decidido realizar una encuesta a un grupo de personas especializadas en lo que respecta al desarrollo de software bioinformático. Para ello se necesita primeramente seleccionar a los expertos que serán consultados, y luego de confeccionar un listado inicial de personas que al parecer cumplían los requisitos, se someten a una autovaloración para medir experiencia en el tema.

Se tomó una población inicial de 14 expertos, entre ellos, especialistas del Instituto Nacional de Bioinformática, del Centro de Sanidad Agropecuaria, del Centro Nacional de Investigaciones Científicas, líderes de proyectos productivos de la UCI, etc. Todos ellos con experiencia en la realización de software bioinformático.

Después de haber analizado las autoevaluaciones de cada uno de los expertos de la población inicial, se seleccionaron a los 8 expertos de mejor preparación y a esos 8 se les aplicó finalmente la encuesta que evaluaría la propuesta presentada.

Listado de expertos que evaluaron la propuesta:

1- Eduardo Ernesto Corrales Reyna (CIM)
2- Noel Moreno Lemus (UCI)
3- Dany Naranjo Feliciano (CENSA)
4- Alexander Martínez Fundichely (BIOINFO)

5- Lesley Méndez Cáceres (UCI)
6- Aurelio Antela Collado (UCI)
7- Karina Pérez Teruel (UCI)
8- Kalet León Monzón (CIGB)

CRITERIO DE EXPERTO

Nombre y apellidos: _____

Categoría docente: _____

Cargo que ocupa: _____

Años de experiencia en la materia: _____

Nombre de la Tesis: _____

_____

Autor: _____

Pasos para la Metodología	C1 Muy adecuado	C2 Bastante adecuado	C3 Adecuado	C4 Poco adecuado	C5 No adecuado	TOTAL
P-1						
P-2						
P-3						
P-4						
P-5						
P-6						
P-7						
P-8						

Figura 3.7: Modelo para la valoración de la propuesta.

3.3.1 RESULTADOS DE LA VALORACION DE LA PROPUESTA.

La propuesta a evaluar consta de 8 pasos, para valorar cada uno de los pasos se ofrecieron 5 categorías, muy adecuado, bastante adecuado, adecuado, poco adecuado y no adecuado. Después de categorizados cada uno de los pasos por los expertos se obtuvo que para:

Paso 1: bastante adecuado
Paso 2: muy adecuado
Paso 3: muy adecuado
Paso 4: bastante adecuado
Paso 5: muy adecuado
Paso 6: bastante adecuado
Paso 7: bastante adecuado
Paso 8: adecuado

3.4 CONCLUSIÓN.

Dado lo anterior, se puede afirmar que gracias a la realización de este árbol y a la arquitectura de información del sitio en análisis, es posible discutir en términos muy prácticos cuál será la oferta de elementos de información e interacción que tendrá el usuario. Da una sólida idea al usuario de cuáles serán los temas que serán abordados en el sitio y al incluir elementos sencillos de diseño, se permite que la discusión sobre la estructura se desarrolle en aspectos concretos, sin que intervengan aún consideraciones estéticas más rebuscadas como habitualmente suceden.

La propuesta presentada tuvo un nivel de aceptación adecuado en dependencia del criterio que ofrecieron los expertos que la evaluaron.

CONCLUSIONES.

Después de un estudio minucioso, una revisión bibliográfica detallada y una vez concluido el presente trabajo de diploma se puede garantizar que se cumplió con el objetivo general de la investigación.

Dentro de los principales resultados obtenidos con el presenta trabajo está la guía para lograr la fiabilidad de las aplicaciones bioinformáticas vista desde la obtención de los requisitos y el sitio creado con el objetivo de organizar el conocimiento para facilitar la superación de los equipos de proyectos que se dedican en la universidad la producción de software bioinformático.

Se demostró la hipótesis.

Desde la etapa de obtención de requisitos no se puede garantizar el cumplimiento del atributo de fiabilidad (certificaciones), pues se considera que para esto el software debe estar concluido en su totalidad, es decir debe haber pasado por todas las etapas del ciclo de desarrollo de un software.

GLOSARIO DE TERMINOS.

Requisito: Es una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo. Es una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal. [IEEE 1990]

Genoma: Total de genes y cromosomas de un organismo. La definición del genoma de un organismo es precisa (un organismo: un genoma).

Proteoma: Total de proteínas expresadas por el genoma.

Proteómica: Contempla los métodos de exploración del proteoma, la evaluación de efectos sobre un sistema biológico a través de la expresión de proteínas. La expresión de proteínas depende del ciclo celular, de las condiciones de cultivo y de la historia previa (memoria inmunológica).

Sistema Biológico: Complejo de elementos interactuantes, de cuyas interacciones surge un comportamiento como un todo.

Aplicaciones de la Biosimulación: Puede ser aplicada a cada una de las etapas del proceso de investigación biológico-farmacéutica. Puede ser utilizada para la predicción de nuevas drogas y también de nuevos modelos.

Biofármacos: Son medicamentos biológicos de primera generación, hechos a partir de microorganismos vivos, de los cuales se obtienen sustancias idénticas o muy parecidas a las proteínas humanas, las cuales actúan con más eficacia y menos efectos secundarios que los químicos que hasta el momento se han usado para luchar contra el Alzheimer, tumores cerebrales, artritis reumatoidea, infartos cerebrales, insuficiencia cardiaca, lepra, leucemia o lupus. Este tipo de fármacos se manufacturan con ayuda de la ingeniería biotecnológica, la cual usa como principios activos (ingredientes con propiedad terapéutica), órganos y tejidos de vegetales o animales, así como células y fluidos de humanos o animales.

Ácidos nucléicos: Son las moléculas que tienen la información genética de los organismos y son las responsables de su transmisión hereditaria. Existen dos tipos de

ácidos nucléicos, ADN y ARN.

Biomoléculas: Son las moléculas que constituyen a los seres vivos y están formadas por sólo cuatro elementos, que son hidrógeno, oxígeno, carbono y nitrógeno, representando el 97,6 % de los átomos de los seres vivos. Estos cuatro átomos forman las biomoléculas debido a sus tamaños atómicos y distribución electrónica.

Microarreglos: Micromatrices de material biológico. Técnica que permite el análisis simultaneo de un número grande de genes. Proporciona datos cuantitativos y reproducibles, reduce los tiempos, costos y riesgos asociados al descubrimiento y desarrollo de nuevos productos. Con esta tecnología se puede acelerar la investigación básica y el diagnóstico de enfermedades. Permite la caracterización temporal y espacial de la expresión génica, obtener información de genes ortólogos, mapeo de polimorfismos, determinación de especie, detección de cepas, entre otras muchas aplicaciones.

RECOMENDACIONES.

Para la realización de cualquier proyecto bioinformático se debe contar con especialistas (biólogos, químicos, bioquímicas, microbiólogos, especialistas en medicamentos, etc.) para que estén a tiempo completo junto a los desarrolladores, con el objetivo de que cuando estos presenten dudas tengan una vía rápida de erradicarla.

Aplicar las metodologías ágiles (XP), como metodología de desarrollo para este tipo de aplicaciones, pues nos brinda la posibilidad de llegar más rápido al modelo no funcional, y por tanto a lo que quiere el cliente, se desecha fácil lo que se hizo y el cliente descartó, es ideal para el trabajo con módulos pequeños, entre otras ventajas que lo adaptan con facilidad a los software bioinformáticas.

En la UCI el segundo perfil de la facultad 6, que es la que se dedica a la producción de aplicaciones para la bioinformática, se especialice por ramas de la bioinformática (tratamiento de imágenes, inteligencia artificial, programación distribuida y paralela, minería de datos, etc.)

BIBLIOGRAFIA.

BIBLIOGRAFIA CITADA.

1. Proyecto Genoma Humano (PGH). , 2001.

2. A.TOVAL, J. N. Ingeniería del Software. Gestión de Requisitos., ICE-Universidad de Murcia, 1999. p.

3. ÁLVAREZ, P. M. Fiabilidad y Tolerancia de Fallos. , 2002.

4. AMADOR DURÁN, B., RUÍZ, TORO. . 1999.

5. BARBADILLA, A. *Genética y Microbiología.*, Universidad Autónoma de Barcelona., 1996. [Disponible en: http://alarcos.inf-cr.uclm.es/doc/cmsi/trans/S1.pdf]

6. BEDINI, G. *Calidad Tradicional y de Software.* Chile, Dpto de Industrias, Universidad Técnica Federico Santa María., 2000.

7. BETHESDA, M. *International Consortium Completes Human Genome Project.* , 2003.

8. BLAHA, P., 1998.

9. BOOCH., 1999.

10. BRACKETT., 1990.

11. C. CALERO, F. R., M. PIANTTINI. *Ontologies for Software Engineering and Software Technology.* . SPRINGER., . 2006.

12. CARAZO, J. M. *Software para el análisis de datos de genómica y proteómica.* .

13. CHRISTEL, K., 1992.

14. DAVIS., 1993.

15. DÍEZ. 2001.

16. DOMÍNGUEZ, M. J. R. *Ing. Tec. Informática de Gestión. (s.f.).* , 2001.

17. DURÁN, A., 1998.

18. ELIZABETH CASTILLO M., M. L. Z., JUAN ANDRÉS MUÑOZ C. *Evaluación de Atributos de Calidad para un Sistema de Información*., 2003.

19. ESCALONA, T., MEJIAS. . 2002.

20. FEBLES, J. P., 2006.

21. GARCÍA, O. Z. *(s.f.)*, Universidad Ibero Americana, 2003. [Disponible en: http://www.solociencia.com/biologia/bioinformatica-concepto.htm

22. GOGUEN. 1994. p.

23. HSIA, 1993.

24. IBM. IBM, 1997.

25. IEEE. 1990.:

26. INSFRÁN, P., WIERINGA. . 2002.

27. ISO, N. *Norma ISO 8402 UNE*, 1992.

28. ISO, *Norma ISO 9126. (s.f.) Calidad de producto.* .

29. J. M. BARREIRO, F. M., V. MAOJO, F. SANZ. *Biological and Medical Data Analysis Lecture Notes in Computer Science.* . Berlin (Alemania), Springer., 2004.

30. J. P. FEBLES RODRÍGUEZ, A. F. E. *La calidad de Software en Aplicaciones Bioinformáticas.* . Ciudad de la Habana., 2005.

31. JACOBSON., 1993.

32. JACOBSON., 1995.

33. JONHSON, P., ZHU. . 2001.

34. KOCH, N., 2001.

35. KRUCHTEN, 1998.

36. LEÓN, 2006.

37. LILLY, 2000.

38. LIU, Y., 2001.

39. LOWE, H., 1999.

40. M. F. BERTOA, A. V., J. M. TROYA. *Atributos de calidad para componentes*

COTS. . Málaga, España., Dpto de Lenguajes y Ciencias de la Computación, Universidad de Málaga., 2002.

41. M. J. ESCALONA, N. K. *Ingeniería de Requisitos en Aplicaciones para la web. Un estudio comparativo.* Sevilla, España., Dpto. de Lenguajes y Sistemas Informáticos, Universidad de Sevilla., 2002.

42. MCCALL. *Modelo de McCall.* , 1977.

43. MIRANDA, M. T. V. *Fallas con el Software. El Modelo de Madurez de Capacidades (CMM)*, 2002. [Disponible en: http://www.enterate.unam.mx/Articulos/2002/septiembre/fallas.htm

44. MOUNT. 2004.

45. MUS. 1987.

46. NATURE, R. S. Y.: *Revistas Nature y Science.* , 2003.

47. NCBI, 2006.

48. NCBI, N. C. F. B. I., 2001.

49. O. M. FERNÁNDEZ CARRASCO, D. G. L., A. BELTRÁN BENAVIDES. Informes Técnicos. Un enfoque actual sobre la calidad del software. *ACIMED*, 1995.

50. PEÑA. 1998.

51. POHL, 1997.

52. Prentice-Hall, *Software Configuration Management*, 1980.

53. PRESSMAN, R., 1992. p.

54. PRESSMAN, R., 1997. p.

55. PUENTE, J. A. D. L. *Fiabilidad y Tolerancia a fallos.* , 1997. [Disponible en: http://atc.inf-cr.uclm.es/Asignaturas/Grado/DSC/rts-slides-upm-c5.pdf

56. RAGHAVAN, Z., FORD. . 1994.

57. ROMBACH. 1990. p.

58. S. MURIEL HERRERO, G. D. S. *Pliego de Cláusulas Técnicas que regirán la realización de un contrato de "Servicio de Certificación de Aplicaciones y Productos de Software"*, 2007. [Disponible en: http://www.red.es/concursos/concursos_detalle/569_06_DC/PCT_569_06_DC.pdf

59. T. STRACHAN. *Human Molecular Genetics.* , Bios Scientific 2004.

60. TERUEL, K. P. *Modelo de referencia para la Ingeniería de Requisitos en proyectos bioinformáticos.*: Facultad 3. Cuidad de la Habana, Universidad de Ciencias Informáticas, 2007. p.

61. THA., 1997. p.

62. TSG. 1995.

63. UML. 2001.

64. VILAIN, S., SIECKENIUS. . 2002.

65. WEIDENHAUPT, P., JARKE, & HAUMER., 1999.

BIBLIOGRAFIA CONSULTADA.

1. Disponible en: http://www.iibce.edu.uy/2000-08/index.html

2. Disponible en: http://www.ugr.es/~eianez/Biotecnologia/forensetec.htm#3

3. Disponible en: http://www.ideal.es/waste/genomabiochip.htm

4. Disponible en: http://barrapunto.com/articles/100/02/10/2317259.shtml

5. Disponible en: http://www.bioinformacion.net/biochip.htm

6. Disponible en: http://www.inmuno.org/pdf/basesdedatos.pdf

7. Disponible en: http://www.seis.es/i_s/i_s19/i_s19l.htm

8. Disponible en: http://www.lsi.us.es/docencia/pagina_asignatura.php?id=48

9. BERNÁRDEZ, B. *Una Aproximación Empírica al Desarrollo de Heurísticas Basadas en Métricas para Verificación de Requisitos.* , Universidad de Sevilla, 2004. p.

10. DEAN LEFFINGWELL, D. W. *Managing Software Requirements*, Addison Wesley, 2003.

11. DURÁN, A. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información.*, Universidad de Sevilla, 2000. p.

12. G. KONTOYA, I. S. *Requirements Engineering: Processes and Techniques.*, 1997.

13. I. SOMMERVILLE, P. S. Requirements Engineering: A Good Practice Guide. España, Departamento de Lenguajes y Sistemas Informáticos E.T.S. Ingeniería Informática. Universidad de Sevilla1997.

14. LAUESEN., S. Software Requirements: Styles and Techniques., Addison-Wesley, 2002.

15. ROLANDO ALFREDO HERNÁNDEZ LEÓN, J. L. R. D., SAYDA COELLO GONZÁLEZ. El paradigma cuantitativo de la investigación científica., 2003.

16. SUZANNE ROBERTSON, J. R. Mastering the Requirements Process Second Edition, Addison Wesley Professional, 2006.

17. WIEGERS, K. E. More about Software Requirements: Thorny Issues and Practical Advice, 2006.

18. WOHLIN, C. Experimentation in Software Engineering: An Introduction. , Kluwer Academic Publishers, 2000.

ANEXOS.

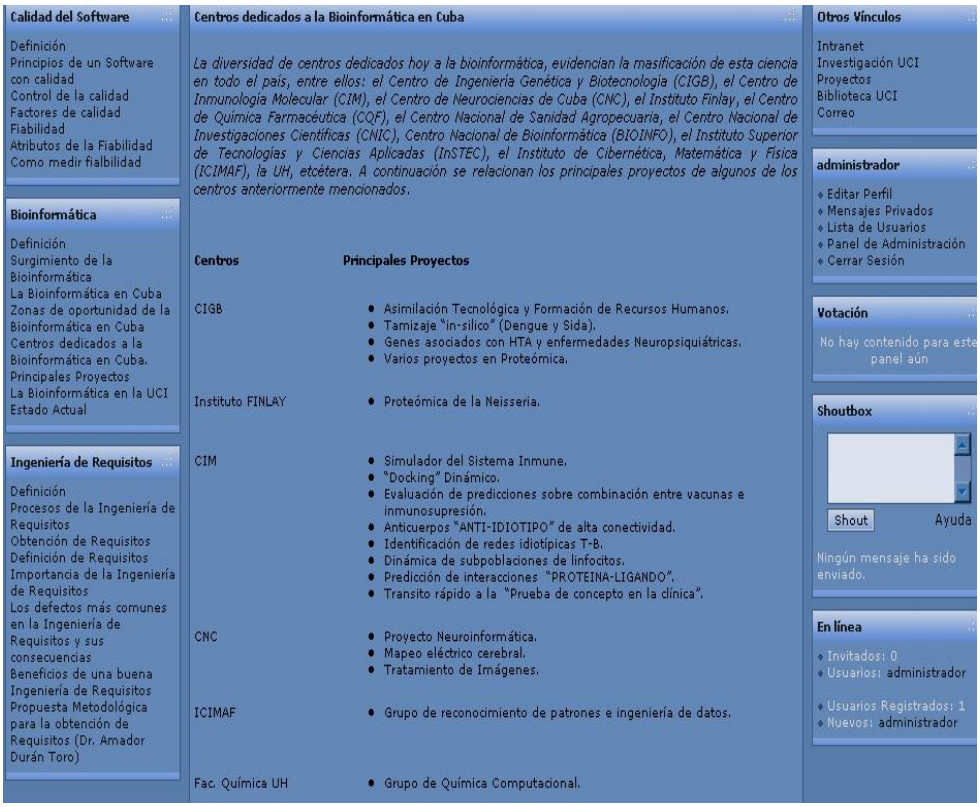
Anexo #1: Pantalla página de Inicio.

Anexo #2: Pantalla Guía Propuesta.

Anexo #3: Pantalla Factores de Calidad de Software.

Anexo #4: Pantalla Atributos de Fiabilidad.

Anexo #5: Pantalla Centros dedicados a la Bioinformática en Cuba.

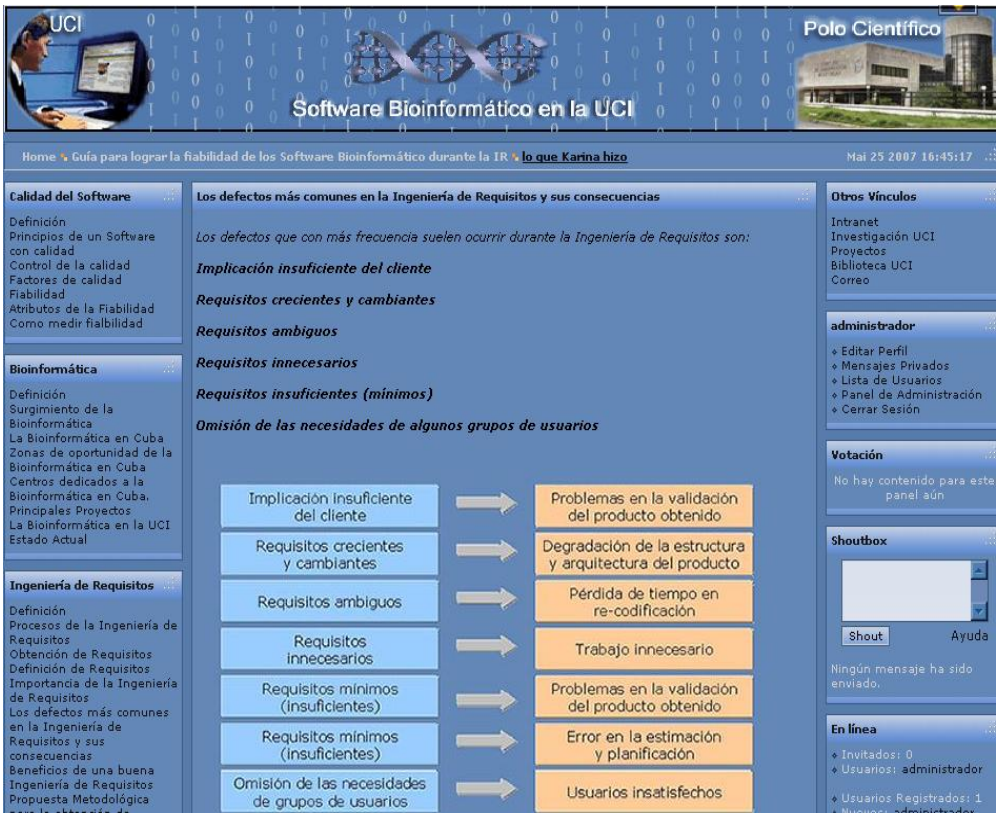


Anexo #6: Pantalla La Bioinformática en la UCI.

Anexo #7: Pantalla Procesos de la Ingeniería de Requisitos.

Anexo #8: Pantalla Obtención de Requisitos.

Anexo #9: Pantalla Los defectos más comunes en la IR y sus consecuencias.



Reference Type: Electronic Source
Record Number: 66
Author: Elizabeth Castillo M., Margarita Lizana Z., Juan Andrés Muñoz C.
Title: Evaluación de Atributos de Calidad para un Sistema de Información
Short Title: Evaluación de Atributos de Calidad para un Sistema de Información