2. INGENIERÍA DE REQUERIMIENTOS.

El proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema de software es llamado Ingeniería de Requerimientos. La meta de la ingeniería de requerimientos es entregar una especificación de requerimientos de software correcta y completa. La ingeniería de requerimientos apunta a mejorar la forma en que comprendemos y definimos sistemas de software complejos. Existen varias definiciones de requerimientos, de entre las cuales podemos citar las siguientes:

Los Requerimientos fueron definidos por la IEEE como [IEEE90]:

1. Condición o capacidad requerida por el usuario para resolver un problema o alcanzar un objetivo

2. Condición o capacidad que debe satisfacer o poseer un sistema o una componente de un sistema para satisfacer un contrato, un standard, una especificación u otro documento formalmente impuesto

3. Rrepresentación documentada de una condición o capacidad como en 1 o 2.

Según Zave:

- Rama de la ingeniería del software que trata con el establecimiento de los objetivos, funciones y restricciones de los sistemas software.

- Asimismo, se ocupa de la relación entre estos factores con el objeto de establecer especificaciones precisas.

Según Boehm:

- Ingeniería de Requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema.

Según Loucopoulos:

- Trabajo sistemático de desarrollo de requisitos, a través de un proceso iterativo y cooperativo de análisis del problema, documentando los resultados en una variedad de formatos y probando la exactitud del conocimiento adquirido.

Según Leite:

- Es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. Este proceso utiliza una combinación de métodos, herramientas y actores, cuyo producto es un modelo del cual se genera un documento de requerimientos.

Esta tesis adhiere a esta ultima definición [Leite89]. Se desarrollan sistemas de software para satisfacer una necesidad percibida por un cliente. Pueden formularse las necesidades reales del cliente como requerimientos. Los requerimientos son desarrollados conjuntamente por el cliente, usuario y diseñadores del sistema de software. La ingeniería de requerimientos es un proceso de descubrimiento, refinamiento, modelización, especificación y validación de lo que se desea construir. En este proceso tanto el cliente como el analista juegan un papel muy importante.

Cuando nos encontramos al frente de un proyecto de desarrollo de sistemas es importante dejar claramente definidos los requerimientos del software, en forma consistente y compacta, esta tarea es difícil básicamente porque consiste en la traducción de unas ideas vagas de necesidades de software en un conjunto concreto de funciones y restricciones. Además el analista debe extraer información dialogando con muchas personas y cada una de ellas se expresará de una forma distinta, tendrá conocimientos informáticos y técnicos distintos, y tendrá unas necesidades y una idea del proyecto muy particulares.

Es justamente este último punto uno de los que se atacará en esta tesis: que el usuario y el desarrollador compartan el mismo lenguaje asegura la comunicación entre ambos. [Leite89] sugiere que en particular el uso del lenguaje propio del usuario mejora considerablemente esta comunicación. En el proceso de Ingeniería de Requerimientos la validación de los diferentes productos requiere una fuerte interacción con el usuario [Loucopoulos], la que se ve facilitada por el vocabulario común usuario-desarrollador. Las diferentes representaciones que se construyen en el proceso de desarrollo de software encuentran en el vocabulario del usuario un marco referencial que permite al desarrollador, obtener un vocabulario de trabajo que es un subconjunto de la terminología del cliente, lo que a su vez facilita el acceso a la documentación por parte de todos los participantes en el desarrollo. En muchos casos es difícil especificar los requerimientos de un problema, con la mayor calidad posible en una etapa temprana del proceso de desarrollo.

La construcción de prototipos constituye un método alternativo, que da como resultado un modelo ejecutable del software a partir del cual se pueden refinar los requerimientos. Para poder construir el prototipo se necesitan técnicas y herramientas especiales.

El análisis de requerimientos establece el proceso de definición de requerimientos como una serie de tareas o actividades mediante las cuales se

busca ganar conocimientos relevantes del problema, que se utilizarán para producir una especificación formal del software necesario para resolverlo. En este proceso se deben conciliar diferentes puntos de vista y utilizar una combinación de métodos, personas y herramientas. El resultado final constituye la documentación de los requerimientos. Éstos deben expresarse de forma clara y estructurada de manera que puedan ser entendidos tanto por expertos como por el usuario, quien deberá participar en la validación, [Rumbaugh].

Como resultado del análisis se desarrolla la especificación de requerimientos del software. La revisión es esencial para asegurar que el realizador del software y el cliente tengan la misma percepción del sistema.

Una vez finalizado nuestro proyecto de desarrollo de sistemas, y contando con un buen análisis de requerimientos, podremos evaluar la calidad del producto terminado.

2.1. Proceso de Análisis de Requerimientos.

El proceso del establecimiento de requerimientos de un sistema de software, como ya mencionamos, es el primer paso esencial en entregar lo que el cliente desea. A pesar de esto, la insuficiencia de tiempo y esfuerzo son a menudo encontrado en esta actividad y existen pocos métodos sistemáticos para soportarlo. Entre los métodos conocidos se puede citar a los siguientes:

Para Pressman, en el proceso de análisis de requerimientos del software se puede identificar cinco tareas o etapas fundamentales:

1. *Reconocimiento del problema*
   Se deben de estudiar inicialmente las especificaciones del sistema y el plan del proyecto del software. Realmente se necesita llegar a comprender el software dentro del contexto del sistema. El analista debe establecer un canal adecuado de comunicación con el equipo de trabajo involucrado en el proyecto. En esta etapa la función primordial del analista en todo momento es reconocer los elementos del problema tal y como los percibe el usuario.

2. *Evaluación y síntesis*
   En esta etapa el analista debe centrarse en el flujo y estructura de la información, definir las funciones del software, determinar los factores que afectan el desarrollo de nuestro sistema, establecer las características de la interfaz del sistema y descubrir las restricciones del diseño. Todas las tareas anteriores conducen fácilmente a la determinación del problema de forma sintetizada.

3. *Modelización*
   Durante la evaluación y síntesis de la solución, se crean modelos del sistema que servirán al analista para comprender mejor el proceso funcional, operativo y de contenido de la información. El modelo

servirá de pilar para el diseño del software y como base para la creación de una especificación del software.

4. *Especificación*
   Las tareas asociadas con la especificación intenta proporcionar una representación del software. Esto más adelante permitirá llegar a determinar si se ha llegado a comprender el software, en los casos que se lleguen a modelar se pueden dejar plasmados manuales.

5. *Revisión*
   Una vez que se han descrito la información básica, se especifican los criterios de validación que han de servir para demostrar que se ha llegado a un buen entendimiento de la forma de implementar con éxito el software. La documentación del análisis de requerimientos y manuales, permitirán una revisión por parte del cliente, la cual posiblemente traerá consigo modificaciones en las funciones del sistema por lo que deberán revisarse el plan de desarrollo y las estimaciones previstas inicialmente.

Método CORE

El método Controlled Requirements Expression (CORE) [Norris] es un conjunto de notaciones textuales y gráficas, con guías especificadas para la captura y validación de requerimientos del sistema, en las etapas iniciales del diseño del sistema. CORE ha sido, por tradición, pensado como puramente una técnica de captura y análisis de requerimientos (RCA), aunque soporta algunos aspectos de diseño tales como estructuras de datos. CORE está basada en el principio de primero definir el problema a ser analizado (definición del problema), y luego dividirlo en unidades o puntos de vista a considerar.

El método CORE consiste en siete etapas. Cada una produce salidas que alimentan a la etapa subsecuente como entrada o que forman parte de la especificación de requerimientos final. CORE pretende examinar el sistema y su ambiente en un número de niveles, con detalles más finos progresivamente en cada nivel. Las siete etapas se presentan a continuación:

1. *Definición del problema*.
   El propósito de la definición del problema es identificar los límites del mismo. Contiene detalles de los objetivos de la empresa de los usuarios del sistema, la base para la necesidad de un nuevo sistema, limitaciones de costo y tiempo, y quién va a ser el responsable de la revisión y aceptación de los resultados finales.

2. *Estructuración del punto de vista*.
   El propósito de esta etapa es descomponer el ambiente del sistema en los elementos para que el sistema propuesto pueda ser analizado desde los puntos de vista de todas las entidades que se comunican con él, la más importante de las cuales son los usuarios. Durante esta etapa, todas las entidades que son fuentes potenciales de información deben ser identificadas.

3. *Colección tabular*.
   Esta etapa es cuando la información sobre los flujos de datos entre los puntos de vista y el procesamiento de éstos son reunidos. Esto ayuda a establecer la totalidad y consistencia.

4. *Estructuración de datos.*
   En la etapa previa, los elementos de información que pasan entre los puntos de vista son referidos por sus nombres generales. En esta etapa, se da una vista más cercana al contenido, a la estructura y a la derivación de datos, al producir diagramas de estructura de datos.

5. *Modelación individual de puntos de vista*.
   Esta etapa puede dividirse en dos partes. Lo único concerniente a la primera es convertir las TCF'S en una notación diferente para producir los diagramas individuales del modelo de punto de vista. La segunda parte se refiere a agregar alguna información nueva perteneciente a flujos de datos internos, control de acciones y tiempo de acciones.

6. *Modelación combinada de punto de vista*.
   Esta etapa facilita el análisis de una secuencia de eventos de más de un punto de vista. Cada diagrama de modelo combinado de punto de vista producido durante esta etapa es una representación del procesamiento de información que ocurre entre puntos de vista.

7. *Análisis de restricciones*.
   En esta etapa, se consideran restricciones adicionales tales como desempeño y seguridad. Éstas pueden afectar los diagramas de puntos de vista ya producidos. Las restricciones se documentan en una especificación de restricción del sistema.

Los métodos analizados, como gran parte de los existentes, conciernen a la producción del documento de especificación de requerimientos y no direcciona el proceso de captura de requerimiento en detalle, a menudo sólo proporcionan guías simples de técnicas de entrevista y algunas veces áreas claves que deben investigarse pero que a menudo se olvidan. Cada método individualmente, dará diferentes soluciones para diferentes proyectos, incluyendo aquellos casos en los que el dominio y el área del problema son el mismo. Por esta razón, considero que no existe un modelo de proceso ideal para la Ingeniería de Requerimientos; encontrar el método o la técnica perfecta es una ilusión, pues cada método y técnica ofrece diferentes soluciones ante un problema, no obstante la aplicación de alguno de ellos nos ayuda a la producción de especificación de requerimientos.

El proceso de ingeniería de requerimientos en el que se basara este trabajo de tesis es el que propone Loucopoulos [Loucopoulos], en el que se

plantea que en esta fase hay que considerar por lo menos tres aspectos fundamentales:

- **Comprender** el problema
- **Describir** formalmente el problema
- **Obtener** un acuerdo sobre la naturaleza del problema

Esto nos llevaría a simplificar el proceso a tres etapas para obtener los requerimientos del problema que estamos atacando, estas etapas son las siguientes [Loucopoulos]:

- **Elicitación de requerimientos**
- **Especificación**
- **Validación**

Elicitación de requerimientos

El propósito de la elicitación de requerimientos es ganar conocimientos relevantes del problema, que se utilizarán para producir una especificación formal del software necesario para resolverlo. "Un problema puede ser definido como la diferencia entre las cosas como se perciben y las cosas como se desean"[1]. Aquí vemos nuevamente la importancia que tiene una buena comunicación entre desarrolladores y clientes; de esta comunicación con el cliente depende que entendamos sus necesidades.

Al final de la fase de análisis de requerimientos el analista podría llegar a tener un conocimiento extenso en el dominio del problema.

Especificación

Una especificación puede ser vista como un contrato entre usuarios y desarrolladores de software, que define el comportamiento funcional deseado del artefacto de software (y otras propiedades de éste, tales como perfomance, confiabilidad, etc.), sin mostrar como será alcanzada tal funcionalidad.

Validación

La validación es el proceso que certifica que el modelo de los requerimientos es consistente con las intenciones de los clientes y los usuarios; ésta es una visión más general que el concepto común de validación, pues se produce en paralelo con la elicitación y la especificación, tratando de asegurar que tanto las ideas como los conceptos presentados en una descripción se identifican y explican con claridad

La necesidad de validación aparece cuando:

- Se incorpora una nueva pieza de información al modelo actual.

[1] Gause & Weinberg. Turn your lights on, 1989

- Cuando diferentes piezas de información se incorporan en un todo coherente.

La validación no sólo se aplica al modelo final de los requerimientos, sino también a los modelos intermedios.

En la figura 1 podemos ver el esquema del proceso planteado

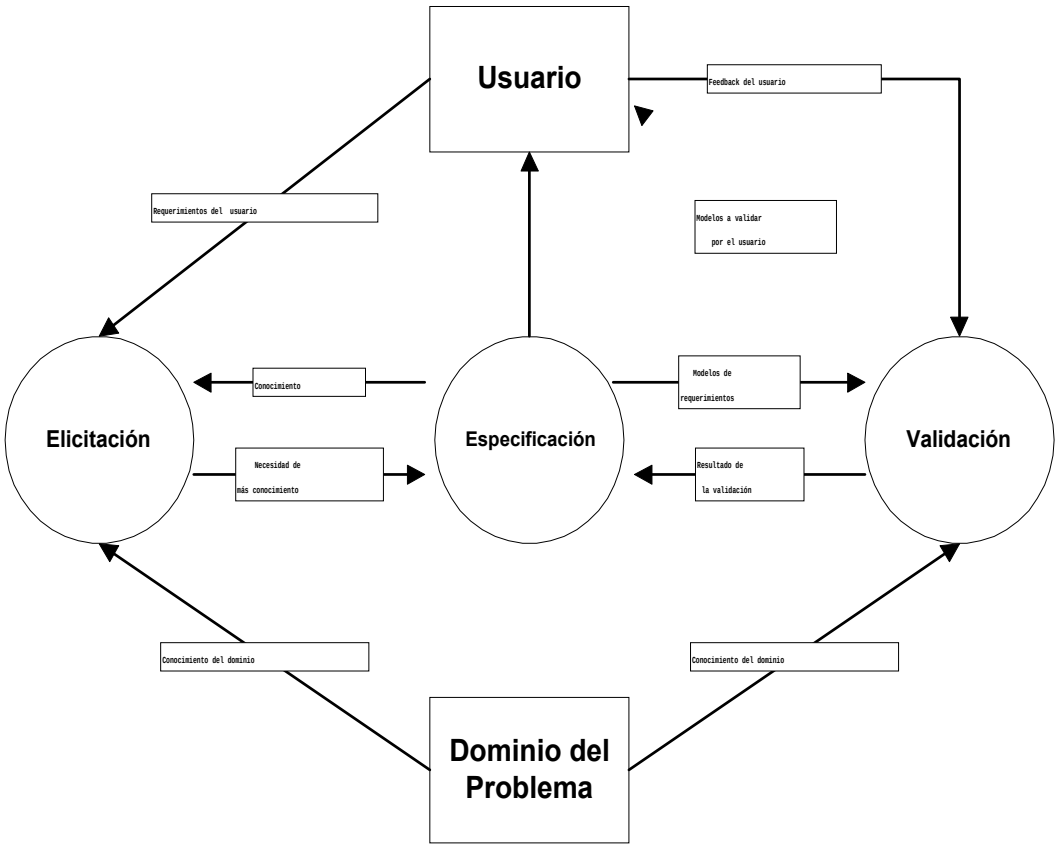

Figura1: Esquema del proceso de ingeniería de requerimientos (Fuente: Loucopoulos)

2.2. Áreas de problemas.

Uno de los problemas principales que podemos encontrar a la hora de especificar los requerimientos, es la necesidad de un entendimiento de los problemas del cliente, deseos y ambiente en el cual el sistema será instalado. . Para [Booch], uno de los factores principales de problemas es la complejidad del dominio, en el cual se encuentran los requerimientos. También tenemos que tener en cuenta, que el análisis de requerimientos es una actividad de intensa comunicación. Siempre que existe comunicación, se pueden producir malas interpretaciones, omisiones, ect. que pueden producir problemas tanto al analista como al cliente [Pressman]. Estos problemas se producen por la difícil iteración entre los clientes de un sistema y sus desarrolladores. Vemos que por un lado

hay un cliente que necesita resolver un problema, y por otro, hay un técnico que tratará de darle la solución desarrollando un sistema, tanto los clientes como los desarrolladores a menudo utilizan terminología diferente provocando problemas de comunicación. Los usuarios encuentran generalmente muy difícil dar precisión sobre sus necesidades de forma que los desarrolladores puedan comprender.

Normalmente, los clientes y desarrolladores tienen diferentes perspectivas de la naturaleza del problema y hacen suposiciones diferentes sobre la naturaleza de la solución. Por lo tanto, el alcance de la especificación de requerimientos hace de su producción una tarea especializada que es más difícil de hacer por ciertos factores, entre los cuales se incluye:

1. Poca calidad en la comunicación

2. Uso de herramientas y/o técnicas no adecuadas

3. La tendencia a reducir el tiempo del análisis de requerimientos

4. La no-consideración de alternativas antes de especificar el software

5. Cambios de los requerimientos debido a las variaciones en el ambiente y en la comprensión tanto del cliente como del desarrollador conforme progresa el desarrollo.

Un enfoque organizado para el desarrollo de software es esencial para reducir estos factores de dificultad. Hay muchas oportunidades para cometer errores por requerimientos mal interpretados en el camino hacia una implantación incorrecta.

Con el fin de servir como guía y agilizar el proceso de desarrollo de software, se desarrollaron varios modelos de ciclo de vida. No hay un modelo correcto para todos los propósitos, aun así, el uso de un modelo apropiado puede ayudar de manera considerable en el control de un proyecto de software. Entre los más populares y por todos conocidos podemos nombrar: El modelo de cascada, modelo en espiral de Boehm y Prototipado Evolutivo.

El modelo de cascada o ciclo de vida clásico

La versión original del modelo en cascada, fue presentada por Royce en 1970, aunque son más conocidos los refinamientos realizados por Boehm (1981), Sommerville (1985) y Sigwart et al. (1990).

En este modelo, el producto evoluciona a través de una secuencia de fases ordenadas en forma lineal, permitiendo iteraciones al estado anterior.

El número de etapas suele variar, pero en general suelen ser:

- Análisis de requerimientos del sistema.
- Análisis de requerimientos del software.
- Diseño preliminar.

- Diseño detallado.
- Codificación y pruebas.
- Explotación (u operación) y mantenimiento.

Modelo en espiral de Boehm

En 1988 Boehm propone el modelo en espiral, para superar las limitaciones del modelo en cascada. La espiral se forma a partir de una serie de ciclos de desarrollo y va evolucionando. Los ciclos internos del espiral denotan análisis y prototipado y los extemos el modelo clásico. En la dimensión radial están los costos acumulativos y la dimensión angular representa el progreso realizado en cada etapa.

En cada ciclo se empieza identificando los objetivos, las alternativas y las restricciones del mismo. Se deben evaluar las alternativas de solución respecto de los objetivos, considerando las restricciones en cada caso. Es en este momento en que se puede llevar a cabo el siguiente ciclo.

Prototipado Evolutivo

El uso de prototipos se centra en la idea de ayudar a comprender los requerimientos que plantea el usuario, sobre todo si este no tiene una idea muy acabada de lo que desea. También pueden utilizarse cuando el ingeniero de software tiene dudas acerca de la viabilidad de la solución pensada.

Esta versión temprana de lo que será el producto, con una funcionalidad reducida, en principio, podrá incrementarse paulatinamente a través de refinamientos sucesivos de las especificaciones del sistema, evolucionando hasta llegar al sistema final.

Al usar prototipos, las etapas del ciclo de vida clásico quedan modificadas de la siguiente manera:

- Análisis de requerimientos del sistema.
- Análisis de requerimientos del software.
- Diseño, desarrollo e implementación del prototipo
- Prueba del prototipo.
- Refinamiento iterativo del prototipo.
- Refinamiento de las especificaciones del prototipo.
- Diseño e implementación del sistema final.
- Explotación (u operación) y mantenimiento.

Todos los ciclos de vida para el desarrollo de software descriptos antes incluyen una fase para el análisis de los requerimientos del sistema.

Uno de los mayores problemas que surgen en la fase de análisis y definición de los requerimientos en una etapa temprana del desarrollo, es cómo organizar toda la información que adquiere el analista en sus entrevistas con las personas involucradas en un proyecto y cómo poner de acuerdo a todas estas

personas sobre cuál es la solución más adecuada. Mientras que los métodos clásicos se basan en entrevistas bilaterales (el analista y cada una de las partes) con lo que dejan para el analista toda la labor de organización y obtención de un consenso, recientemente se tiende a prácticas más relacionadas con el *brainstorming* en el que cada parte expone sus ideas y propuestas y se produce un debate de forma que las posiciones vayan acercándose sucesivamente hasta que se llegue a un consenso [Raghavan].

2.3. Técnicas de elicitación.

El análisis de requerimientos siempre comienza con una comunicación entre dos o más partes. En el libro *Ingeniería de Software* de R. Pressman [Pressman], nos sugiere que un cliente tiene un problema al que puede encontrar una solución basada en computadora. El desarrollador responde a la petición del cliente. La comunicación ha comenzado. Pero, el camino entre la comunicación y el entendimiento está lleno de baches.

Antes de mantener las reuniones con los clientes y usuarios e identificar los requerimientos es fundamental conocer el dominio del problema. Enfrentarse a un desarrollo sin conocer las características principales ni el vocabulario propio de su dominio suele provocar que el producto final no sea el esperado por clientes ni usuarios. Por otro lado, mantener reuniones con clientes y usuarios sin conocer las características de su actividad hará que probablemente no se entiendan sus necesidades y que su confianza inicial hacia el desarrollo se vea deteriorada enormemente.

Para conocer el dominio del problema se puede obtener información de fuentes externas al negocio del cliente: folletos, informes sobre el sector, publicaciones, consultas con expertos, etc. En el caso de que se trate de un dominio muy específico puede ser necesario recurrir a fuentes internas al propio negocio del cliente, en cuyo caso pueden utilizarse las técnicas de elicitación de requerimientos como el estudio de documentación, observación *in situ*, cuestionarios, etc.

En realidad una primera reunión entre el cliente y el analista servirá como un período corto de preguntas y respuestas, el cual, en adelante debe sustituirse por reuniones que busquen entender el problema del usuario.

Normalmente encontramos que los clientes y analistas se enfrascan en el proyecto de forma unilateral y no en equipo. Cada parte define su propio "territorio" y se comunica a través de una serie de notas, impresos formales, documentos y sesiones de preguntas y respuestas. Este enfoque no es muy efectivo, abundan los malentendidos, se pierde información importante y nunca se establece una relación de trabajo satisfactoria.

Con estos problemas presentes, se desarrollaron numerosas técnicas para tratar de superar este difícil momento, que es el inicio del proceso. Cada técnica puede aplicarse en una o más actividades de la ingeniería de requerimientos; en la práctica, la técnica más apropiada para cada actividad dependerá del proyecto que esté desarrollándose.

A continuación se resumen algunas de las más conocidas:

Entrevistas

Las entrevistas son la técnica de elicitación más utilizada, y de hecho son prácticamente inevitables en cualquier desarrollo. En las entrevistas se pueden identificar claramente tres fases [Piattini]: preparación, realización y análisis, que se describen a continuación.

Preparación de entrevistas

Las entrevistas no deben improvisarse, por lo que conviene realizar las siguiente tareas previas:

- **Estudiar el dominio del problema**: se debe conocer la terminología básica del dominio del problema, evitando que el cliente tenga que explicar términos que para él son obvios. Para ello se puede recurrir a la técnica auxiliar de estudio de documentación, a bibliografía sobre el tema, documentación de proyectos similares realizados anteriormente, etc.

- **Seleccionar a las personas a las que se va a entrevistar**: se debe minimizar el número de entrevistas a realizar, por lo que es fundamental seleccionar a las personas a entrevistar. El orden de realización de las entrevistas también es importante. Normalmente se aplica un enfoque *top–down*, comenzando por los directivos, que pueden ofrecer una visión global, ayudar a determinar los objetivos y reducir ciertas *reticencias* en sus subordinados, y terminando por los futuros usuarios, que pueden aportar información más detallada.

- **Determinar el objetivo y contenido de las entrevistas**: para minimizar el tiempo de la entrevista es fundamental fijar el objetivo que se pretende alcanzar y determinar previamente su contenido.

- **Planificar las entrevistas**: la fecha, hora, lugar y duración de las entrevista deben fijarse teniendo en cuenta siempre la agenda del entrevistado.

Realización de entrevistas

Dentro de la realización de las entrevistas se distinguen tres etapas, tal como se expone en [Piattini]:

- **Apertura**: el entrevistador debe presentarse e informar al entrevistado sobre la razón de la entrevista, qué se espera conseguir, cómo se utilizará la información, la mecánica de las preguntas, etc.

- **Desarrollo**: la entrevista en sí no debería durar más de dos horas, distribuyendo el tiempo en un 20% para el entrevistador y un 80% para el entrevistado. Se deben evitar los monólogos y mantener el control por parte del entrevistador, contemplando la posibilidad de que una

tercera persona tome notas durante la entrevista o grabar la entrevista en cinta de vídeo o audio, siempre que el entrevistado esté de acuerdo.

- **Terminación**: se debe recapitular sobre la entrevista para confirmar que no ha habido confusiones en la información recogida, agradecer al entrevistado su colaboración y citarle para una nueva entrevista si fuera necesario, dejando siempre abierta la posibilidad de volver a contactar para aclarar dudas que surjan al estudiar la información o al contrastarla con otros entrevistados.

Análisis de las entrevistas

Una vez realizada la entrevista es necesario leer las notas tomadas, pasarlas a limpio, reorganizar la información, contrastarla con otras entrevistas o fuentes de información, etc. Una vez elaborada la información, se puede enviar al entrevistado para confirmar los contenidos. También es importante evaluar la propia entrevista para determinar los aspectos mejorables.

Brainstorming

El *brainstorming* o tormenta de ideas es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios [Raghavan].

Este método comenzó en el ámbito de las empresas, aplicándose a temas tan variados como la productividad, la necesidad de encontrar nuevas ideas y soluciones para los productos del mercado, encontrar nuevos métodos que desarrollen el pensamiento creativo a todos los niveles, etc. Pero pronto se extendió a otros ámbitos, incluyendo el mundo de desarrollo de sistemas; básicamente se busca que los involucrados en un proyecto desarrollen su creatividad, promoviendo la introducción de los principios creativos.

Las sesiones de brainstorming suelen estar formadas por un número de cuatro a diez participantes, uno de los cuales es el *jefe* de la sesión, encargado más de comenzar la sesión que de controlarla.

Como técnica de elicitación de requerimientos, el brainstorming puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de elicitación, cuando los requerimientos son todavía muy difusos.

Fases del brainstorming

En el brainstorming se distinguen las siguientes fases [Raghavan]:

Preparación: la preparación para una sesión de brainstorming requiere que se seleccione a los participantes y al jefe de la sesión, citarlos y preparar la sala donde se llevará a cabo la sesión. Los participantes en una sesión de brainstorming para elicitación de requerimientos son normalmente clientes, usuarios, ingenieros de requerimientos,

desarrolladores y, si es necesario, algún experto en temas relevantes para el proyecto.

Generación: el jefe abre la sesión exponiendo un enunciado general del problema a tratar, que hace de *semilla* para que se vayan generando ideas. Los participantes aportan libremente nuevas ideas sobre el problema *semilla*, bien por un orden establecido por el jefe de la sesión, bien aleatoriamente. El jefe es siempre el responsable de dar la palabra a un participante. Este proceso continúa hasta que el jefe decide parar, bien porque no se están generando suficientes ideas, en cuyo caso la reunión se pospone, bien porque el número de ideas sea suficiente para pasar a la siguiente fase. Durante esta fase se deben observar las siguientes reglas:

- Se prohíbe la crítica de ideas, de forma que los participantes se sientan libres de formular cualquier idea.
- Se fomentan las ideas más *avanzadas*, que aunque no sean factibles, estimulan a los demás participantes a explorar nuevas soluciones más creativas.
- Se debe generar un gran número de ideas, ya que cuantas más ideas se presenten más probable será que se generen mejores ideas.
- Se debe alentar a los participantes a combinar o completar las ideas de otros participantes.

Consolidación: en esta fase se deben organizar y evaluar las ideas generadas durante la fase anterior. Se suelen seguir tres pasos:

- **Revisar ideas**: se revisan las ideas generadas para clarificarlas. Es habitual identificar ideas similares, en cuyo caso se unifican en un solo enunciado.
- **Descartar ideas**: aquellas ideas que los participantes consideren excesivamente *avanzadas* se descartan.
- **Priorizar ideas**: se priorizan las ideas restantes, identificando las absolutamente esenciales, las que estarían bien pero que no son esenciales y las que podrían ser apropiadas para una próxima versión del sistema a desarrollar.

Documentación: después de la sesión, el jefe produce la documentación oportuna conteniendo las ideas priorizadas y comentarios generados durante la consolidación.

Casos de Uso

Los casos de uso son una técnica para la especificación de requerimientos funcionales propuesta inicialmente en [Jacobson] y que actualmente forma parte de la propuesta de UML [Booch99].

Un caso de uso es la descripción de una secuencia de interacciones entre el sistema y uno o más *actores* en la que se considera al sistema como una caja negra.

Los casos de uso son una técnica para especificar el comportamiento de un sistema: "Un caso de uso es una secuencia de interacciones entre un sistema y actores que usan alguno de sus servicios" [Scheneider].

Los actores son personas u otros sistemas que interactúan con el sistema cuyos requerimientos se están describiendo. Un actor puede participar en varios casos de uso y un caso de uso puede estar relacionado con varios actores.

Los casos de uso presentan ciertas ventajas sobre la descripción meramente textual de los requerimientos funcionales [Firesmith97], ya que facilitan la elicitación de requerimientos y son fácilmente comprensibles por los clientes y usuarios. Además, pueden servir de base a las pruebas del sistema y a la documentación para los usuarios.

La idea de especificar un sistema a partir de su interacción con el ambiente es original de McMenamin y Palmer, dos precursores del análisis estructurado, que en 1984 definieron un concepto muy parecido al del caso de uso: el *Evento*. Para McMenamin y Palmer, un evento es algo que ocurre fuera de los límites del sistema, ante lo cual el sistema debe responder.

Sin embargo, existen algunas diferencias entre los casos de uso y los eventos. Las principales son:

- Los eventos se centran en describir qué hace el sistema cuando el evento ocurre, mientras que los casos de uso se centran en describir cómo es el diálogo entre el usuario y el sistema.

- Los eventos son "atómicos": se recibe una entrada, se la procesa, y se genera una salida, mientras que los casos de uso se prolongan a lo largo del tiempo mientras dure la interacción del usuario con el sistema. De esta forma, un caso de uso puede agrupar a varios eventos.

- Para los eventos, lo importante es qué datos ingresan al sistema o salen de él cuando ocurre el evento (estos datos se llaman datos esenciales), mientras que para los casos de uso la importancia del detalle sobre la información que se intercambia es secundaria. Según esta técnica, ya habrá tiempo más adelante en el desarrollo del sistema para ocuparse de este tema.

Los casos de uso combinan el concepto de evento del análisis estructurado con otra técnica de especificación de requerimientos bastante poco difundida: aquella que dice que una buena forma de expresar los requerimientos de un sistema es escribir su manual de usuario antes de construirlo [Howes]. Esta técnica, si bien ganó pocos adeptos, se basa en un concepto muy interesante: al definir requerimientos, es importante describir al sistema desde el punto de vista de aquél que lo va a usar, y no desde el punto de vista del que lo va a construir. De esta forma, es más fácil validar que los requerimientos documentados son los verdaderos requerimientos de los usuarios, ya que éstos comprenderán fácilmente la forma en la que están expresados.

Los Casos de Uso y UML

A partir de la publicación del libro de Jacobson, gran parte de los más reconocidos especialistas en métodos Orientados a Objetos coincidieron en considerar a los casos de uso como una excelente forma de especificar el comportamiento externo de un sistema. De esta forma, la notación de los casos de uso fue incorporada al lenguaje estándar de modelado UML (Unified Modeling Language) propuesto por Ivar Jacobson, James Rumbaugh y Grady Booch, tres de los precursores de las metodologías de Análisis y Diseño Orientado a Objetos, y avalado por las principales empresas que desarrollan software en el mundo. UML va en camino de convertirse en un estándar para modelado de sistemas de software de amplia difusión.

A pesar de ser considerada una técnica de Análisis Orientado a Objetos, es importante destacar que los casos de uso poco tienen que ver con entender a un sistema como un conjunto de objetos que interactúan, que es la premisa básica del análisis orientado a objetos "clásico". En este sentido, el éxito de los casos de uso no hace más que dar la razón al análisis estructurado, que propone que la mejor forma de empezar a entender un sistema es a partir de los servicios o funciones que ofrece a su entorno, independientemente de los objetos que interactúan dentro del sistema para proveerlos.

Como era de esperar, es probable que en el futuro los métodos de análisis y diseño que prevalezcan hayan adoptado las principales ventajas de todos los métodos disponibles en la actualidad (estructurados, métodos formales, métodos orientados a objetos, etc.).

Otras técnicas

Un ejemplo son las *técnicas para facilitar la especificación de la aplicación (*TFEA) [Pressman], cuyos dos enfoques más populares son Joint Application Development (JAD – Desarrollo Conjunto de Aplicaciones), desarrollado por IBM, y The METHOD (EL METODO), desarrollado por Perfomance Resources, Inc., Falls Church, VA.

La técnica denominada *JAD* (*Joint Application Development*), desarrollada por IBM en 1977, es una alternativa a las entrevistas individuales que se desarrolla a lo largo de un conjunto de reuniones en grupo durante un periodo de 2 a 4 días. En estas reuniones se ayuda a los clientes y usuarios a

formular problemas y explorar posibles soluciones, involucrándolos y haciéndolos sentirse partícipes del desarrollo.

Esta técnica se base en cuatro principios [Raghavan]: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación (diagramas, transparencias, multimedia, herramientas CASE, etc.), mantener un proceso organizado y racional y una filosofía de documentación *WYSIWYG* (*What You See Is What You Get*, *lo que se ve es lo que se obtiene*), por la que durante las reuniones se trabaja directamente sobre los documentos a generar.

Debido a las necesidades de organización que requiere y a que no suele adaptarse bien a los horarios de trabajo de los clientes y usuarios, esta técnica no suele emplearse con frecuencia, aunque cuando se aplica suele tener buenos resultados, especialmente para elicitar requerimientos en el campo de los sistemas de información [Raghavan].

En comparación con las entrevistas individuales, presenta las siguientes ventajas:

- Ahorra tiempo al evitar que las opiniones de los clientes se contrasten por separado.
- Todo el grupo, incluyendo los clientes y los futuros usuarios, revisa la documentación generada, no sólo los ingenieros de requerimientos.
- Implica más a los clientes y usuarios en el desarrollo.

Podríamos analizar muchos métodos que ofrece esta técnica, pero todos enfocan hacia las siguientes directrices básicas:

> *Se efectúa una reunión en un lugar neutral, se establecen reglas para la preparación y participación, se elabora una agenda, se elige un facilitador, se utiliza un mecanismo de comunicación y primeramente el objetivo principal debe ser identificar el problema.*

En el cuadro 1 resumimos Como resumen podemos presentar las principales ventajas y desventajas de cada una de las técnicas utilizadas en las etapas de la Ingeniería de Requerimientos.

Técnica	Ventajas	Desventajas
Entrevistas y Cuestionarios	• Mediante ellas se obtiene una gran cantidad de información adecuada a través del usuario. • Pueden ser usadas para obtener una visión general del dominio del problema. • Son flexibles. • Permiten combinarse con	• La información obtenida al principio puede ser redundante o incompleta. • Si el volumen de información manejado es alto, requiere mucha organización de parte del analista, así como la habilidad para tratar y comprender el

	otras técnicas.	comportamiento de todos los involucrados.
Tormenta de Ideas	• La producción de ideas en grupos puede ser más efectiva que la individual. • Aflora una gran cantidad de ideas sin ataduras.	• Es necesaria una buena compenetración del grupo participante.
Casos de Uso	• Representan los requerimientos desde el punto de vista del usuario. • Permiten representar más de un rol para cada afectado. • Identifica nuevos requerimientos , dentro de un conjunto de requerimientos.	• En sistemas grandes, toma mucho tiempo definir todos los casos de uso. • El análisis de calidad depende de la calidad con que se haya hecho la descripción inicial.
JAD	• Ahorra tiempo, elimina retrasos del proceso y mejora la calidad del sistema. • Es una de las mejores maneras de reducir los errores arrastrados de los resultados de los requerimientos iniciales. • Con la participación de gerentes y usuarios apropiados, el riesgo cultural típico se mitiga. • Evita funcionalidad sobredimensionada. • Evita que los requerimientos sean demasiado específicos o demasiados vagos, que son dos problemas comunes en el análisis.	• Es costoso Involucrar al patrocinador del proyecto y gerente, experto en la tecnología, y expertos de la materia, como parte del equipo del proyecto.

Cuadro 1: Principales ventajas y desventajas de las técnicas de elicitación.

Es importante destacar, que independientemente a la técnica de elicitación que se utilice y durante todo el proceso de desarrollo del software es indispensable mantener una adecuada comunicación entre los participantes.

Para esto, un enfoque valioso es la utilización del mismo léxico. En la fase de elicitación, será tarea del analista ocuparse de homogeneizar el vocabulario, centrándolo en el léxico propio del cliente [Leite89], esto evitará que el analista utilice el lenguaje común en su medio ambiente. Simultáneamente, el cliente trasmitirá sus necesidades en su lenguaje habitual.