

Capítulo VI

Análisis de requisitos

Toni Granolles i Saltiveri y Jesús Lorés Vidal

El concepto que en última instancia mide la calidad de un sistema de software viene determinado a partir de la concordancia entre sus requisitos y el mayor o menor grado de su consecución.

Hasta hace poco, se interpretaba que los requisitos del sistema sólo hacían referencia a sus funcionalidades. Actualmente se está dando mayor importancia a aspectos relacionados con el uso de estos sistemas, aspectos directamente relacionados con la eficiencia, la efectividad y la satisfacción percibidas por parte de los usuarios finales.

Así pues, la definición de los requisitos de los sistemas es una tarea sumamente importante que tiene una repercusión directa con el sistema finalmente implementado. Si esta definición no se realiza adecuada y minuciosamente, y si no se le dedican enormes esfuerzos desde el primer momento, el sistema final denotará dolencias funcionales y de usabilidad y, lo que suele ser peor, repercutirá en el aspecto negativo en el coste económico del sistema.

1. Introducción al análisis de requisitos

A. Sutcliffe (2002) comenta en la introducción de su libro que los requisitos son una parte ubicua de nuestras vidas que está completamente relacionada con la comunicación.

“El problema requisitos-comunicación aparece constantemente cuando nos esforzamos en expresar nuestras propias necesidades a otras personas.”

Este autor argumenta que las personas detectamos por primera vez el problema tan sólo unas horas después de nacer: lloramos para expresar unos requisitos físi-

cos que aunque suelen ser para reclamar alimento, los padres no tienen la certeza de que el motivo sea éste. Aparece un problema de comunicación en el que una persona (el bebé) comunica (llorando) a otras personas (sus padres) sus requisitos, que éstos no siempre son capaces de interpretar; el grito de un bebé es ambiguo.

Los requisitos en la ingeniería del software adquieren una importancia relevante, puesto que constituye la manera de relacionar las necesidades de los usuarios de los sistemas con los que los tienen que construir.

1.1. Requisitos, ¿qué son?

Lo primero que nos preguntaremos será, precisamente, qué son los requisitos de un sistema interactivo.

Formalmente, los *requisitos* se definen como las descripciones de cómo el sistema debe comportarse, la información acerca del dominio de aplicación, las restricciones operativas del sistema o las especificaciones de las propiedades o atributos del sistema (Kotonya, 1997). Con los requisitos se pretende averiguar qué es lo que la gente quiere de un sistema y entender cuáles son sus necesidades en términos de diseño.

En el ámbito de los sistemas basados en ordenadores, entender sobre los requisitos no se limita tan sólo a entender las necesidades de los usuarios, sino que se extiende a entender las implicaciones del dominio y conocer aquello que es realizable. Por lo tanto, analizar los requisitos de un sistema interactivo supone determinar, enumerar y clasificar todas las características, capacidades y restricciones que éste debe cumplir o a las que se ve sometido.

Los requisitos de los sistemas interactivos suelen estar enfocados a lo que debe hacer el sistema y no a cómo debe hacerlo.

Suelen dividirse básicamente en requisitos funcionales y no funcionales, aunque también obedecen a criterios de prioridad (esenciales, deseados y deseados pero de baja prioridad –“estaría bien tenerlos”–).

Ejemplo (Kotonya, 1997)

- El sistema debe tener registrado todo el material de la librería, incluyendo libros, series, periódicos y revistas, cintas de vídeo y audio, informes, colecciones de transparencias, disco de ordenador, y CD-ROM.

- El sistema debe permitir buscar un artículo por título, autor, o ISBN.
- La interfaz del usuario debe implementarse utilizando un navegador (*browser*) de Internet.
- El sistema debe soportar un mínimo de veinte transacciones por segundo.
- Las facilidades que el sistema dispone para el público en general deberán requerir un máximo de diez minutos para su explicación.
- El lenguaje de programación del sistema debe ser Java.

1.2. Importancia del análisis de requisitos

Esta fase es tan inmensamente crítica e importante que de la misma dependerá la buena continuación del proyecto. Jared Spool (2002) argumenta que un buen trabajo en esta fase repercutirá directamente en una disminución del número de iteraciones necesarias para conseguir el proceso global. Esto, en definitiva, es dinero, ya que se invertirá menos tiempo y recursos.

El desarrollo del software –como casi cualquier otra disciplina o ámbito de la vida– tiene sus “mitos”, los cuales, cuestionables o no, tienen cierta influencia en el desarrollo de una aplicación interactiva.

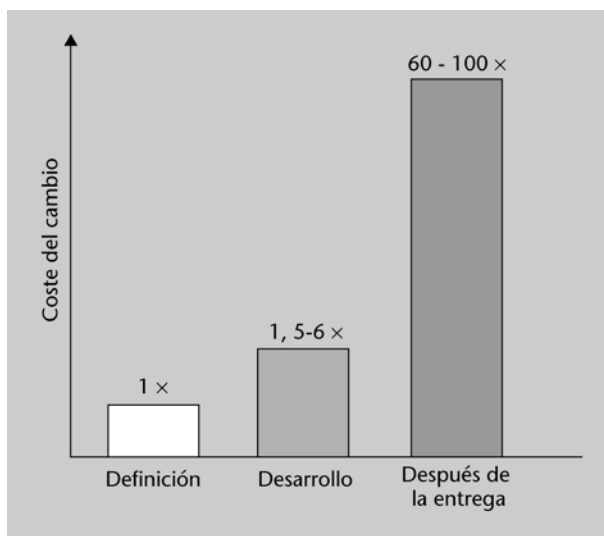
Encontramos mitos tanto desde el punto de vista del desarrollador y del gestor como del cliente.

Centrémonos en estos últimos. Los mitos del cliente conducen a que éste se cree una falsa expectativa acerca del producto final, y lo llevan irreversiblemente a la insatisfacción hacia la solución aportada por el desarrollador del software.

Éstos son los mitos más relevantes con relación a los clientes (Pressman, 2000).

1) *Mito*: basta con una declaración general de los objetivos para comenzar a escribir los programas; podemos dar los detalles más adelante. *Realidad*: una mala definición inicial es la principal causa del trabajo estéril en software.

2) *Mito*: los requisitos del proyecto cambian continuamente, pero los cambios pueden acomodarse fácilmente, ya que el software es flexible. *Realidad*: es cierto que los requisitos cambian, pero el impacto del cambio varía según el momento en el que éste se introduzca.

Figura 6.1. Impacto de los cambios en el coste final

Este gráfico ilustra perfectamente el impacto que producen en el coste final los cambios con relación a la fase del desarrollo del producto.

Un estudio hecho por el grupo Standish Group (Standish Group Report, 1995) sobre una muestra de ocho mil proyectos de trescientas cincuenta empresas demostró que el 31% de los proyectos se cancelaban antes de finalizar, y como máximo el 16% lograban ser entregados dentro de los plazos acordados y de los presupuestos estimados. El mismo informe revela que la principal causa (un 13,1%) del fracaso de los proyectos era debida a errores en la definición de los requisitos. Aunque el estudio fue realizado en el año 1995, el reflejo de la importancia de esta fase se mantiene totalmente válido hoy en día.

De entre los proyectos de software que no llegan a su fin, el mayor tanto por ciento de culpa de estos fracasos se debe a un análisis incompleto de los requisitos.

El proyecto europeo PRUE llevado a cabo por los ingenieros de Serco Usability, liderados por el reconocido ingeniero en usabilidad Niegel Bevan, pone datos más recientes y contundentes sobre la mesa:

“La falta del cumplimiento de los requisitos de los usuarios fue la razón principal de los costes excesivos y de los retrasos en los que se incurrió a la hora de instalar en Inglaterra un nuevo sistema de software de emisión de pasaportes desarrollado por Siemens.”

Public Accounts Committee (1999). *Improving the delivery of government IT projects*.

Por lo tanto, de un análisis pobre de los requisitos de un sistema deriva una multitud de problemas, como por ejemplo:

- 1) Los requisitos no reflejan las necesidades reales que el cliente desea que el sistema realice.
- 2) Los requisitos son inconsistentes o incompletos.
- 3) La repercusión de los cambios en los requisitos una vez ya han sido detallados y acordados suele ser económicamente cara.
- 4) Suele haber malentendidos entre los clientes, los que hacen el análisis de requisitos y los ingenieros de software que desarrollan o mantienen el sistema.

1.3. Personas, comunicación y requisitos

Una de las principales razones por las que no resulta fácil realizar el análisis de los requisitos se debe a que el simple hecho de tratar con personas es, en esencia, una tarea difícil. Cada uno tenemos puntos de vista particulares e ideas propias; en definitiva, nuestros propios modelos mentales. Realizamos muchas acciones, de manera espontánea o rutinaria, inconscientemente, mientras que otras las ejecutamos voluntariamente en la privacidad (Sutcliffe, 2002).

Lo que ocurre es que la práctica del análisis de requisitos se enfrenta con los problemas propios de la comunicación humana, como son los siguientes.

- 1) *El conocimiento tácito*: las acciones que las personas realizamos con cierta frecuencia son muy difíciles, y a veces es imposible describir cómo procedemos en su ejecución. Esto simplemente se debe a que a causa del conocimiento adquirido de las mismas, nuestra mente no es consciente de que las estamos llevando a cabo, se realizan de manera automática.
- 2) *La ambigüedad*: a pesar de que disponemos de suficientes mecanismos de comunicación humana, frecuentemente expresamos nuestras conductas y pensamientos inadecuadamente, de manera que obtenemos expresiones enigmáticas que dan lugar a varias posibles interpretaciones.

3) *Las actitudes y las opiniones particulares*: las vivencias particulares, las creencias religiosas y/o políticas, la tradición familiar y cultural o las aficiones son algunos de los aspectos que influyen en la manera de llevar a cabo determinadas acciones y de dirigir las opiniones sobre las cosas.

Y por este motivo, aunque contemos con usuarios honestos y cooperativos, difícilmente conseguiremos un conjunto de requisitos preciso y “honesto”, y la labor del equipo que realiza el análisis de los requisitos precisa de expertos en campos de conocimiento como la psicología y la sociología.

1.4. La ingeniería de los requisitos

La ingeniería de los requisitos, IR, es un término relativamente nuevo que cubre el conjunto global de las actividades relacionadas con el descubrimiento, la documentación y el mantenimiento del conjunto de requisitos concernientes a un sistema interactivo.

Hay distintas definiciones de ingeniería de los requisitos (Sommerville, 1989; Dubois, 1983; Bubenko, 1993), y la que a nuestro parecer proporciona el mejor enunciado es la siguiente:

La *ingeniería de los requisitos* es la rama de la IS que se preocupa de los objetivos del mundo reales, de las funcionalidades y de las restricciones de los sistemas de software. También se preocupa de la relación entre estos factores y necesita especificaciones del comportamiento del software y sobre su evolución en el tiempo (Zave, 1995).

Los objetivos principales de la IR pueden resumirse en los siguientes puntos (Sutcliffe, 2002):

- 1) Capturar un conjunto completo de requisitos de los usuarios.
- 2) Analizar detalladamente los requisitos de los usuarios, encontrar todas sus implicaciones y comprenderlas.
- 3) Especificar cómo estos requisitos deberán manifestarse durante el diseño del sistema.
- 4) Completar el análisis de los requisitos con un conjunto de restricciones aceptable en términos temporales y económicos.

1.5. Documentar el análisis de requisitos

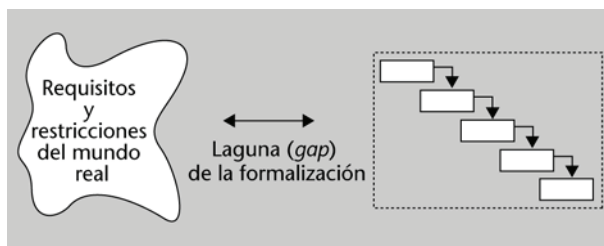
El documento del análisis de requisitos es un documento formal cuya función consiste en comunicar los requisitos a los clientes, ingenieros y gerentes.

El documento de requisitos describe:

- 1) Los servicios y funciones que el sistema debe proporcionar.
- 2) Las restricciones bajo las cuales debe operar.
- 3) El conjunto completo de propiedades del sistema, por ejemplo, las restricciones de las propiedades emergentes del sistema.
- 4) Definiciones de otros sistemas con los cuales el sistema que se debe desarrollar tiene que operar o con los que incluso debe integrarse.
- 5) Información sobre el dominio de la aplicación del sistema, por ejemplo, cómo se deben realizar tipos particulares de computación.
- 6) Restricciones sobre el proceso utilizado a la hora de desarrollar el sistema.
- 7) Descripción del hardware sobre el que deberá “correr” el sistema.

Formalizar los requisitos y restricciones del mundo real de manera consistente y precisa para que puedan ser tratados convenientemente no es una tarea fácil. La dificultad principal que encontramos es la “laguna (*gap*) de la formalidad”, y éste es precisamente el aspecto que debemos ser capaces de minimizar (Dix y otros, 1993, pág. 155).

Figura 6.2. Laguna, o *gap*, de la formalización del mundo real con un diseño estructurado



Esta formalización se puede especificar con descripciones en lenguaje natural, que, como hemos visto, son propensas a malas interpretaciones, o especificaciones basadas en notaciones o lenguajes formales, que pese a ser muy concretos y

precisos, difícilmente resultan accesibles para las personas no versadas en la ingeniería del software.

Opinamos que adoptar técnicas de documentación bien estructuradas y definidas como las propuestas en Durán (2002), o por la IEEE 830 (Mazza y otros, 1994), genera documentos expresados en lenguaje natural convenientemente organizados y, por lo tanto, se evita tener informes que contengan cantidades extensas de “texto que nadie desea leer” o “no es capaz de comprender”.

1.6. Análisis

La fase del análisis de requisitos vista desde la óptica de la ingeniería del software (IS) “clásica” establece los servicios que el sistema debe proporcionar y las restricciones bajo las cuales tiene que operar. Se establecen los requisitos que determinan qué debe hacer el sistema y cómo debe hacerlo. Los tipos de requisitos suelen ser, como ya se ha mencionado:

- 1) *Funcionales*, que describen una funcionalidad o un servicio del sistema.
- 2) *No funcionales*, que suelen ser restricciones al sistema (por ejemplo, tiempo de respuesta) o para su proceso de desarrollo (utilizar un determinado lenguaje).

La fase del análisis de requisitos en el modelo de la ingeniería de la usabilidad incluye todo lo que recoge la IS y le añade una serie de factores que hay que tener en cuenta y que, de realizarse, garantizarán el desarrollo de un producto con un grado mucho más alto en lo que respecta a su usabilidad y accesibilidad.

A lo largo de este capítulo veremos cuáles son estos factores y de qué mecanismos disponemos para materializarlos.

Los clientes no pueden apreciar sus necesidades reales hasta que ven e interactúan con las opciones de las que disponen (muchos requisitos son descubiertos una vez los usuarios interactúan con el prototipo ejecutable¹ o *runnig prototype*, Brooks, 1995); o lo que es más, no aprecian las nuevas tecnologías has-

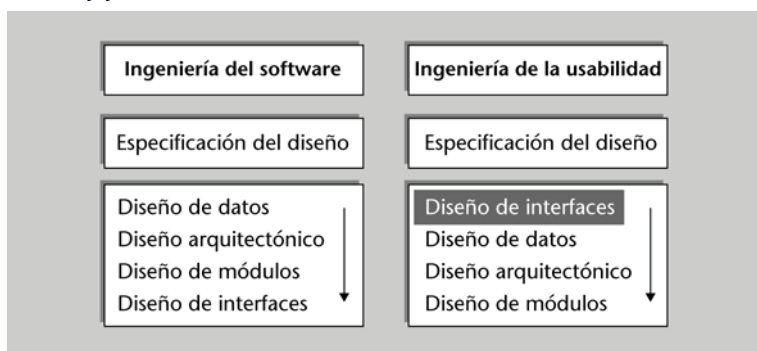
1. Es un prototipo que permite al usuario ejecutar una serie de tareas (que en función del objetivo de la evaluación serán unas u otras).

ta que las prueban, ¡simplemente porque las desconocen! No se sabe lo que se ha desarrollado hasta que no se ha hecho el primer sistema malo² o *wrong system* (Rosson y Carroll, 2002).

De las afirmaciones anteriores se deduce que será imposible determinar todos estos objetivos en una primera fase o visita con el cliente. Deberá ser el equipo de desarrollo el que, mediante técnicas de prototipado y evaluación proporcionadas por el modelo de proceso –junto con la aplicación de las tareas de protección de la gestión de la configuración del software (GCS)–, defina toda esta información tan detalladamente como sea posible. Aun así, los cambios son inevitables, y por este motivo tenemos la obligación de reducir su número.

El modelo de proceso MPEu+a aplica la gestión de la configuración del software mediante un cambio decisivo para conseguir el diseño deseado centrado en el usuario: en la ingeniería del software, el diseño de las interfaces se emprende después de haber especificado el diseño de los datos, el arquitectónico y el de los módulos, mientras que en el MPEu+a el diseño de las interfaces pasa a primer término y el resto es condicionado, si procede, por la interfaz.

Figura 6.3. Diseño de interfaces de usuario desde el punto de vista de la IS y de la IU



Este cambio, aunque parece menor e insignificante, es muy importante, ya que con mucha frecuencia se producen cambios en, por ejemplo, la estructura de las bases de datos, debidos a una especificación que viene impuesta por un

2. La primera versión de un sistema siempre es mala. Si ya es una versión un poco avanzada del sistema, desearíamos no tirarla (el caso más habitual), y arrastraremos sus defectos durante toda su vida útil.

requisito de la interfaz. Si no se ataca el problema desde este punto de vista, el código presentará inconsistencias.

Idealmente, los requisitos y el diseño son dos cosas completamente separadas, la realidad no es así y existe una fuerte relación entre ambos elementos.

2. La recogida de requisitos en el modelo de proceso

La fase del análisis de requisitos en el MPIu+a se interesa por todos aquellos aspectos que destacan todos los elementos relacionados con la IPO en la que el usuario es el centro del diseño.

La interfaz resultante se tendrá que adaptar al modelo mental del usuario y no al del programador o diseñador que la ha creado.

En esta fase se estudia el dominio del problema, y se interactúa con clientes y usuarios para obtener y registrar información sobre sus necesidades.

Las actividades que esta fase contempla para una buena captura de requisitos son las siguientes.

2.1. Análisis etnográfico

“Las personas con frecuencia consideran difícil describir lo que hacen, es natural de ellas mismas. Muchas veces, la mejor manera de entenderlo es observarlas en su trabajo; es decir, haciendo un trabajo etnográfico.”

Kotonya y Sommerville (1997).

La etnografía es un término que deriva de la antropología. Se puede considerar como un método de trabajo de ésta. Se traduce etimológicamente como el estudio de las etnias y significa el análisis del modo de vida de una raza o grupo de individuos, mediante la observación y descripción de lo que la gente hace, cómo se comportan y cómo interactúan entre sí, para describir sus creencias, valores, motivaciones, perspectivas y cómo éstos pueden variar en diferentes momentos y circunstancias. Podríamos decir que describe las múltiples formas de vida de los seres humanos.

Para “hacer etnografía” de manera transparente es necesario adentrarse en el grupo que se quiere estudiar para aprender su lenguaje y sus costumbres –es decir, para hacer interpretaciones adecuadas de los acontecimientos debere-mos tener en cuenta sus significados. En definitiva, no se trata de hacer una fotografía con los detalles externos, sino que hay que ir más atrás y analizar los puntos de vista de los sujetos y las condiciones historicosociales en las cua-les se dan.

Las herramientas de investigación etnográfica pueden responder a aquellas cuestiones sobre organizaciones y mercados a las que otros métodos no pueden res-ponder. En los últimos tiempos, las grandes compañías de desarrollo de software se han dado cuenta de la gran capacidad de información útil que la etnografía puede aportar para sus desarrollos, e incluso han incorporado etnógrafos en sus propios equipos de trabajo.

En la tabla 6.1 se resumen las líneas básicas de la etnografía (*ethnography guideli-nes*) aplicadas al desarrollo de sistemas interactivos (Kotonya y Sommerville, 1997).

Tabla 6.1. Líneas básicas de la etnografía

-
- Asumir que las personas realizan correctamente su trabajo y que con mucha frecuencia intentan hacerlo de manera que difiera de la estándar.
 - Disponer y pasar tiempo suficiente en conocer a las personas e intentar establecer una buena relación con ellas.
 - Guardar notas detalladas de todas las actividades de su trabajo. Analizarlas y sacar conclusiones con ellos mismos.
 - Combinar la simple observación con entrevistas abiertas.
 - Organizar sesiones regulares de resumen en las que el etnógrafo habla con los usuarios fuera del proceso.
 - Combinar la etnografía con otras técnicas de adquisición (*elicitation*).
-

A modo de conclusión podemos destacar que realizar un estudio etnográfico ayudará de manera altamente determinante a algunas de las otras fases del aná-lisis de requisitos, puesto que, por ejemplo, seremos capaces de:

1) Describir el contexto, el puesto de trabajo y cómo realizan las personas sus tareas.

2) Detallar y relacionar los objetos que estas personas utilizan (directa o indirectamente).

3) Aumentar la información relativa a la organización de las tareas y a su consecución. La experiencia del día a día de los trabajadores aporta gran información, debido a que cada uno ve una misma tarea de diferente manera.

4) Aumentar cualitativa y cuantitativamente las conclusiones de una observación de campo.

El principal problema de utilizar esta técnica es que, si quiere hacerse correctamente, el proceso es muy lento, pues la principal herramienta de la investigación es uno mismo, y conseguir que esta “herramienta” sea percibida como natural en un contexto “extraño” no se asimila de forma inmediata.

2.2. Análisis de implicados (*stakeholders*)

Veamos una serie de definiciones de autores destacados del término *implicado* en el contexto del desarrollo de sistemas interactivos para entender qué son los implicados y cuál es su relevancia:

1) Un implicado en una organización es (por definición) cualquier grupo o individuo que puede afectar a la consecución de los objetivos de la organización o puede ser afectado por ésta (Freeman, 1984).

2) Los implicados son aquellos participantes (en el proceso de desarrollo) junto con otros individuos, grupos u organizaciones cuyas acciones pueden influir o ser influidas por el desarrollo y uso del sistema, ya sea directa o indirectamente (Pouloudi, 1999).

3) Los implicados son personas u organizaciones que serán afectadas por el sistema y que tienen influencia directa o indirecta en los requisitos del sistema (Kotonya y Sommerville, 1997).

4) El implicado es cualquier persona cuyo trabajo será alterado, aquel que proporciona u obtiene información de esto, o cuyo poder o influencia dentro de la organización incrementará o decrementará (Dix y otros, 1993).

Clasificación

Existen varias propuestas en cuanto a la clasificación de los implicados. La más simple se basa en dividirlos entre aquellos que utilizarán el sistema directa o indirectamente (Newman y Lamming, 1995):

- Ingenieros de software responsables del desarrollo.
- Usuarios finales del sistema.
- Directores de los usuarios que son responsables de su trabajo.
- Los que están relacionados con el desarrollo del sistema.
- Socios y/o proveedores tecnológicos.

Y una de las más detalladas es la que identifica cuatro categorías de implicados (MacCaulay, 1994):

- Los responsables del diseño y el desarrollo.
- Los que tienen un interés financiero o económico (responsables de la venta o compra).
- Los responsables de su implantación y mantenimiento.
- Los que tienen interés sobre su uso (los usuarios).

Hay otras clasificaciones de los implicados, pero sea cual sea, en todas queda muy claro que el *usuario final* de la aplicación es un implicado más. Un implicado directo y muy especial, de ahí que se trate el análisis de sus características en un punto aparte (que a continuación veremos) del conjunto global de las tareas del análisis de las restricciones.

Identificación

Los implicados normalmente, aunque no siempre, se identifican por sus roles más que individualmente, y entre éstos son tan importantes los actores que están principalmente interesados en el problema que hay que resolver (por ejemplo, los usuarios finales) como los interesados en la solución (por ejemplo, diseñadores del sistema).

La identificación de los implicados no es una tarea obvia; más bien todo lo contrario (aunque cierta literatura de los sistemas de información así lo cree), y con este objetivo existen varias aproximaciones metodológicas.

En todo proyecto existe un grupo de implicados “elementales”, obvios, cuya identificación resulta muy fácil. Debemos realizar un esfuerzo especial para la identificación de los que no son tan elementales.

El objetivo es encontrar a todos los implicados, incluso aquellos que pueden influir negativamente en el proyecto.

Reuniones con los implicados

Una vez identificados los implicados, se debe proceder a obtener toda la influencia relativa al proyecto que éstos pueden aportar al mismo. Una de las maneras más habituales de obtener esta información es la realización de una serie de reuniones de implicados (*stakeholders meetings*).

Propuesta de Nigel Bevan de cómo se hace una reunión de implicados (Bevan, 2000)

1) Planificar una reunión de un solo día e invitar a los implicados que tienen conocimiento sobre las intenciones de los usuarios y de su uso, incluyendo:

- a) Responsable del negocio (*business manager*)
- b) Responsable del proyecto (*project manager*)
- c) Usuario (o usuarios) representativos
- d) Responsables de marketing
- e) Desarrollador o desarrolladores
- f) Responsables de la formación
- g) Responsables del mantenimiento

2) El miembro del equipo de desarrollo encargado de esta reunión preparará una lista con todas las cuestiones que deben ser tratadas durante la reunión.

3) Las principales recomendaciones para preparar esta reunión son:

- a) Antes de la reunión
 - Identificar puntos clave que necesitamos explorar.
 - Proporcionar la agencia y la lista con los puntos que se deben tratar a todos los participantes.
- b) Durante la reunión
 - Una vez discutidos los puntos clave, deberemos intentar obtener un consenso en aquellos puntos en los que haya habido incertidumbre o disconformidad.
 - Si se ha echado en falta información, deberá acordarse cómo se obtendrá.
 - Realizar una discusión sobre temas menores.

- c) Después de la reunión
 - Obtener toda la información que faltaba.
 - Si la información no es fácil de obtener, organizar un estudio de campo para observar a los usuarios en su ambiente de trabajo. Por ejemplo, en un sistema educativo investigar cómo se realizan la enseñanza, el aprendizaje y las actividades de soporte actualmente.
 - Proporcionar a todos los participantes un resumen con las conclusiones de la reunión.

Para finalizar este punto, un par de recomendaciones:

1) La reunión se celebrará, si es posible, antes de que los requisitos funcionales hayan finalizado, aunque la reunión es importante incluso si el diseño centrado en el usuario se ha introducido tarde en el proceso de desarrollo (el impacto en el proyecto será mayor cuanto más tarde se produzca este hecho).

2) Todos los implicados deben asistir a la primera reunión. Las reuniones posteriores suelen ser sólo con los directamente relacionados para ciertos detalles que todavía hay que detallar.³

2.3. Clasificar a los usuarios

En repetidas ocasiones hemos subrayado que el objetivo principal de la metodología propuesta es conseguir simplificar la interacción de los usuarios con el sistema. Debemos conocer muy bien tanto a los usuarios como sus características, para ser capaces de desarrollar esta capacidad y trasladarla al sistema en forma de interfaces lo bastante usables y accesibles para los usuarios, y que se adapten a sus modelos mentales.

Por este motivo, realizaremos una clasificación de los usuarios en términos de dos características distintas, pero complementarias:

a) El *perfil de usuario*, que responde al criterio de agrupar a los usuarios según sus capacidades y habilidades y que da lugar a grupos de población con características semejantes.

b) El *rol*, que se orienta a las funcionalidades del sistema.

3. M. B. Rosson y J. M. Carroll (2002). *Usability Engineering: scenario-based development of HCI* (págs. 54-58). San Francisco: Morgan Kaufmann.

2.3.1. Perfil de usuario

El diseño de los sistemas interactivos tiene dos leyes importantes: (1) conocer a los usuarios, y (2) tener claro que los usuarios no somos nosotros (Retting, 1994). Estas leyes nos provocan la necesidad de esforzarnos para conocer quiénes serán los usuarios y cuáles serán sus características y objetivos.

Seguir el diseño de un sistema interactivo centrado en el usuario supone hacer hincapié en este apartado del proyecto y poder identificar las peculiaridades, intereses, necesidades y expectativas de los usuarios primarios (aquellos que usan el sistema frecuentemente) y los secundarios (los que lo usan ocasionalmente).⁴

El objetivo principal de esta fase del análisis de requisitos es obtener una clasificación de los distintos tipos de usuarios y una descripción de las características más relevantes de esta población potencial que usará la interfaz para eliminar la separación entre aquello que los usuarios conocen y lo que necesitan conocer.

Habrà que destacar aspectos muy distintos como, por ejemplo, el grado de conocimiento/uso de equipos/programas informáticos que tienen, la experiencia profesional, el nivel de estudios, la experiencia en el puesto o tipo de trabajo, el entorno social, etc. Estas características nos servirán para poder tomar decisiones a la hora de diseñar la interfaz de usuario y para identificar las categorías de usuarios.

Identificación de los perfiles de usuarios

El perfil de los usuarios se puede obtener por distintos métodos. Los más utilizados y que presentan una efectividad probada más alta son los cuestionarios y las entrevistas. Una vez analizados, los resultados permiten identificar patrones de usuarios (características y necesidades semejantes) y reflejarlos en el sistema final.

2.3.2. Roles

Los roles determinan clases de usuarios que tienen asignados algunos subconjuntos determinados de tareas, ya sea por elección propia o como resultado

4. Algún autor distingue un tercer grupo de usuarios, terciarios, que corresponde a aquellos que se ven afectados por la introducción del sistema o que influenciarán en su compra. Esta clasificación corresponde a los implicados, pues aunque pueden influir en algùn aspecto del diseño del sistema no lo van a manipular y, por lo tanto, no se les puede catalogar como usuarios del sistema.

de la organización en la que los usuarios se encuentran (Veer, 1996). Por definición, los roles son genéricos para el mundo de las tareas. El concepto de rol está estrechamente vinculado a las tareas.

Los actores o usuarios pueden tener representaciones internas (mentales) de sus propios roles, o bien externas, mediante comportamientos simbólicos, instrumentos u objetos identificadores.

A su vez, más de un usuario puede estar involucrado en un mismo rol, y un mismo usuario puede ejercer más de un rol al mismo tiempo. Es más, los roles pueden ser temporales, negociarse entre los actores y ser aceptados o rechazados.

Además, existe el concepto de *organización*, que hace referencia a la relación entre actores y roles en un contexto determinado, que describe la estructura jerárquica y de delegación de responsabilidades entre roles, así como el papel de los actores en los distintos roles.

La estructura de los roles en una organización puede presentar distintas formas: por ejemplo, un rol puede ser un subtipo de otro rol (un delegado de ventas es al mismo tiempo un delegado), o bien un rol puede ser una parte de otro rol (una enfermera es una parte de un departamento de salud, que forma parte del departamento de personal).

2.3.3. Relación entre los perfiles de usuarios y los roles

Hay una estrecha relación entre los perfiles de usuarios y los roles que éstos adoptan. La relación entre las dos clasificaciones es del tipo $n \dots m$, es decir, que uno o más perfiles de usuario pueden estar asociados a uno o más de un rol, y al revés.

Por lo tanto, la correcta clasificación de los distintos perfiles de usuario y la determinación de los roles asociados están estrechamente vinculadas a las tareas que el sistema de software debe ser capaz de ejecutar. Si resolvemos acertadamente este entramado, la disposición de las funcionalidades se adaptará bien al modelo mental de los distintos perfiles de usuarios y favoreceremos la usabilidad del sistema.

2.4. Análisis contextual de tareas

Los sistemas interactivos no existen por separado ni trabajan aisladamente, sino que se encuentran inmersos en una instancia o situación de servicio.

El desarrollo de un sistema de software dará respuesta a una serie de actividades estrechamente vinculadas a la actividades, la organización y las costumbres de trabajo y las maneras particulares de llevar a cabo determinadas tareas. Todos estos aspectos definen el contexto de uso del sistema.

El análisis contextual de las tareas intenta hacer un estudio de las tareas de los usuarios, de cómo las llevan a cabo, de qué patrones de trabajo utilizan –si es que los utilizan– dentro del contexto en el que estas tareas se desarrollan, y llegar a entender y especificar los objetivos de los usuarios.

No se trata aquí de hacer el análisis de las tareas, sino de determinar, a partir del análisis etnográfico previo, todas las tareas que el sistema es capaz de llevar a cabo relacionadas con el contexto específico en el que se desarrollan, entendiendo como sistema el conjunto de personas, objetos, métodos y herramientas que intervienen efectivamente.

Otro aspecto destacable de la contextualización de las tareas, ejecutadas tanto por usuarios como por implicados (en condición de actores que interactúan con el sistema e intervienen en el mismo), es que compromete la estructura organizativa en la que se encuentran, y esto ayuda a cohesionar las tareas que el sistema debe llevar a cabo y el contexto en el que se ejecutan.

Finalmente, otro aspecto que habrá que tener en cuenta cuando analizamos el contexto es el entorno, que influye en el trabajo de los actores y condiciona su manera de llevarlo a cabo. Este factor también condiciona e influye en el diseño de un sistema, ya que, aunque el resultado final de una tarea sea el mismo, los pasos para llegar al mismo pueden ser muy distintos.

2.5. Objetos

Consideraremos *objeto* (o *artefacto*) cualquier cosa que intervenga, directa o indirectamente, en el proceso de interacción entre la persona y el sistema interactivo, por innecesaria que parezca.

Es importante destacar que, como podrá verse a continuación, el concepto de objeto que aquí se describe dista enormemente del concepto de objeto que un ingeniero de software dispone.

Los *objetos* pueden ser cosas físicas (un bolígrafo, una tarjeta de crédito, la puerta de entrada para acceder al cajero automático, etc.) o conceptuales (mensajes, gestos, contraseñas, firmas, etc.).

El uso de objetos o artefactos constituye una valiosa fuente para el análisis de información. Permite el afloramiento de aspectos de la interacción que, aun siendo relevantes para el desarrollo del sistema, podrían pasar por alto y emerger en el momento más inoportuno del desarrollo. El contenido y el significado de los objetos o su disposición en el entorno codifican sus estados y disparan acciones subyacentes a esta codificación: un papel en un sitio determinado de la mesa puede significar “archívame”, mientras que en otro lugar de la oficina –como por ejemplo en una bandeja junto con otros documentos– puede significar “en curso” o, incluso, en la misma bandeja con un Post-it® que sale, puede significar “para ser revisado” (Dix y otros, 2003). Lo más significativo de estas actividades es que no están recogidas en el protocolo oficial del funcionamiento del sistema, pero modifican su estado de manera tan natural que podrían tener implicaciones en el diseño.

2.6. Plataforma (posibilidades y restricciones)

Este punto va directamente relacionado con la plataforma tecnológica escogida para albergar el producto. En función de esta elección, se estudiará y documentará el conjunto de posibilidades que esta plataforma nos ofrece, así como las restricciones tecnológicas que nos impone.

Evidentemente, esto nos definirá un conjunto de opciones posibles y/o imposibles (posibilidades y/o restricciones) para tenerlas en cuenta a la hora de diseñar la interfaz de usuario.

2.7. Perfil del entorno

El entorno (o contexto) en el que se realiza un determinado trabajo suele influir muy directamente en la manera en la que éste se realiza, por lo que deberá tenerse muy en cuenta este factor a la hora de realizar el diseño de los sistemas, pues las tareas necesarias para su consecución pueden variar mucho del concepto mental que podamos tener.

Conocido y frecuente es el caso de una empresa con varias sedes en diferentes ubicaciones (entornos distintos) que para ofrecer el mismo producto, debe utili-

zar estrategias comerciales totalmente distintas unas de otras, debido a las diferencias del entorno (culturales, económicas, ambientales, etc.). Aun más, distintos centros de fabricación de una misma empresa que para producir un mismo producto acabado usan líneas y procesos de producción muy diferentes unos de otros debido, por ejemplo, a las diferencias de contratación laboral o a disponer de una determinada materia prima distinta entre los dos centros. De nuevo, el entorno en el que se realiza la actividad influye en el proceso.

2.8. Objetivos

No hay ninguna aplicación que se realice sin al menos un objetivo concreto. Una empresa, persona, grupo o institución decide desarrollar una aplicación como consecuencia de haberse marcado unos objetivos a cumplimentar y ven, por tanto, necesaria la implementación de la aplicación como herramienta para conseguirlo.

A la hora de recoger los objetivos de una aplicación tendremos en cuenta tanto los requisitos *funcionales* como los *no funcionales* (tiempos de respuesta, utilización de un determinado lenguaje de programación, etc.), e incluso los objetivos marcados en cuanto a la usabilidad y/o accesibilidad del sistema.⁵

Una vez definidos tanto los objetivos de usabilidad y/o de accesibilidad concretos para cada aplicación deberemos contrastarlos con los objetivos funcionales principalmente para analizar si alguno de los objetivos aquí definidos entra en contradicción con dichos objetivos funcionales. Si se da el caso, por cierto muy frecuente, se procederá a “negociar” la solución a adoptar para resolver el conflicto.

2.8.1. Objetivos funcionales

La lista de objetivos funcionales es, en principio, la más fácil de recoger. No porque sea un tema obvio de conseguir, sino porque es el caso más habitual de

5. Todos estos objetivos no podrán ser determinados en una primera fase o visita con el cliente. Deberá ser el desarrollador quien, ayudándose de técnicas como el prototipaje y la evaluación que el modelo de proceso proporciona junto con la aplicación de las actividades de protección, sea capaz de definir toda esta información lo más detallada posible.

cualquier aplicación interactiva: normalmente las aplicaciones sólo responden a criterios de las funcionalidades que tienen que poder realizar.

Es evidente que si las aplicaciones no son capaces de realizar dichas funcionalidades, de nada servirá el resto. Por más usable y accesible que sea un determinado sistema, de nada servirá si no realiza las tareas que se le han encomendado. Aunque si esto sucede, realmente la aplicación no puede ser usable puesto que ya vimos que en la definición de usabilidad está que los usuarios puedan conseguir objetivos específicos, o sea, los objetivos funcionales de dicho sistema.

La definición de los objetivos funcionales se basará en un listado de dichas funcionalidades donde cada una de ellas contará con las especificaciones necesarias que permitan al equipo desarrollador implementar los procesos necesarios para que el sistema pueda ofrecer los resultados esperados a sus usuarios.

2.8.2. Objetivos de usabilidad

No reincidiremos en este apartado sobre los beneficios y la necesidad que aporta la usabilidad a un sistema interactivo.

Tan sólo, y a modo de recapitulación, recordaremos que la usabilidad es vista generalmente para asegurar que los productos interactivos sean fáciles de aprender, efectivos y agradables para sus usuarios. Todo esto supone optimizar las interacciones que las personas llevan a cabo con sus productos interactivos para poder conseguir realizar sus actividades en el trabajo, la escuela, y, en definitiva, su vida cotidiana.

Concretando un poco más, los objetivos básicos de la usabilidad los podemos enumerar en la lista siguiente:

- Facilidad en el aprendizaje
- Consistencia
- Flexibilidad
- Robustez
- Recuperabilidad
- Tiempo de respuesta

- Adecuación a las tareas
- Disminución de la carga cognitiva

Un aspecto importante que hay que considerar es que suele darse el caso de que algún objetivo funcional entra en conflicto con otro de usabilidad. Esta situación se resuelve, por ejemplo, con un preciso análisis de pros y contras en una reunión con los implicados en esta situación.

2.8.3. Objetivos de accesibilidad

Por todos los motivos mencionados, la accesibilidad de cualquier sistema debe constituir por sí misma un objetivo primordial que no puede descuidarse ni dejarse a la improvisación.

Al igual que en la definición de los objetivos de usabilidad, en este subapartado no enumeraremos ninguna lista de objetivos, dado que éstos se tienen que definir para cada aplicación particular. El aspecto más importante es definirlos de manera clara y precisa de acuerdo con las discapacidades que pretendamos cubrir y, para esto, nos basaremos en los estándares y las normativas vigentes.

Ahora bien, hemos de ser conscientes de que con esta definición de objetivos nunca podremos ofrecer una funcionalidad total para todos los tipos de discapacidades, ni podremos solucionar todos los problemas relativos a una misma tipología. Por lo tanto, seamos ambiciosos, pero no nos marquemos objetivos imposibles.

Otro aspecto que habrá que tener en cuenta son los dispositivos actuales y la tecnología relacionada. Sabemos que estos dispositivos y esta tecnología no disponen de suficiente capacidad para satisfacer todas las necesidades; a pesar de esto, el avance en este campo progresa espectacularmente.

Resulta especialmente importante definir los objetivos de usabilidad y/o de accesibilidad concretos para cada aplicación. Una vez definidos, se contrastarán con los objetivos funcionales, principalmente para analizar si alguno de los objetivos aquí definidos entra en contradicción con estos objetivos funcionales. Y si se da el caso, se procederá a “negociar” la solución que hay que adoptar para resolver el conflicto.

Resumen

En este capítulo se ha intentado que el lector haya aprendido a ser consciente de que cuando se desarrolla un sistema interactivo teniendo en cuenta los parámetros y factores de usabilidad y accesibilidad, el análisis de los requisitos resulta mucho más extenso que si no los tenemos presentes.

Asimismo subrayamos, viendo el abanico de actividades que hay que desarrollar, la importancia y la necesidad de los equipos multidisciplinares para una buena recogida de requisitos para el diseño de sistemas interactivos. Con estos equipos se consigue desarrollar sistemas que se aproximan más a los modelos mentales de los usuarios finales, y que al mismo tiempo están más alejados de los modelos de los desarrolladores.

La enorme ventaja de realizar prácticas de este tipo va en beneficio exclusivo de los usuarios finales, que agradecerán disponer de sistemas que, además de ofrecerles nuevas funcionalidades que les “facilitan” distintos aspectos de su trabajo o vida cotidiana, serán fáciles de usar, lo harán eficientemente y estarán a la disposición de todas las personas, sin importar que puedan sufrir algún tipo de discapacidad.

Capítulo VII

Diseño

Toni Granollers i Saltiveri y Jesús Lorés Vidal

La fase de diseño es la segunda fase del ciclo de vida de todo proceso de desarrollo de software y, por tanto, también es la segunda en el modelo de proceso de la ingeniería de la usabilidad y de la accesibilidad (MPlu+a) en su vertiente integradora de la ingeniería del software.

Repetidamente, se llega a esta fase después de llevar a cabo actividades relacionadas con el análisis de requisitos, que proporcionan la información necesaria para que el equipo de desarrollo pueda modelizar el sistema para, posteriormente, proceder a su codificación.

Durante el desarrollo de un sistema, tanto si es nuevo como si no, una vez resueltas las funcionalidades que debe cubrir y determinado el resto de las características derivadas del contexto de la interacción, en esta fase se pasa al *diseño de la actividad* y el *diseño de la información* como principales actividades que conforman el proceso global de *diseño de la interacción*.

El *diseño de la actividad* está directamente relacionado con la especificación funcional, la tecnología y las nuevas posibilidades que el sistema ofrece para que las personas sean capaces de utilizar sistemas interactivos para la consecución de sus actividades. El diseño de la actividad se consigue a partir del análisis de las funcionalidades y las tareas necesarias que permiten llevarlas a cabo (el análisis de las tareas) y la modelización en el nivel conceptual de una aproximación al modelo mental de los usuarios –previamente incorporados como miembros activos de desarrollo. El diseño de la actividad cubre el espacio entre las funcionalidades definidas y la interfaz del usuario.

El *diseño de la información*, que tiene como objetivo principal dar apoyo a la percepción, la información y la comprensión de la información de los sistemas interactivos, comprende aspectos relacionados con la parte física de la interac-

ción (colores, organización de los elementos en el espacio, etc.), el lenguaje (visual para las interfaces visuales, auditivo para las auditivas, etc.), los modelos de la información y la consistencia y coherencia. No nos referimos sólo a qué colores y qué tipografías hay que utilizar, sino a la disposición de los elementos interactivos en la interfaz de acuerdo con las tareas y en compromiso con los objetivos de usabilidad, y a la consecución funcional de estas tareas. En definitiva, el diálogo con el usuario.

La *affordance*, o comprensión intuitiva, será un aspecto importante en esta fase, que relacionará los factores humanos con la capacidad de los elementos de la interfaz de transmitir la sensación adecuada.

1. Objetivos

El diseño de un sistema interactivo es una tarea que además de ser compleja es determinante.

Determinante en el sentido en que supone la conexión de los requisitos con la implementación y, evidentemente, del resultado de esta conexión derivará la interfaz con la que el usuario interactuará. Y su complejidad, motivada en parte por dicha importancia, sólo puede ser resuelta con garantía de éxito si el equipo de desarrollo utiliza convenientemente compaginados los mecanismos y metodologías de diseño centrado en el usuario con los de la ingeniería del software, teniendo presente en todo momento la interdisciplinariedad de los componentes de dichos equipos.

Básicamente existen dos maneras de abordar el diseño de los sistemas interactivos:

- Una es la *aproximación empírica*. El diseño se basa en la propia experiencia del diseñador o bien en la de otros diseñadores que se recoge mediante compendios de recomendaciones (guías, reglas de oro, estándares, etc.) más o menos relevantes para la construcción de una interfaz con éxito. Estos resultados generalmente están avalados por unos estudios de evaluación realizados por el usuario (tests de usabilidad).

- Otra es la *aproximación metodológica*. Basada en unos fundamentos teóricos y en la aplicación de una serie de pasos para la realización del diseño. La aproximación metodológica posee bastantes aportaciones de otras disciplinas, sobre todo de las teorías cognitivas, ya que aportan mecanismos para la descripción del conocimiento que el usuario posee del sistema. Mientras que la aproximación empírica se basa en las aportaciones más relevantes (como por ejemplo reglas de diseño) de las aportaciones teóricas.

Desde nuestro punto de vista, dicha problemática se aborda a partir de la aproximación metodológica analizando las peculiaridades de este tipo de sistemas y los mecanismos existentes para su análisis y diseño. Para ello incidiremos en los siguientes aspectos:

- Ver y comprender aspectos vistos acerca del factor humano y qué implicaciones tiene en el diseño de interfaces de usuario.
- Conocer el proceso de diseño de sistemas interactivos.
- Realizar un diseño centrado en el usuario.
- Representación del modelo conceptual (MC).
 - Análisis de tareas.
 - Notaciones para el diálogo.
- Estrategias generales de diseño.

2. Relación con los factores humanos

Comprender la relación de los factores humanos con el diseño de las interfaces de usuario proporciona una serie de características que, de tenerlas presentes durante el diseño de dichas interfaces, mejorarán enormemente sus aspectos interactivos.

Dado que pretendemos obtener interfaces que reflejen el modelo mental de los usuarios y no de los programadores, para empezar estableceremos una relación entre distintos aspectos ya vistos.

2.1. Percepción y comprensión

2.1.1. Procesamiento visual

La implicación de la interpretación visual es que las imágenes son susceptibles de malas interpretaciones, ya que cada persona adjunta el conocimiento propio a lo que ve. La comprensión visual puede resumirse como “lo que uno ve depende de lo que uno conoce” (Sutcliffe yWatts, 2003).¹

Estas imágenes se almacenan en la memoria de trabajo y, a pesar de que su extracción es rápida, esta velocidad también varía de acuerdo con la complejidad de cada imagen y el conocimiento sobre su dominio.

Algunas de las implicaciones que el procesamiento visual tiene en el diseño de interfaces multimedia son:

- No esperemos a que los usuarios atiendan detalladamente a distintas áreas de una imagen al mismo tiempo.
- El efecto dominante de imágenes en movimiento se puede truncar si hay más de una al mismo tiempo.
- La información que un usuario comprenderá de una determinada imagen es muy difícil de predecir, por lo que proporcionar ayudas en forma de indicaciones (lingüísticas, otras imágenes, etc.) facilitará su correcta comprensión.

2.1.2. Modalidad auditiva: oído y habla

Durante el proceso comunicativo, el sentido auditivo comparte protagonismo con el visual; al igual que pasaba con la visión, lo que una persona oye no depende sólo del sonido que recibe, sino de su interpretación.

Desde el punto de vista humano, el aspecto más importante del sonido es el habla y el lenguaje. El sistema auditivo clasifica las entradas en tres categorías: *ruido* (sonidos sin importancia que pueden ser ignorados), *sonido significativo* (con sentido y significado) y *expresiones significantes* que componen el lenguaje.

1. Los contenidos de este apartado están basados en el documento siguiente: A. Sutcliffe y L. Watts (2003). “Multimedia Design for the Web”. *Interact 2003 Tutorial*. Zurich.

Este sistema, al igual que sucedía con la visión, hace uso de la experiencia pasada para interpretar las entradas.

Otra característica importante del sonido es que, al tratarse de un medio transitorio, a menos que éste se procese muy rápidamente, el mensaje se pierde. A pesar de esto, las personas disponemos de suficiente capacidad como para comprender efectivamente el lenguaje hablado e interpretar otros sonidos ágilmente. El mayor problema viene dado por las interferencias producidas por otros sonidos que compiten por igual con el mensaje principal (Gadiner y Christie, 1987).

Las implicaciones que pueden afectar al diseño de interfaces pueden ser resumidas en las siguientes:

- El sonido de “fondo” es interpretado como ruido si entorpece la comunicación.
- Los cambios de sonido producen un impacto ambiental, aspecto que puede aprovecharse, por ejemplo, para comunicar situaciones de alerta y advertencia.
- El sonido, como medio de difusión, puede molestar a las personas no afectadas por el mensaje.
- El discurso es interpretado según las reglas gramaticales y las referencias que cada uno posee en la memoria, que corrigen automáticamente muchas imperfecciones de la lengua hablada.

2.2. Atención selectiva

El proceso de atención es selectivo y está estrechamente relacionado con la percepción, al mismo tiempo que difiere mucho de un individuo a otro, dependiendo, entre otros, de factores cognitivos.²

2.2.1. La memoria de trabajo

El proceso de la atención es limitado en el momento en el que se quieren atender dos tareas mentales de manera concurrente (por ejemplo, leer un perió-

2. Un director de orquesta distingue los instrumentos individualmente; un oyente “normal” no puede.

dico al mismo tiempo que se escucha la radio) y por esta razón se realiza un proceso de selección.³

De esto se deduce que si tratamos de poner demasiados focos de atención en una determinada interfaz, se activan demasiadas demandas atencionales que compiten por la atención del usuario, lo cual conduce al estrés y a la fatiga, entorpeciendo la comunicación en lugar de favorecerla (Sutcliffe y Watts, 2003).

Cuando hablábamos de la memoria de trabajo, vimos que los contenidos que ésta recoge decaen rápidamente y continuamente se ven sobreimpresos por otros nuevos. Este factor tiene determinantes implicaciones que hay que recordar durante el proceso de diseño de interfaces.⁴

Las características de la memoria de trabajo (que vimos en el capítulo de los factores humanos) repercuten en el diseño en aspectos como los siguientes:

- Debemos intentar no sobrecargar la memoria de trabajo, tanto en términos de cantidad de información como en términos de su tiempo de exposición.
- Los medios de comunicación dinámicos rápidamente saturan la capacidad de la memoria de trabajo, y queda sólo la idea esencial de la información.
- La memoria de trabajo debe ser refrescada para que la información esté disponible cuando se necesite.
- Al mismo tiempo, la estructuración de esta información contribuye al proceso de memorización.

2.2.2. La memoria a largo plazo

La memoria a largo plazo (MLP) –o simplemente *memoria* en lenguaje coloquial– se caracteriza por los procesos de *reconocimiento* y de *recuperación* de la información que almacena.

La frecuencia de utilización de la información almacenada en la MLP, así como una construcción inadecuada del contenido inicial (distracciones durante

3. En el capítulo II vimos que las personas tenemos una capacidad limitada para poder procesar un número limitado de entradas simultáneamente.

4. Por ejemplo, el tiempo que una imagen debe permanecer en escena para extraer su información.

el proceso de memorización, atención diversificada, etc.) repercuten muy directamente en la activación de cada uno de estos dos procesos. Esta repercusión afecta al rendimiento de este almacén cognitivo.

La memoria es uno de los factores más críticos que limitan directamente el procesamiento de la información por parte de los humanos (Sutcliffe y Watts, 2003). Somos capaces de entender y memorizar información compleja, dividiendo la complejidad en componentes más simples mediante aproximaciones jerárquicas.

El principio de estructuración de la información⁵ se ve reforzado con el de la *consistencia*, el cual proporciona, por ejemplo, patrones de reconocimiento que facilitan el aprendizaje.

Todas estas características repercuten en el diseño de las interfaces en implicaciones como las siguientes:

- La eficiencia de la memoria (de largo plazo) correlaciona directamente con la profundidad de su tratamiento en su proceso de entrada.
- La distracción supone la mayor causa del olvido del material recientemente aprendido.
- Entradas similares para el almacenamiento de información perjudican el proceso de su recuperación.
- La recuperación resulta más fácil para una imagen acompañada de texto que para una imagen o un texto por separado.
- Encajar bien la temática del ítem que hay que recordar dentro de su contexto, así como agrupar y estructurar información de temática similar favorece enormemente su reconocimiento y posterior recuperación.
- Hacer razonar y preguntar por la comprensión son dos actividades que contribuyen al proceso de memorización.
- Todas las técnicas de estructuración de la información (palabras clave, acrónimos, memorización espacial, etc.) pueden utilizarse como ayudas para la recuperación de la información.
- La consistencia en la estructura y asociaciones crea mejores contextos para la memorización y su posterior recuperación.

5. Cuanto más estructurada esté la información, menor será el esfuerzo necesario para su aprendizaje y, por tanto, su reconocimiento y recuperación.

2.3. Motivación

La *motivación* es otro fenómeno que afecta directamente al rendimiento en la consecución de las tareas, en la toma de decisiones y en la atención.

A su vez, la *atención* se ve seriamente afectada por la dificultad de la tarea, la distracción en el entorno y la motivación individual.

El proceso de la motivación tiene una fase inicial de despertamiento (*arousal*),⁶ que es la encargada de estimular los sentidos del receptor para atender a determinados estímulos.

Las imágenes en movimiento suelen despertar mejor los estímulos que las estáticas, pero a su vez, una imagen de un paisaje con una puesta de sol tiene efectos relajantes y tiende a reducir el despertamiento.

La relación que podemos encontrar con el diseño de interfaces (multimedia) es:

a) En cuanto al contenido, las imágenes en movimiento favorecen la atracción e incrementan el despertamiento, pero un exceso de animación resulta molesto y distrae.

b) En lo que respecta a la estética, este campo es muy difícil, pues, por ejemplo, lo que resulta placentero para unas personas puede despertar justo lo contrario para otras.

c) En relación con el diseño, puede determinar el tipo de usuario al cual va dirigida la interfaz, intentando despertar la atención de sus destinatarios.

Y las implicaciones que estos aspectos pueden provocar en el diseño son:

a) Para despertar la motivación, será importante adaptar el contenido y la funcionalidad a las motivaciones de los usuarios.

b) Estas motivaciones también deben reflejarse en el estilo de la interfaz.

6. Las referencias que actualmente tenemos del despertamiento en la literatura (Reeves y Nass, 1996) explican que este proceso se encuentra todavía en fase de comprensión y aún es poco comprendido. Sin embargo, lo que sí se asegura es que de una u otra manera afecta a la memoria (se recuerdan mejor eventos después de un desenlace desagradable que si éste es placentero). Se conoce que existe interacción, compleja, entre el despertamiento, la motivación y la atracción.

3. Modelo mental y modelo conceptual

En el capítulo II hemos visto el concepto de modelo mental y la relación que existe entre éste y los diferentes tipos de memoria que forman parte del concepto global de la memoria de las personas.

3.1. Modelo mental

Durante el aprendizaje, una persona adquiere conocimientos de las relaciones estructurales y del funcionamiento del sistema con el que está interactuando. Se forma, por lo tanto, su propio modelo mental.

El modelo mental constituye, al contrario que el modelo conceptual, una abstracción del conocimiento interno que tiene el usuario del sistema (medida real de lo que el usuario conoce o piensa). A su vez, se trata de un modelo conceptual que el usuario piensa sobre el sistema.

Como sabemos, las características que definen los modelos mentales son que no son estáticos y que su representación es incompleta, ejecutable mentalmente, inestable, acientífica, no tiene unos límites claros e incluye supersticiones y creencias erróneas sobre la conducta del sistema. Además, es parsimoniosa.

Por consiguiente, el modelo mental no es ni completo, ni consistente, ni exacto. Todo lo contrario que el modelo conceptual (como veremos a continuación).

3.2. Modelo conceptual

“Un modelo conceptual correcto permite al usuario predecir los efectos de sus actos.

La falta de modelo conceptual supone actuar de memoria: el usuario no comprende la razón de por qué funcionan las cosas.”

D. A. Norman

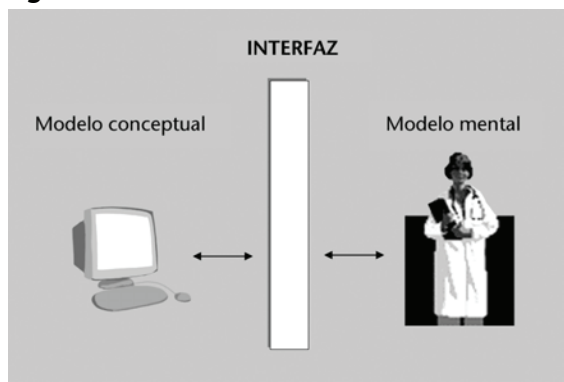
El modelo conceptual de un sistema de software constituye una abstracción externa que describe, con diagramas y notaciones (más o menos) formales, el conocimiento que debe tener una persona sobre un sistema.

El diálogo, que es el proceso de comunicación entre dos o más participantes, en el diseño de interfaces de usuario representa la estructura de la conversación entre el usuario y el ordenador.

Así pues, desde el punto de vista del usuario:

- 1) El modelo conceptual representa la información que cualquier usuario debería tener o adquirir sobre el sistema.
- 2) Está formado por un conjunto de elementos (conceptos) y relaciones entre los mismos observables desde el exterior.

Figura 7.1.



El diálogo entre el modelo conceptual del sistema y el modelo mental de los usuarios se realiza por medio de la interfaz

El modelo conceptual debe suministrar información al usuario sobre qué hace el sistema y los mecanismos para llevarlo a cabo. Su importancia radica en que debe favorecer el aprendizaje del sistema, es una guía para predecir el comportamiento del sistema y, además, el usuario utilizará este modelo para establecer estrategias encaminadas a resolver sus problemas. El modelo conceptual se tiene que basar, por lo tanto, en tres principios, que deben ser:

- 1) *Asimilable* (mediante el uso de conceptos familiares).
- 2) *Consistente* (coherente y bien formulado).
- 3) *Simple* (uso de descripciones comprensibles para un usuario medio).

No basta con diseñar el sistema para la efectividad (funcionalidad); el diseño de la interacción, además, debe ser comprensible.

Cuando las persona utilizan un ordenador para realizar una determinada tarea, usan cualquier conocimiento previo que les permita escoger y realizar las acciones apropiadas. La figura 1 contrasta el modelo del diseñador de una tienda en línea con el de un comprador. El diseñador del sistema debe, por tanto, aprovechar este conocimiento del comprador para mejorar su utilización.

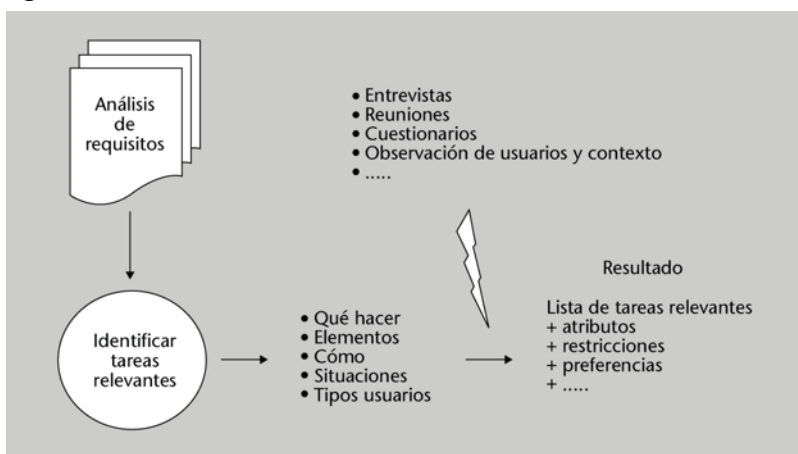
4. Actividades del modelo de proceso para el diseño

4.1. Análisis de tareas

Como fruto de la fase anterior –Análisis de Requisitos– el equipo de desarrollo dispone de una serie de requisitos funcionales que el sistema debe satisfacer y de una serie de tareas contextualizadas que los usuarios realizan. Ambas series influyen en la materialización de las mismas y deberán traducirse en líneas de código para que el sistema las realice.

El análisis de tareas supone el paso intermedio entre las descripciones de las tareas obtenidas y su codificación, estando orientado a *describir las interacciones usuario-sistema* de manera sistemática y eficiente.

Figura 7.2. Proceso de obtención de la lista de tareas a analizar.



Realizar esta actividad antes de la codificación, además de ayudar a concretar las secuencias necesarias para la consecución de cada una de las diferentes tareas, sirve como herramienta de soporte para un nuevo ciclo de análisis de requisitos puesto que a menudo hace aflorar detalles que no acabaron de quedar perfectamente explicitados.

Formalmente, el *análisis de tareas* se define como “el proceso de analizar la manera en la que las personas realizan sus trabajos: las cosas que hacen, las cosas sobre las cuales o con las cuales actúan y las cosas que necesitan conocer” (Dix y otros, 1993).

En todo proceso de análisis de tareas, hay dos fases bien diferenciadas:

- 1) *Obtención de la información* necesaria para comprender las actividades que hace el usuario (fase de análisis).
- 2) *Representación de la información* sobre un modelo adecuado (fase de modelado).

Los diferentes métodos⁷ existentes para realizar el análisis de tareas –que posteriormente veremos– parten de un *objetivo* (estado o logro que el usuario quiere alcanzar dentro de una aplicación) para cuya consecución realizaremos una serie de *tareas* (actividades necesarias para conseguir el objetivo, que pueden ser tanto actividades mentales como físicas) y de *acciones*⁸ (pasos a seguir para estructurar el orden y el cómo deben realizarse dichas tareas) asociadas a dichas tareas.

- La *utilidad* de esta actividad radica en aspectos como:
 - Ayuda para *comprender el dominio de la aplicación* (identificación de las actividades más importantes y sus relaciones).
 - Sirve para *fijar, organizar y describir* los datos previamente recogidos.
 - Facilita las *discusiones interdisciplinares* (el conocimiento de las tareas puede ser útil al diseñador, usuarios, analistas, psicólogos, sociólogos, etc.).

7. Los diferentes métodos para realizar el análisis de tareas se diferencian básicamente entre sí por el grado de formalismo de su notación y por el poder de expresividad y de finalidad.

8. Podemos considerar una acción como una tarea que no implica una solución de un problema o una estructura de control.

- El diseño de la nueva aplicación de forma *consistente con el actual modelo conceptual*, preservando las características más relevantes del funcionamiento lógico.
- El *análisis y evaluación de usabilidad* (se puede predecir el rendimiento humano e identificar problemas de uso).

4.1.1. Métodos

Los diferentes métodos que permiten llevar a cabo el análisis de tareas se agrupan en función de sus características bajo estas tres categorías:

1) Métodos cognitivos

- Identifican secuencias de comportamiento correctas.
- Representan el tipo de conocimiento que debe tener el usuario acerca del uso del sistema.
- Generan una especificación del usuario a partir de la descripción de tareas.

2) Métodos predictivos

- Evalúan el rendimiento humano.
- Describen secuencias de comportamiento.
- Hacen un análisis centrado en rutinas de comportamiento.

3) Métodos descriptivos

Los métodos descriptivos permiten obtener una descripción más o menos completa del sistema a partir de la información de las tareas.

Los métodos descriptivos más destacados de cada grupo son:

a) Cognitivos

- Análisis jerárquico de tareas (HTA)
- *Goals-operations-methods-selection* (GOMS)
- *User action notation* (UAN)

b) Predictivos

- *KeyStroke level mode* (KLM)
- *Task action grammar* (TAG)

c) Descriptivos

- *Concur task trees* (CTT)

A continuación se presentarán tres de los métodos más utilizados.

El análisis jerárquico de tareas

La técnica del análisis jerárquico de tareas (*hierarchical task analysis*), HTA, desarrollada por Annet y Duncan (Annet y Duncan, 1967), aunque es la más antigua, continúa siendo ampliamente usada.

En HTA se realiza una descripción de tareas en términos de operaciones y planes:

- 1) Las *operaciones* (descomposición en subtareas), actividades que realizan las personas para alcanzar un objetivo.
- 2) Los *planes*, una descripción de las condiciones que se tienen que dar cuando se hace cada una de las actividades.

Las operaciones se pueden descomponer de manera jerárquica y se asigna un plan a cada una de las subtareas que aparecen.

La representación puede ser en formato gráfico, en forma de árbol descendente o en formato tipo texto.

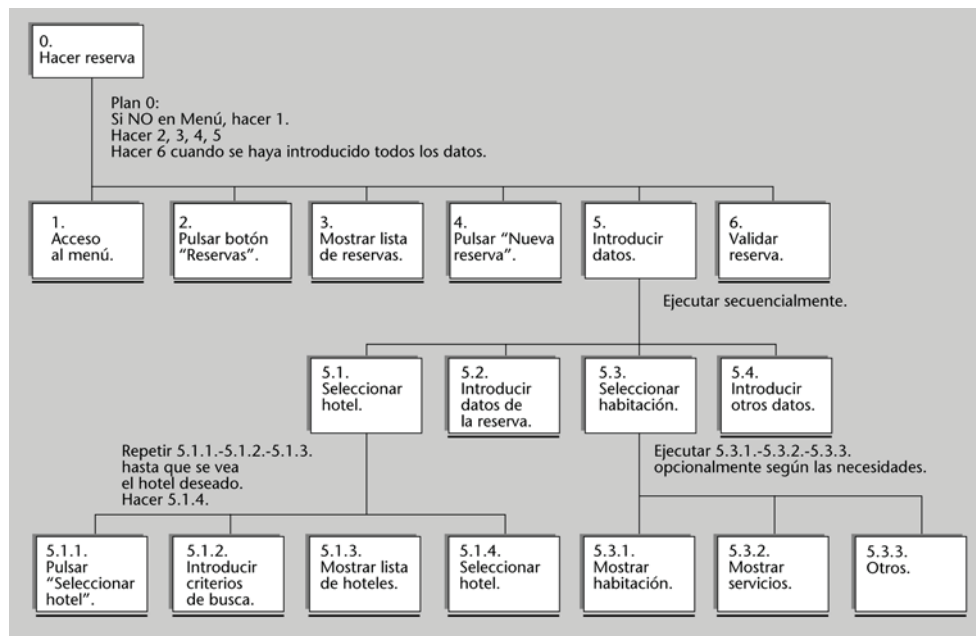
Metodológicamente, encontramos tres etapas en la realización del análisis.

1) *Etapla inicial*: definir la tarea principal, que puede ser dividida entre cuatro y ocho subtareas.

2) *Etapla intermedia*: decidir el nivel de detalle que se requiere y en qué punto acabar la descomposición.

3) *Parte final*: revisión y evaluación del trabajo hecho para comprobar su consistencia.

La figura 7.3 nos muestra la representación gráfica más utilizada correspondiente al análisis de una tarea específica realizada con la finalidad de rediseñar el espacio interactivo de una recepción de un hotel.

Figura 7.3. Un análisis de tareas con el método HTA

El método GOMS (*Goals - Operations - Methods - Selection*)

El método GOMS (Card y otros, 1983) comprende una familia de lenguajes que se basan en la visión del usuario como un sistema procesador de información (modelo de procesador humano).

El modelo GOMS se basa en el mecanismo de razonamiento humano para la resolución de problemas y formaliza aquellas actividades (físicas y mentales) que intervienen en esta tarea.

La metodología de este método es la siguiente:

Para cada tarea se describe

1) El *objetivo* o meta (*goal*) que se debe satisfacer.

- Las metas que se propone el usuario (lo que desea obtener).
- Los objetivos pueden servir como un punto de referencia en caso de un error.
- Un objetivo contiene información de la intención del usuario.
- Para esto, debe hacer una serie de operaciones básicas.

2) El *conjunto de operaciones (operations)* que el sistema pone a disposición del usuario para la interacción.

- Las operaciones son unidades elementales de percepción, motoras o actos cognitivos cuya ejecución es necesaria para cambiar algún aspecto del modelo mental del usuario, o bien para modificar el entorno.
- Este tipo de acciones puede afectar al sistema (pulsar una tecla) o bien sólo al estado mental del usuario (leer el cuadro de diálogo).
- Existe un grado de flexibilidad acerca de la granularidad de las operaciones (amplitud de cada operación)

3) Los *métodos disponibles* para llevar a cabo estas operaciones (*methods*).

- Para llevar a cabo estas operaciones, existen varias posibilidades de descomposición de una tarea en subtareas.
- Por ejemplo, en un gestor de ventanas se puede cerrar la ventana mediante ratón en un menú o teclado (atajo). Cada una de estas posibilidades será un método.

4) Y un *conjunto de reglas de selección (selection)* para determinar la alternativa más conveniente en cada caso (descritas mediante estructuras de control *if-then*).

- Cuando hay más de una alternativa, podemos indicar una serie de condiciones y reglas para tomar la mejor alternativa (método).

Cada tarea se podría descomponer en otras tareas primitivas formando un árbol jerárquico.

La notación GOMS puede servir también para medir rendimientos. La profundidad de subtareas se puede usar para estimar los requerimientos de la memoria de corto plazo (MCP) e, incluso, para estimar el tiempo de respuesta (asumiendo tiempos constantes para cada operación).

Descripción de una tarea con la notación GOMS

En este ejemplo tenemos una descripción de la tarea de mover una pieza en una partida de ajedrez.

Figura 7.4

```

GOAL: MOVER-PIEZA
  GOAL: DETERMINAR-TURNO
    [select GOAL: CONOCER-ÚLTIMO-MOVIMIENTO
      MOVER-A-LA-HISTORIA-DE-MOVIMIENTOS
      DETERMINAR-ÚLTIMO-MOVIMIENTO
      COMPROVAR-SI-NO-ERES-TÚ
      GOAL: CONOCER-MOVIMIENTO-SIGUIENTE
      MOVERSE-EN EL-TABLERO
      IDENTIFICAR-POSICIÓN-DEL-RELOJ
      COMPROBAR-SI-RELOJ-EN-TU-POSICIÓN]
  GOAL: ESCOGER-MEJOR-ESTRATEGIA
  GOAL: REALIZAR-MOVIMIENTO
    GOAL: SELECCIONAR-PIEZA-ADECUADA
      [select GOAL: IDENTIFICAR-PIEZA
        SELECCIONAR-TECLADO
        ESCRIBIR-IDENTIFICACIÓN-PIEZA
        CONFIRMAR
        GOAL: TOMAR-PIEZA
          MOVER-CURSOR-A-PIEZA
          PULSAR-BOTÓN-RATÓN]
    GOAL: ELEGIR-DESTINO
      [select GOAL: IDENTIFICAR-DESTINO
        MOVER-CURSOR-ARRASTRANDO-PIEZA
        ESCRIBIR-IDENTIFICACIÓN-POSICIÓN
        CONFIRMAR
        GOAL: DEJAR-IR-PIEZA
          MOVER-CURSOR-ARRASTRANDO-PIEZA
          DEJAR-IR-BOTÓN-RATÓN]
    GOAL: CONFIRMAR-MOVIMIENTO
      [select GOAL: TECLA-CONFIRMACIÓN
        PULSAR-ENTER
        GOAL: PARAR-RELOJ
          MOVER-CURSOR-RELOJ
          PULSAR-BOTÓN-RATÓN]

```

```
Selection Rule for GOAL: DETERMINAR-TURNO
    Si es una visualización gráfica, usar el método CONOCER-
        MOVIMIENTO-SIGUIENTE
    de otro modo, usar el CONOCER-ÚLTIMO-MOVIMIENTO
Selection Rule for GOAL: SELECCIONAR-PIEZA-APROPIADA
    Si no tiene ratón usar el método IDENTIFICAR-PIEZA,
    de otro modo, usar el método TOMAR-PIEZA
Selection Rule for GOAL: ELEGIR-DESTINO
    Si no tiene ratón usar el método IDENTIFICAR-DESTINO,
    de otro modo, usar DEJAR-IR-PIEZA
Selection Rule for GOAL: CONFIRMAR-MOVIMIENTO
    Si usas teclado usar el método TECLA-CONFIRMACIÓN,
    de otro modo, usar el método PARAR-RELOJ
```

El método CTT





El método *concur task trees*, o simplemente CTT, es un método propuesto y desarrollado por el grupo de investigación en IPO de la Universidad de Pisa (Italia) liderado por Fabio Paternó (2000).

CTT representa las relaciones temporales entre actividades y usuarios que son necesarias para llevar a cabo las tareas que hay que realizar.

Una de las principales ventajas de esta notación es su facilidad de uso, lo que hace que sea aplicable a proyectos reales con aplicaciones de un tamaño medio-largo y que suponen especificaciones de cierta complejidad. La notación genera una representación gráfica en forma de árbol de la descomposición jerárquica de las tareas que hay en el sistema.

En las figuras 7.5, 7.6, 7.7 y 7.8 vemos la notación gráfica correspondiente a los nodos del árbol.

Figura 7.5. Notación gráfica de los nodos en CTT

Notación gráfica de los nodos en CTT	
	Tareas del usuario. Tareas realizadas completamente por el usuario. Son tareas cognitivas o físicas que no interactúan con el sistema. Describen procesos hechos por el usuario usando la información que recibe del entorno (por ejemplo, seleccionar dentro de un conjunto de información la que se necesita en un instante determinado para la realización de otra tarea).
	Tareas de la aplicación. Tareas realizadas por la aplicación y actividades realizadas por la misma aplicación. Pueden obtener información interna del sistema o producir información hacia el usuario. Como ejemplo, podemos ver una tarea que presente los resultados obtenidos de una consulta en una base de datos.
	Tareas de interacción. Tareas que realiza el usuario interactuando con la aplicación por medio de alguna técnica de interacción. Un ejemplo puede ser seleccionar un elemento de una lista desplegable.
	Tareas abstractas. Tareas que requieren acciones complejas y que, por este motivo, no es fácil decidir dónde se harán exactamente. Son tareas que se descompondrán en un conjunto de nuevas subtareas.

El método dispone de una serie limitada (pero suficiente) para la notación entre los nodos del árbol que especifican la interacción con el usuario:

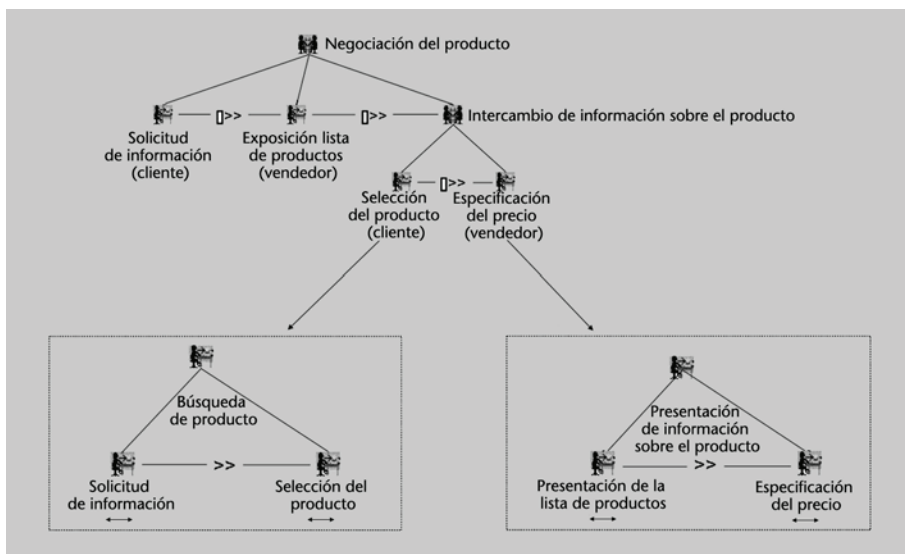
Figura 7.6. Notación de CTT para describir las relaciones que hay entre los diferentes nodos

Notación gráfica de los nodos en CTT	
T1 □ T2	Elección. Selección alternativa entre dos tareas. Una vez se está realizando una, la otra no está disponible al menos hasta que se acabe la que está activa.
T1 T2	Independencia de orden. Las acciones de las dos tareas se pueden hacer en cualquier orden.
T1 T2	Entrelazado (conurrencia independiente), interleaving. Las acciones de las dos tareas concurrentes se pueden hacer en cualquier orden.
T1 T2	Sincronización (conurrencia con intercambio de información). Las dos tareas deben sincronizarse en alguna de sus acciones para intercambiar información.
T1 >> T2	Activar (enabling). Cuando acaba la T1, se activa la T2. Las dos tareas se realizan de manera secuencial.
T1 □>> T2	Activar con paso de información. Cuando acaba T1, ésta genera algún valor que se pasa a T2 antes de ser activada.
T1 > T2	Desactivación. Cuando se da la primera acción de T2, la tarea T1 se desactiva.

Notación gráfica de los nodos en CTT	
T1*	Iteración infinita. La tarea T1 se hace de manera repetitiva. Se estará realizando hasta que otra la desactive.
[T1]	Opcional. No es obligatorio que se haga la tarea. Cuando describimos las subtareas que hay en la tarea de rellenar un formulario, algunas de las subtareas pueden ser opcionales (las de los campos que sean opcionales).
T1 > T2	Suspender/volver a iniciar. Da a T2 la posibilidad de interrumpir T1, y cuando T2 acaba se retoma T1 en el estado en el que fue suspendida.

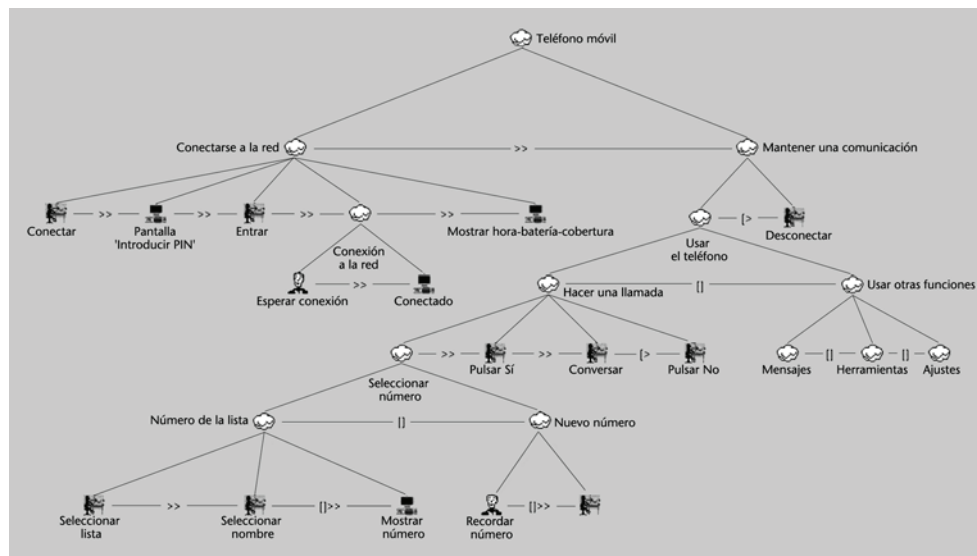
En las figuras 7.7 y 7.8 que presentamos a continuación podemos ver algunos ejemplos de este tipo de notación.

Figura 7.7. Ejemplo de una tarea de trabajo corporativo realizada con notación CTT



Otras características importantes de este método son:

- Está enfocado para representar entornos cooperativos (CSCW, *computer supported cooperative work*).
- Notación de los nodos: categorías de tareas.
- Se pueden definir atributos de objeto y de tarea.
- Dispone de una herramienta de libre distribución, CTT, que facilita su uso y posterior análisis.

Figura 7.8. Descripción del funcionamiento de un teléfono móvil con notación CTT

4.2. Modelos de diálogo como parte del modelo conceptual

Desarrollar un modelo conceptual supone previsualizar el producto propuesto, basándose en las necesidades del usuario y otros requisitos identificados (información extraída de la fase anterior del modelo de proceso).

Podemos distinguir dos grandes grupos de modelos conceptuales:

- 1) Los modelos conceptuales basados en las *actividades* que interactúan con sistemas. Los usuarios se suelen ver envueltos en tareas (instruyendo, conversando, manipulando –y navegando–, explorando y hojeando) que no son excluyentes entre sí.
- 2) Y los basados en los *objetos*, que tienden a ser más específicos que los anteriores, ya que se basan en objetos o artefactos (como una herramienta, un libro, un vehículo, etc.) utilizados en un determinado contexto.

El desarrollo de esta actividad se materializa con los modelos de diálogo, los cuales plasman la estructura del diálogo y describen la comunicación con cada participante. Los modelos de diálogo describen una parte del modelo conceptual del sistema. La descripción de estos modelos de diálogo se puede hacer mediante:

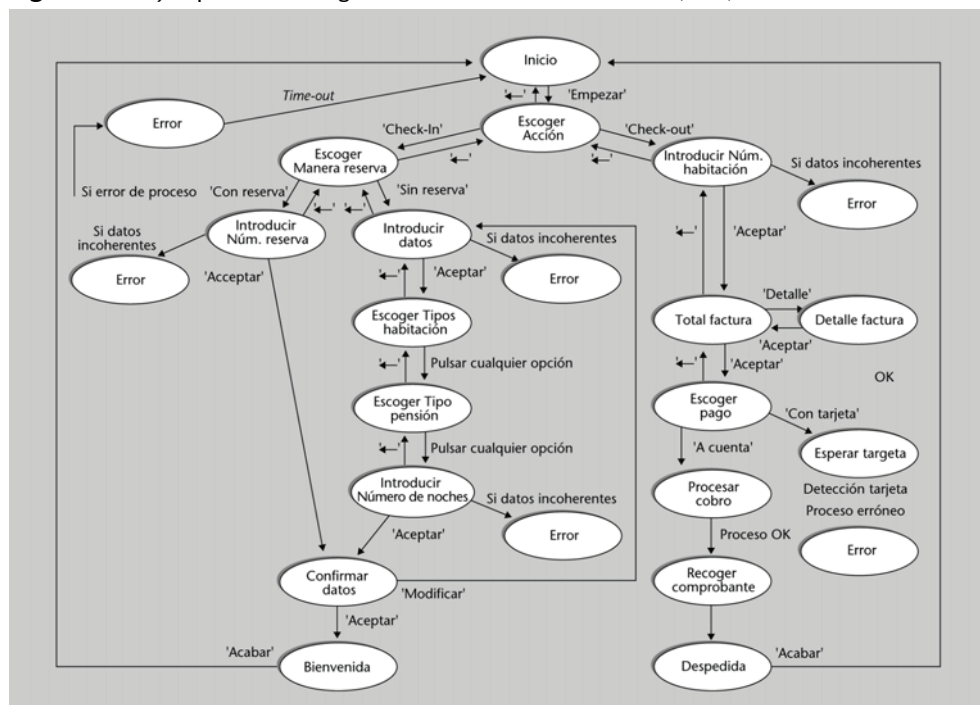
- Gramáticas. Descripción del diálogo como reglas de producción.

- #### 4.2.1. Diagramas de transición de estados

El diagrama de transiciones⁹ está formado por nodos y enlaces, donde los nodos representan los posibles estados del sistema, y los enlaces representan las posibles transiciones entre estados.

En los diagramas de transiciones podríamos, incluso, identificar al actor que ha provocado la acción. Las acciones del usuario serán aquellas transiciones realizadas directamente por la participación directa del usuario.

Figura 7.9. Ejemplo de un diagrama de transición de estados (STN)



9. Conocidos también como *state transition networks* o *STN*.

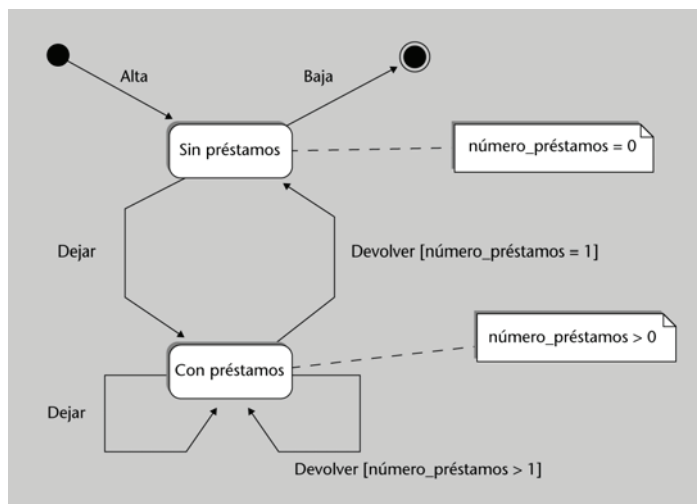
4.2.2. Diagramas de estados y de actividad UML

El lenguaje de modelización unificado o UML (*unified modeling language*) dispone de varios modelos y diagramas, cada uno de los cuales es completo desde su punto de vista del sistema.

Disponemos de dos diagramas de UML que proporcionan herramientas útiles para matizar el modelo conceptual del sistema:

1) El **diagrama de estados UML** (figura 7.10). Estrictamente hablando, un diagrama de estado en notación UML representa los estados por los que pasa un objeto a lo largo de su vida y que modelan el comportamiento de partes del sistema. Este comportamiento es modelado en términos del estado en el cual se encuentra el objeto, qué acciones se ejecutan en cada estado y cuál es el estado al que transita después de un determinado evento.

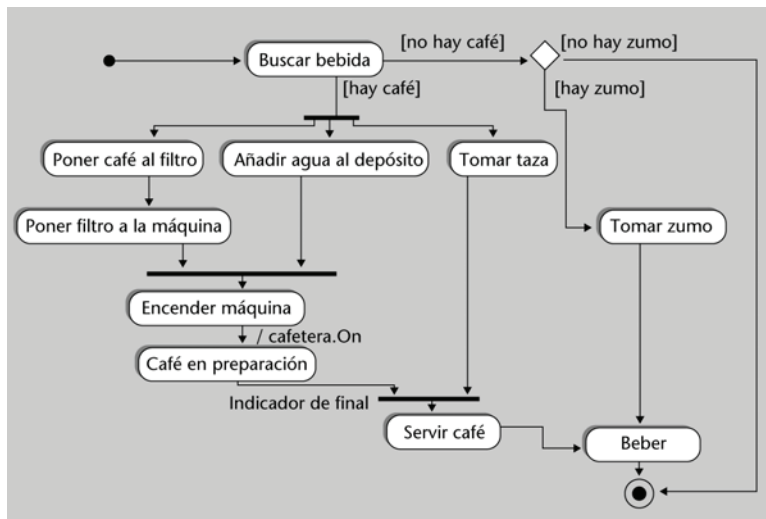
Figura 7.10. Un diagrama de estados representado en notación UML



2) El **diagrama de actividad UML** (figura 7.11). El diagrama de actividades puede especificar el comportamiento de los objetos de una clase, la lógica de una operación (método), una parte o toda la descripción de un caso de uso y la descripción del flujo de trabajo.

Según el detalle requerido, haremos uso de un diagrama u otro, o incluso de los dos.

Figura 7.11. Ejemplo de un diagrama de actividad en notación UML



4.3. Estilo. Estrategias de diseño de la información

El propósito de esta actividad es, como su nombre indica, definir un estilo que garantice la coherencia general de toda aplicación, entendiendo que la coherencia engloba todas las facetas de la interfaz (visual, sonora...), la funcionalidad completa del sistema y la interactividad con la que éste ofrece dicha funcionalidad al usuario mediante la interfaz.

Con relación al estilo, hay dos aspectos que habitualmente demuestran conflictos de comprensión que conviene aclarar:

- Suele confundirse el estilo de una interfaz con el diseño gráfico de la misma, lo que, a su vez, manifiesta dos errores significativos; el primero es que aunque las interfaces visuales son las más extendidas y frecuentes, el concepto de interfaz es, como hemos visto en el capítulo 3, mucho más amplio, y el segundo es que, aun tratando las interfaces visuales su diseño gráfico, constituye tan sólo una parte de dichas interfaces.

- No debemos caer en la frecuente tentación de confundir el diseño de la interfaz del usuario con la propia usabilidad: el estilo de la interfaz es tan sólo una de las características de su usabilidad.

Una vez definido el estilo, es aconsejable que se documente debidamente para que sirva de guía de estilo para toda la aplicación. Además, las aplicaciones evolucionan con el tiempo (nuevas funcionalidades, nuevas versiones, nuevas capacidades, etc.) y un factor habitual que suele ir totalmente en contra de la usabilidad es que dicha evolución va acompañada de un cambio en la forma de presentar las opciones a los usuarios los cuales, por tanto, se encuentran “des-pistados” por la propia interfaz. Disponer de una guía de estilo, seguirla y cumplirla garantiza que esto no se produzca.

4.3.1. Estándares generales

Desarrollar estándares, tanto si es la interfaz como cualquier otro aspecto del sistema, sirve para hacer los desarrollos de software más fáciles y seguros, establecer unos requisitos mínimos de fabricación y eliminar inconsistencias y variaciones innecesarias en las interfaces. Servirán de *guía general* durante el diseño de las interfaces de usuario de nuestra aplicación, de manera que nos aseguren un alto nivel de coherencia y consistencia –fundamentos básicos de la usabilidad. Pueden estar basados en estándares corporativos o en los datos recopilados durante la primera fase de análisis de requisitos y con análisis específicos que de esto se derivan (como por ejemplo, las metáforas que hay que utilizar).

Seguir estándares aporta una serie de ventajas, como por ejemplo que garantizan una terminología e identidad comunes, facilitan el mantenimiento y la evolución, reducen el proceso de formación y benefician la salud y la seguridad.

Hemos visto en otro módulo que tenemos dos tipos de estándares, los *de facto* y los *de iure*.

Seguir, por lo tanto, estándares generales en el diseño de cualquier sistema y proteger la uniformidad y la línea de productos desarrollados mejora la eficiencia del usuario (beneficio de una interfaz común para muchos programas). Los estándares aseguran la coherencia y la consistencia a lo largo del diseño de la interfaz, y en general de todo el sistema.

4.3.2. Estándares particulares

Este apartado recoge aspectos particulares del cliente final de la aplicación. Estos estándares también se conocen como *guías de estilo corporativas*.

Si una organización desea desarrollar su propio estilo corporativo, es recomendable que primero se escoja una guía de estilo comercial, así como ver qué elementos deben ser considerados para tener una imagen coherente.

La idea es usar las guías de estilo junto con características propias que den un estilo propio a la organización.

Acostumbran a tener más importancia cuando se trata de corporaciones en lugar de usuarios concretos, pues éstas disponen de elementos que les suele caracterizar –logotipo de la empresa, colores corporativos, simbología adaptada de otras aplicaciones existentes, fuentes tipográficas, argot propio, etc.

Este apartado incluye las metáforas y colores que provienen no de un trabajo de inspección de campo, sino del destinatario de la aplicación. Una misma aplicación podría servir para dos contextos diferentes cambiando ciertos elementos que, utilizados en un contexto u otro, podrían incluso significar funcionalidades totalmente opuestas.

4.3.3. Metáforas

El término *metáfora* normalmente va asociado al lenguaje y hace referencia a cuando se utiliza una palabra que expresa literalmente una cosa para manifestar otra que tenga cierto parecido con aquélla.

El mismo término, sacado del contexto literario, sirve para asociar conceptos abstractos de una manera más familiar y accesible.

Macintosh introdujo el concepto y el uso de las metáforas en el diseño de las interfaces de usuario y, por lo que se ha visto, tuvo un gran éxito. Las metáforas de la interfaz se basan en modelos conceptuales que combinan objetos y conocimientos familiares con nuevos conceptos. Papeles, carpetas, archivadores, buzones de correo y papeleras son representados en forma de iconos en la pantalla, los cuales poseen su propia funcionalidad y propiedades.

Se puede intuir, por lo tanto, la importancia que tiene definir correctamente las metáforas para el buen uso que los usuarios hagan de un sistema. Si esto se

ha hecho con acierto, el usuario intuirá su funcionalidad, no necesitará ayuda adicional y recordará su uso con facilidad.

Sin embargo, un error que los diseñadores suelen cometer durante esta fase es intentar diseñar una metáfora de interfaz para que se parezca y se comporte literalmente como la entidad física a la cual representa o con la cual es comparada. Incluso es importante tener en cuenta que una misma metáfora puede tener significados distintos, dependiendo del contexto en el que se encuentre (por ejemplo, la metáfora del buzón de voz que se debe usar en América del Norte es muy distinta de la que se tiene que usar en Europa).

Otro error común y por desgracia demasiado frecuente es el de decidir las metáforas sin contar al menos con un solo usuario. ¡Esta metáfora responderá al modelo mental del diseñador y no al del usuario!

4.3.4. Colores

Igual de importante es establecer los colores apropiados dependiendo del contexto y de los usuarios a los que va dirigido.

La buena selección de la combinación de colores contribuye a disponer de una interfaz de usuario agradable; también las funcionalidades principales y/o críticas serán mucho más accesibles dependiendo de los colores utilizados.

El estilo de los colores complementa las metáforas y les da incluso significados diferentes dependiendo del color utilizado (el primer ejemplo que viene a la mente es el uso de una luz como indicadora del estado de una posible opción, la cual en color rojo indica que no está accesible y en color verde, sí).

4.3.5. Principios de la Gestalt

En 1920, un grupo de psicólogos alemanes analizaron los principios de Gestalt de la organización perceptual. Estos principios, que describen las propiedades de configuración de la información visual, constituyen uno de los fundamentos de la psicología perceptual y tienen muchas aplicaciones en el diseño de interfaces que expresan o transmiten información.

Estos principios son los siguientes:

- 1) **Proximidad:** elementos visuales con propiedades comunes o cercanas se interpretan como agrupados.
- 2) **Similitud:** objetos que comparten características visuales (forma, color, etc.) se interpretan como grupo.
- 3) **Cierre:** elementos visuales que tienden a cerrar un área, que se interpreta como cerrada.
- 4) **Continuidad:** discriminación de elementos diferentes según la continuidad natural.
- 5) **Simetría:** hay la tendencia a ver elementos simétricos como partes de una misma figura.
- 6) **Área:** existe la tendencia a agrupar elementos de manera que se cree la figura más pequeña posible.

Los principios de la Gestalt no indican cómo deben representarse las unidades de información coexistentes en una interfaz, sino que hacen referencia a las consecuencias perceptuales acerca de su forma, composición y posición.

4.3.6. Organización de los elementos de la interfaz

La interfaz organiza la información en estructuras de varios niveles perceptuales. Los elementos básicos –píxeles, contornos y caracteres– se agrupan en estructuras como párrafos, iconos, listas de ficheros, barras de menús, etc.

Podemos organizar los elementos de la interfaz siguiendo unas reglas efectivas de diseño tales como las siguientes:

- *El balanceado:* busca el equilibrio entre los ejes horizontal y vertical.
- *La simetría:* duplica la imagen visual a lo largo de un eje de simetría.
- *La regularidad:* elementos ubicados regularmente entre filas y columnas.
- *El alineamiento:* elementos alineados entre sí transmiten una percepción más ordenada de la interfaz.
- *El enrejillado:* es una técnica que facilita los cuatro puntos anteriores.

4.4. Diseño detallado

Esta fase es el resultado de la evolución lógica de las fases anteriores cuando se han prototipado y evaluado como mínimo una vez –aunque el número de iteraciones necesarias para definir esta fase dependerá enormemente de la mayor o menor magnitud del proyecto.

Se procede a realizar un diseño de la interfaz que recoge todos los detalles aglutinados en tareas anteriores, y con el mayor detalle posible, hasta disponer de una versión que dispone de todos los mismos detalles que la versión de software definitiva.