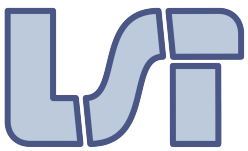
Fundamentos de Ingeniería del Software1

grado en ingeniería informática

Resumen del temario

Autor
Carlos Sánchez Páez



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2017-2018

Índice

1. Tema 1. Introducción a la Ingeniería del Software **3**
1.1. El producto Software . . . 3
1.1.1. Definición de Software . . . 3
1.1.2. Tipos de software . . . 3
1.1.3. Características principales . . . 3
1.1.4. Proceso de producción . . . 4
1.1.5. Problemas en el desarrollo . . . 6
1.2. Concepto de Ingeniería del Software . . . 7
1.2.1. Definiciones de la Ingeniería del Software . . . 7
1.2.2. Terminología usada en Ingeniería del Software . . . 7
1.3. Proceso de desarrollo del software . . . 9
1.3.1. Concepto de proceso de desarrollo . . . 9
1.3.2. Modelo general de proceso . . . 10
1.3.3. Tipos de modelos de proceso . . . 11
1.3.4. Proceso unificado . . . 14
1.3.5. Desarrollo Ágil . . . 16

2. Tema 2. Ingeniería de Requisitos **17**
2.1. Introducción . . . 17
2.1.1. Concepto de requisito y tipos . . . 18
2.1.2. Propiedades de los requisitos . . . 19
2.1.3. Tareas de la Ingeniería de Requisitos . . . 20
2.1.4. Roles . . . 21
2.1.5. Problemas de la Ingeniería de Requisitos . . . 22
2.2. Obtención de Requisitos . . . 22
2.2.1. Proceso de obtención de requisitos . . . 22
2.2.2. Técnicas de obtención . . . 24
2.2.3. Técnicas de entrevista . . . 24
2.2.4. Técnicas de análisis etnográfico . . . 25
2.3. Modelado de Casos de Uso . . . 25
2.3.1. Introducción . . . 25
2.3.2. Diagramas de Casos de Uso . . . 25
2.3.3. Actor . . . 25
2.3.4. Caso de Uso . . . 25
2.3.5. Descripción del actor . . . 25
2.3.6. Descripción del Caso de Uso . . . 25
2.3.7. Relaciones de los Casos de Uso . . . 25
2.3.8. Proceso de construcción del modelo de Casos de Uso . . . 25
2.3.9. Otros aspectos del modelo de Casos de Uso . . . 25

Índice de figuras

1. Etapas del proceso de producción del software. . . . 4
2. Esfuerzo invertido en las distintas etapas. . . . 5
3. Esfuerzo invertido en las distintas etapas del mantenimiento. . . . 5
4. Problemas en la comunicación. . . . 6

5. Impacto del cambio en las distintas etapas. 6
6. Sistema basado en computadora. 7
7. Sistema Software. 8
8. Modelo. 8
9. Estructura del proceso. 10
10. Tipos de flujos de proceso. 10
11. Modelo en cascada. 11
12. Modelo incremental. 12
13. Modelo de prototipos 13
14. Modelo en espiral de Boehm 14
15. Proceso unificado. 14
16. Ejemplo de Proceso Unificado 15
17. Resultados del informe *CHAOS*. 17
18. Tipos de requisito. 18
19. Actividades en el análisis de requisitos. 20
20. Revisión de requisitos. 21

1. Tema 1. Introducción a la Ingeniería del Software

1.1. El producto Software

1.1.1. Definición de Software

El software se puede definir de varias formas:

- Programa o conjunto de programas de cómputo que incluye datos, procedimientos y pautas que permiten realizar distintas tareas en un sistema informático.
- Transformador de información, para lo que adquiere, gestiona, modifica, produce o transmite esa información.

1.1.2. Tipos de software

Podemos clasificar el software en varios tipos:

1. Por *campo de aplicación*:
   a) Software de **sistemas**.
   b) Software de **aplicaciones**.
   c) Sotware de **programación**.
2. Por *tipo de licencia*:
   a) Según derechos de autor
      1) Software de **código abierto**.
      2) Software de **código cerrado**.
      3) Software de **dominio público**.
   b) Según su destinatario
      1) Usuario final (software **hecho a medida**).
      2) Para distribución (software **genérico**).

1.1.3. Características principales

1. El software es un producto lógico:
   a) No se fabrica, sino que se desarrolla.
   b) No se estropea, sino que se deteriora.
2. Crea modelos de la realidad.
3. Está formado por múltiples piezas que deben encajar perfectamente.

1.1.4. Proceso de producción

El proceso de producción del software está formado por las siguientes etapas:

1. **Definición**. Debemos precisar lo que queremos desarrollar. Para ello debemos realizar varias tareas:
   - Ingeniería de Sistemas.
   - Ingeniería de Requisitos.
   - Planificación de proyectos.
2. **Construcción**. Hemos de determinar cómo desarrollaremos el software. Depende de:
   - Diseño del software.
   - Generación del código.
   - Prueba del software.
3. **Evolución**. Consiste en precisar las partes del software que cambiarán.
   - Corrección.
   - Adaptación.
   - Mejora.
   - Prevención.

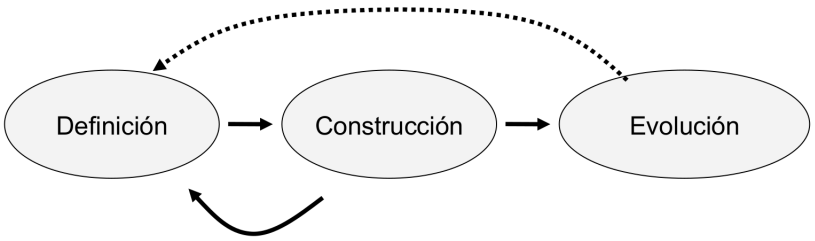


Figura 1: Etapas del proceso de producción del software.

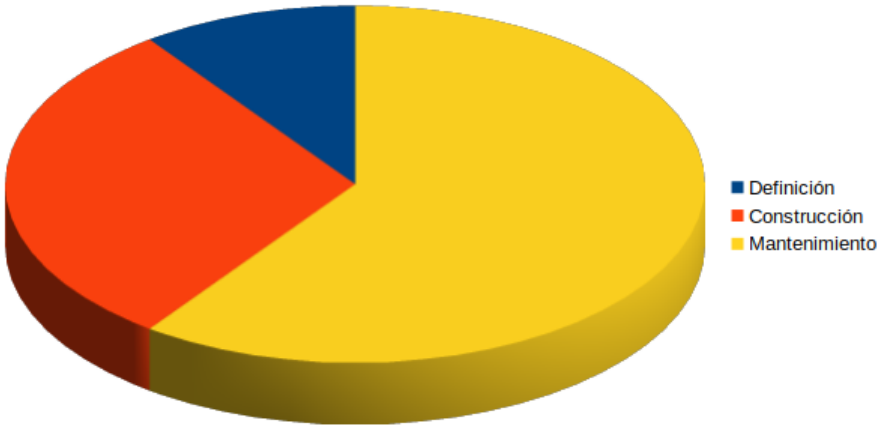


Figura 2: Esfuerzo invertido en las distintas etapas.

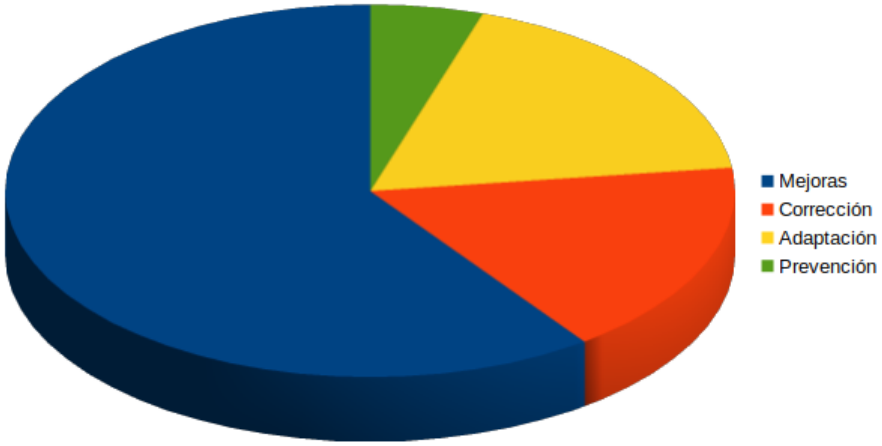


Figura 3: Esfuerzo invertido en las distintas etapas del mantenimiento.

1.1.5. Problemas en el desarrollo

1. **Comunicación** entre personas.

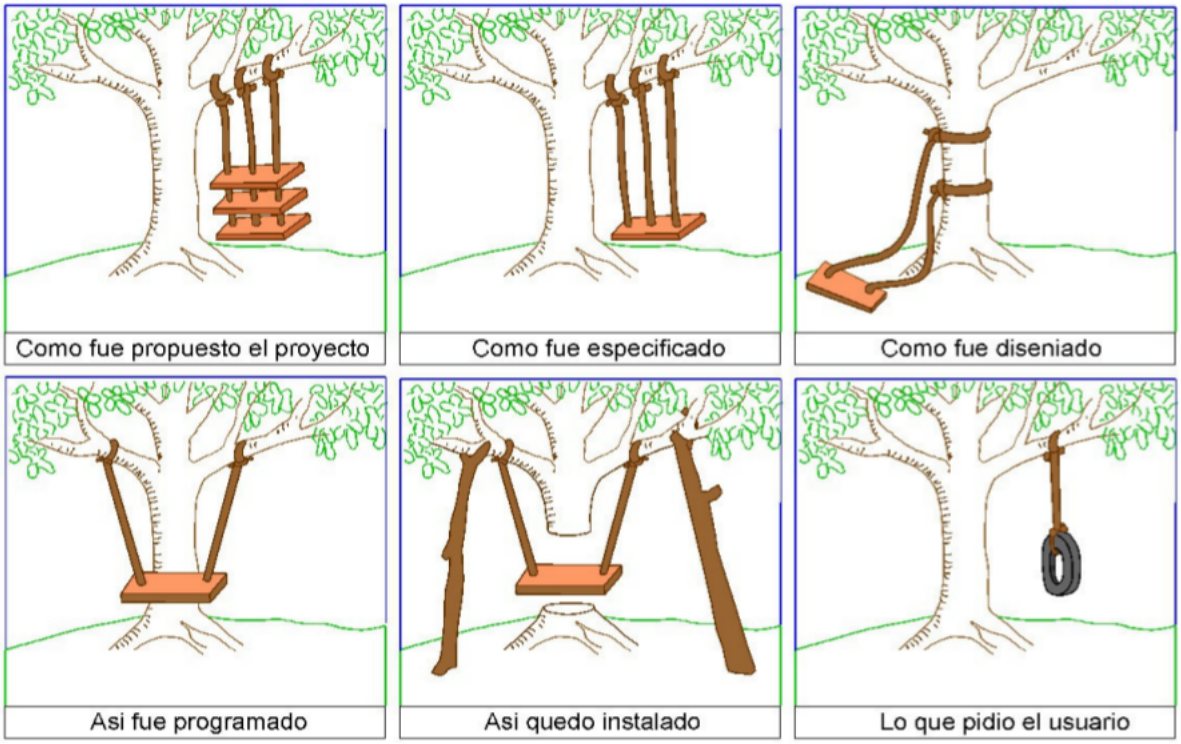


Figura 4: Problemas en la comunicación.

2. Incumplimiento de la **planificación**.
3. Incorporación de **cambios** en etapas avanzadas.

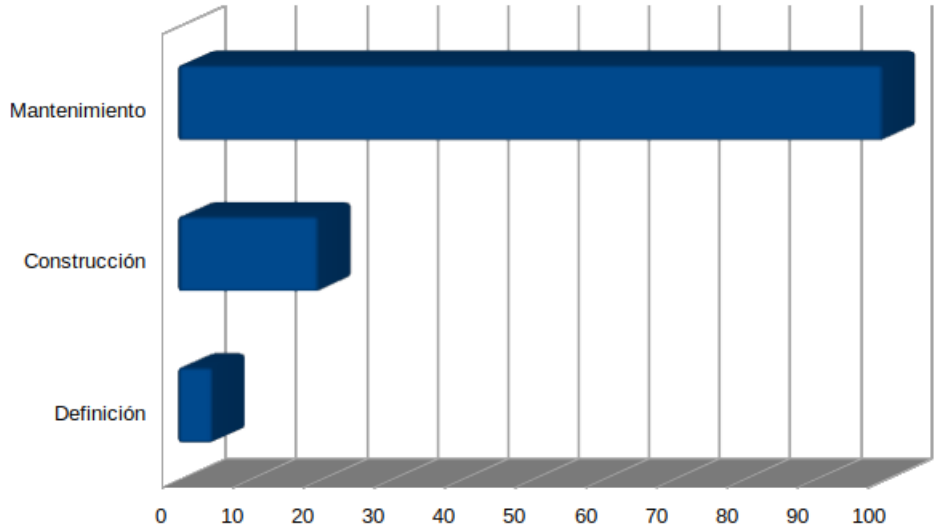


Figura 5: Impacto del cambio en las distintas etapas.

Desastres ocasionados por sistemas software La mayoría de ellos tienen como causa pruebas deficientes, mala documentación, diseños pobres o inexistentes, mal estudio del problema, etc.

1.2. Concepto de Ingeniería del Software

La *Ingeniería del Software* surgió por varias necesidades:

- Mal funcionamiento (calidad).
- Mantenimiento del software existente.
- Demanda creciente del nuevo software.
- Adaptación a las nuevas tecnologías.
- Incremento de la complejidad.

1.2.1. Definiciones de la Ingeniería del Software

1. Establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea fiable y trabahe en máquinas reales.
2. Aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada y requerida para el desarrollo, operación y mantenimiento del programa.
3. Estudio de los principios y métodos para el desarrollo y mantenimiento de sistemas software.
4. Aplicación de un enfoque sistémico, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, aplicación de la ingeniería al software.
5. Conjunto de teorías, métodos e instrumentos (tecnológicos y organizativos) que permitan construir sistemas software con las características de calidad deseadas.
6. Disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primerasa etapas de la especificación hasta el mantenimiento del sistema después de su puesta en operación.

1.2.2. Terminología usada en Ingeniería del Software

- **Sistema**. Conjunto de elementos relacionados entre sí y con el medio, que forman una unidad o un todo organizativo.
- **Sistema basado en computadora**. Conjunto o disposición de elementos organizados para cumplir una meta predefinida al procesar información.

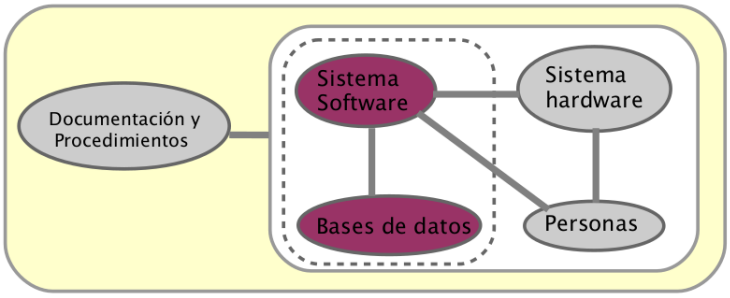


Figura 6: Sistema basado en computadora.

- **Sistema Software**. Conjunto de piezas o elementos software relacionados entre sí y organizados en subsistemas.

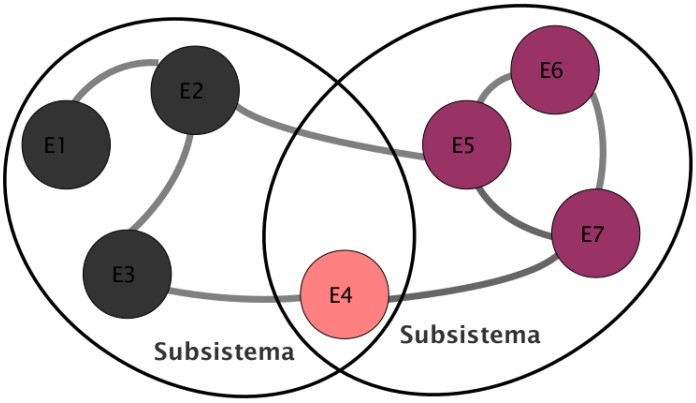


Figura 7: Sistema Software.

- **Modelo**. Representación de un problema en un determinado lenguaje. De un mismo problema se pueden construir muchos modelos.

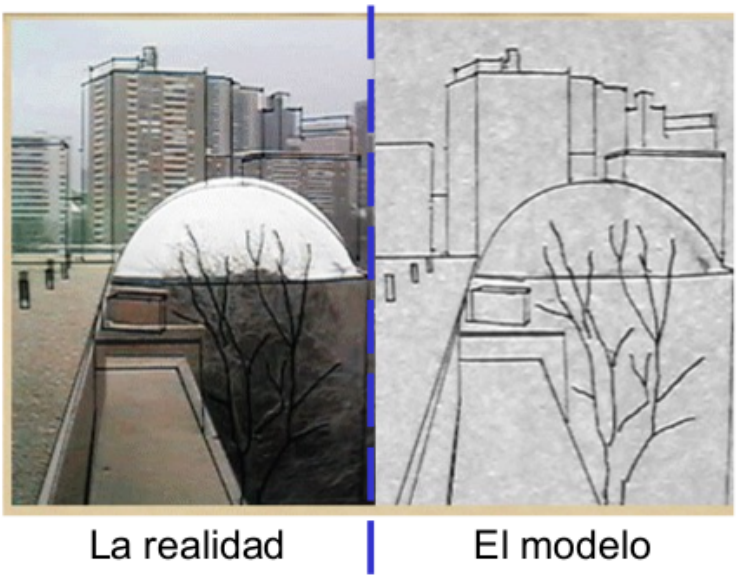


Figura 8: Modelo.

- **Principio**. Elementos que son adquiridos mediante el conocimiento. Determinan las características que debe poseer un modelo para ser una representación adecuada de un sistema.
- **Herramienta**. Instrumentos que permiten la representación de modelos.
- **Técnica**. Modo de utilización de las herramientas.
- **Heurísticas**. Conjunto de reglas empíricas que al ser aplicadas producen modelos que se adecuan a los principios. Ejemplo: *No usar materiales flexibles para representar la maqueta de un edificio.*

- **Proceso**. Estructura que debe establecerse para la obtención eficaz de un producto de Ingeniería.
- **Método**. Proporciona la experiencia técnica para elaborar el producto software. Se basa en principios fundamentales e incluye actividades de modelado.

1.3. Proceso de desarrollo del software

1.3.1. Concepto de proceso de desarrollo

- **Proceso de desarrollo del software**. Conjunto de *actividades, acciones y tareas* que se realizan cuando va a crearse un producto o sistema software.
- **Actividad**. Busca el logro de objetivos amplios e independientes del tipo de aplicación a desarrollar y su complejidad.
- **Acción**. Conjunto de tareas que elaboran un producto importante como resultado.
- **Tarea**. Objetivo pequeño y bien definido que produce un resultado tangible.

Las actividades que se realizan pueden ser de varios tipos:

1. **Estructurales**. Se dedican a obtener el producto:
   a) **Comunicación**. Colaboración con el cliente para entender los objetivos y requisitos del proyecto.
   b) **Planificación**. Definición del plan de proyecto en el que se describen los riesgos probables, recursos adquiridos y productos obtenidos a la vez que se programan las actividades, acciones y tareas.
   c) **Modelado**. Representación mediante modelos del sistema porpuesto junto con la solución o soluciones apropiadas.
   d) **Construcción**. Generación de código y su prueba.
   e) **Despliegue**. Entrega al consumidor y evaluación por parte de éste, lo que sirve como retroalimentación para el equipo de desarrollo.
2. **Sombrilla**. Se aplican a lo largo de todo el proceso. Se dedican a:
   a) Seguimiento y control del proyecto.
   b) Administración del riesgo.
   c) Aseguramiento de la calidad.
   d) Revisiones técnicas.
   e) Mediciones de parámetros del proceso.
   f) Administración de la configuración.
   g) Administración de la reutilización.
   h) Preparación y producción del producto de trabajo.

1.3.2. Modelo general de proceso

- **Estructura del proceso**. Cada una de las actividades, acciones y tareas se encuadran dentro de una estructura que define su relación con el proceso y entre ellas.

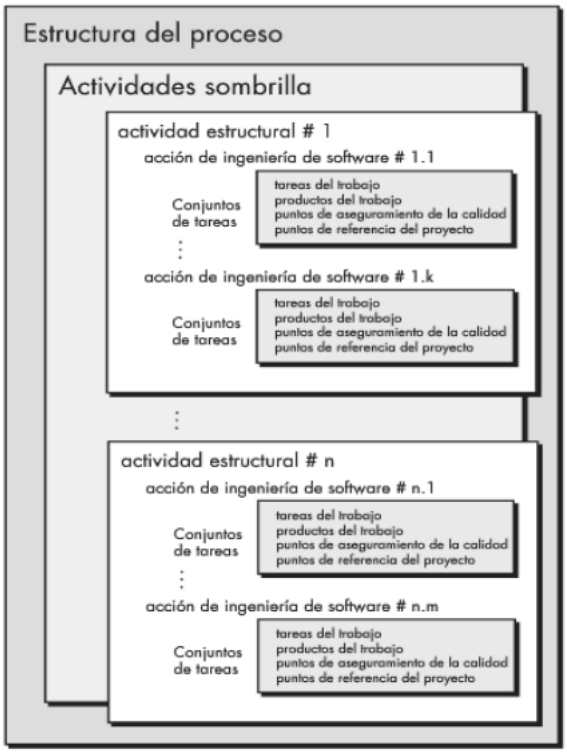
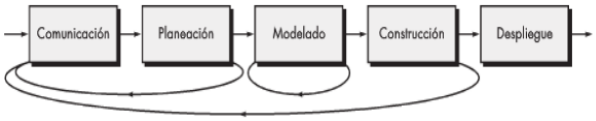


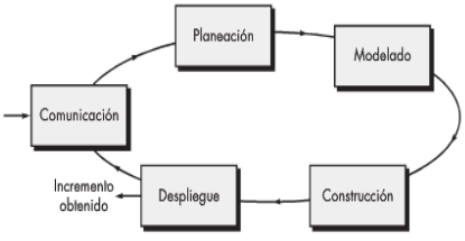
Figura 9: Estructura del proceso.

- **Flujo del proceso**. Describe la forma en la que se organizan las actividades estructurales, acciones y tareas en los procesos con respecto a la secuencia y el tiempo.

(a) Flujo de proceso lineal.



(b) Flujo de proceso iterativo.



(c) Flujo de proceso evolutivo.

Figura 10: Tipos de flujos de proceso.

- **Acciones y tareas de las actividades estructurales**.

  1. **Obtención de requisitos**. Obtención de información respecto a la acción que debe realizar el software.
  2. **Estimación y planificación del proyecto**. Estimar el tiempo y los costes del desarrollo del software.
  3. **Análisis de requisitos**. Documento en el que se especifica lo que debe hacer el sistema software.
  4. **Diseño**. Búsqueda de la solución. Descripción de los componentes, sus relaciones y funciones que le dan solución al problema.
  5. **Implementación**. Traducción del diseño a un lenguaje de programación entendible por una máquina.
  6. **Prueba del software**. Revisión y validación del código que se va desarrollando.
  7. **Evaluación y aceptación**. Evaluación del producto y aceptación por parte de los interesados en él.
  8. **Entrega y asistencia**. Sistema pasa a operar y se ofrece asistencia para su correcto funcionamiento.

1.3.3. Tipos de modelos de proceso

- **Modelo en cascada**. Presenta una estructura secuencial y un flujo lineal. Sin embargo, los proyectos no suelen adecuarse a este modelo, es difícil expresar los requisitos a través de él al principio del proyecto y ofrece poca comunicación con el cliente, ya que hasta el final no hay un ejecutable que se pueda evaluar.

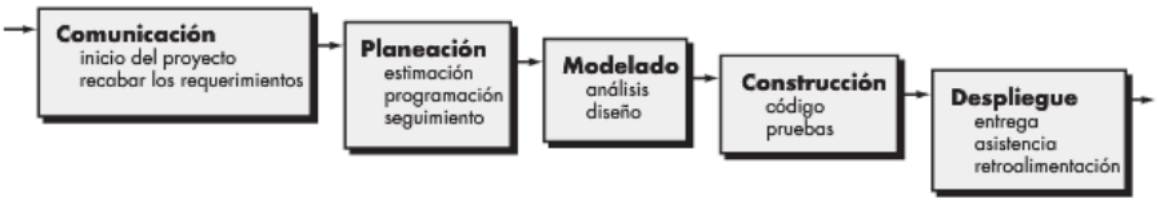


Figura 11: Modelo en cascada.

- **Modelo incremental**. Su estructura es secuencial mientras que el flujo de proceso es lineal y paralelo entre incrementos.

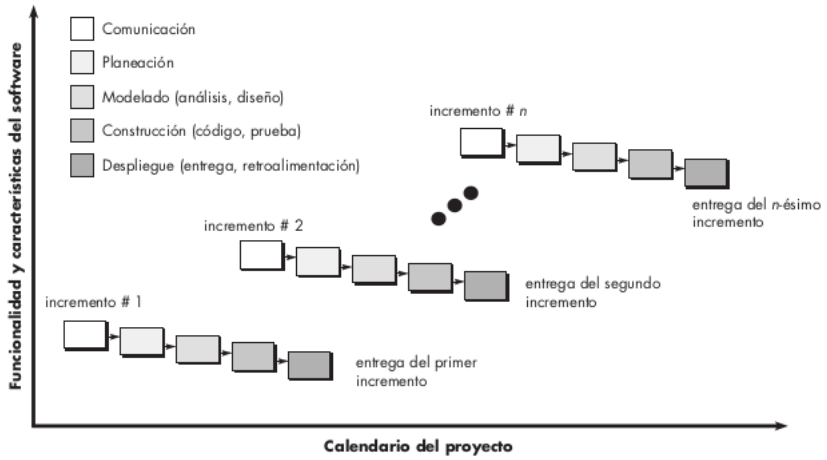


Figura 12: Modelo incremental.

- **Modelo evolutivo**. Es también iterativo. Nace como solución a varios factores, como un tiempo de entrega muy limitado o la necesidad de facilitar la incorporación de cambios. En cada iteración del proceso se obtiene un producto terminado y operativo. Sus características generales son:

  1. Afrontan los riesgos altos (técnicos, de requisitos, etc.) tan pronto como sea posible.
  2. Retroalimentación temprana por parte del cliente.
  3. Manejo de la complejidad (pasos cortos y sencillos).
  4. El conocimiento adquirido durante una iteración de la evolución se puede usar en el resto de iteraciones.
  5. Involucra continuamente al usuario (evaluación, retroalimentación, afinamiento y refinamiento de requisitos, etc.).

Hay dos tipos fundamentales de modelos evolutivos:

1. **Modelo de prototipos**. Un prototipo es una representación limitada de un producto que se utiliza para probar su diseño y comprender mejor el problema y sus posibles soluciones. Los prototipos pueden ser *evolutivos* (productos finales) o *desechables* (usados dentro de otros modelos de proceso).

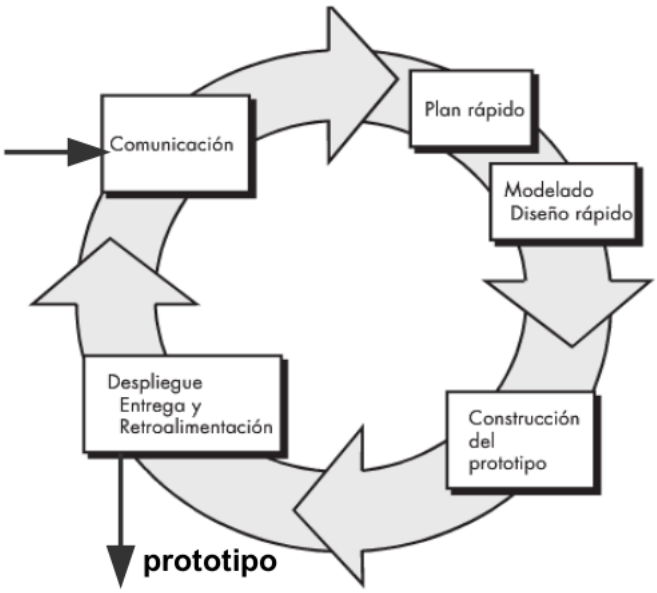


Figura 13: Modelo de prototipos

Este modelo se utiliza para:

1. Facilitar la obtención y validación de requisitos (*desechable*).
2. Estudios de viabilidad (*desechable*).
3. Propuestas de diseños alternativos (*desechable*).
4. En casos muy concretos como producto final (*evolutivo*).

Presentan todas las características de los modelos evolutivos, aunque se les añaden algunos inconvenientes:

- Crear falsas expectativas por parte del cliente (*desechable*).
- Puede que el prototipo *desechable* se elabore con una metodología ineficiente y ésta se mantenga en el producto final.

- **Modelo en espiral de Boehm**. Además de las características de los procesos iterativos, incluye otras más:

1. Se centra en el análisis de riesgo, construyendo prototipos para su estudio.
2. La espiral puede continuar una vez que se entregue el momento para llevar a cabo el mantenimiento.
3. Es adecuado para el desarrollo de sistemas a gran escala.

Sus principales inconvenientes son que no es controlable y que requiere un equipo de desarrollo con gran experiencia en análisis de riesgo.

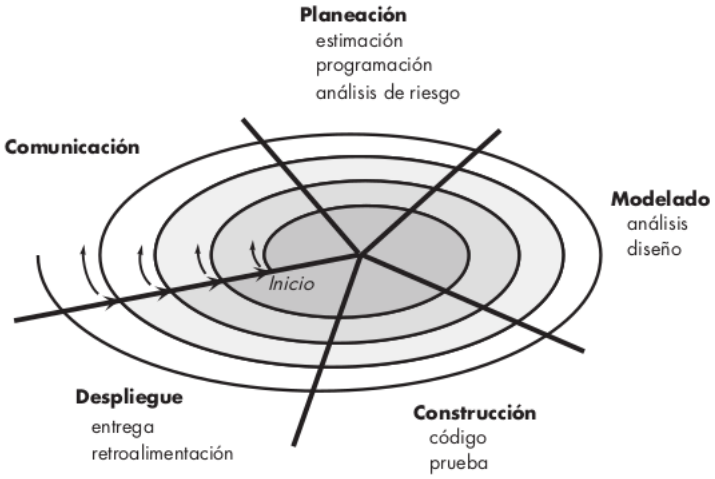


Figura 14: Modelo en espiral de Boehm

1.3.4. Proceso unificado

Es un modelo de proceso evolutivo y compuesto por cuatro fases:

- Inicio o concepción.
- Elaboración.
- Construcción.
- Transición.

Estas etapas se reparten entre las actividades estructurales como podemos ver en el diagrama:

Figura 15: Proceso unificado.

Además de las características de los modelos de proceso evolutivos, el proceso unificado incorpora las siguientes:

1. Es un modelo de proceso adaptable a la complejidad y al tipo de sistema.
2. Está centrado en la arquitectura, mostrando y decidiendo los distintos aspectos arquitectónicos de un sistema software en etapas tempranas. De esta forma pueden servir de base a las posteriores.
3. Está dirigido por casos de uso, desarrollándose uno o varios en cada iteración. En iteraciones tempranas los casos de uso determinarán la arquitectura.

Acciones y tareas en cada fase

- **Inicio**. Agrupa actividades tanto de comunicación del cliente como de planificación. Se propone una arquitectura aproximada para el sistema y se estudian numerosos factores (viabilidad, alcance, riesgos, etc.).
- **Elaboración**. Incluye actividades de comunicación y modelado de la arquitectura básica sobre la que se asentará la fase de construcción.
- **Construcción**. Se completan los modelos de requisitos y se implementan los elementos necesarios para completar el sistema. Según se van terminando los elementos, son probados e integrados al producto final. También se realizan pruebas de aceptación por parte del usuario.
- **Transición**. Consiste en asegurarse de que el sistema cumple con los requisitos especificados (mediante pruebas por parte de los usuarios). Además, se genera el material necesario para lanzar el producto al mercado.

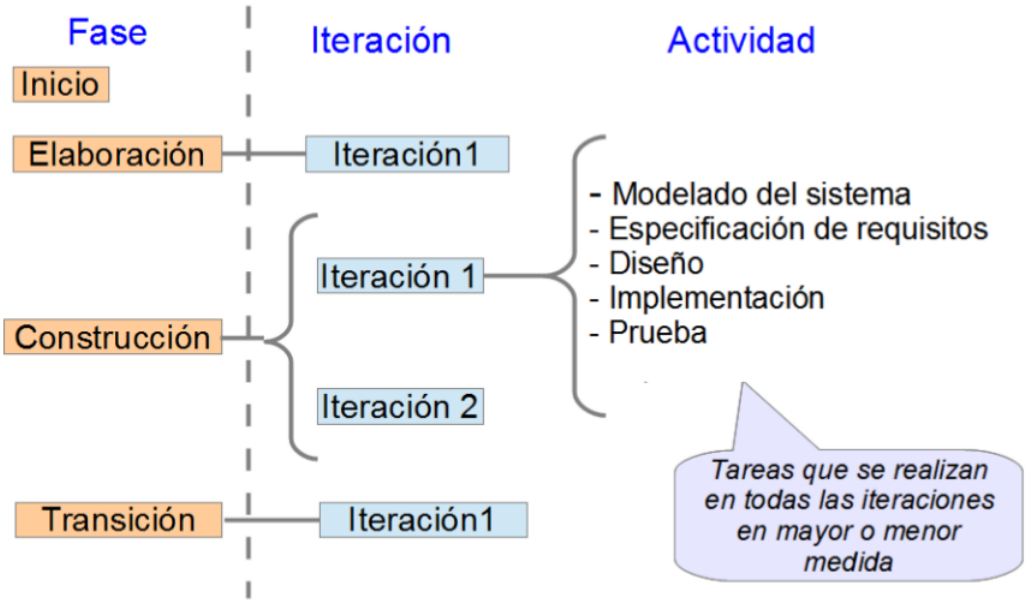


Figura 16: Ejemplo de Proceso Unificado

1.3.5. Desarrollo Ágil

En 2001, diecisiete expertos se preguntaron por qué muchos proyectos generaban menos valor del esperado, no se terminaban a tiempo, tenían problemas de calidad serios, etc. Tras ésto, elaboraron el *Manifiesto el Desarrollo Ágil de Software* con el que intentaban dar solución a estos problemas.

Características del Desarrollo Ágil

1. Es un proceso iterativo e incremental, por lo que es evolutivo.
2. Requiere entregas frecuentes y trabajo en equipo.
3. Establece autonomía en el equipo de desarrollo.
4. Exige revisiones y reuniones retrospectivas frecuentes.

Sus beneficios son que se mejora la productividad y que se manejan mejor los riesgos. Consta de varias técnicas:

- Scrum
- XP (*Extreme Programming*)
- Programación en parejas.
- TDD (*Test Driven Development*)

2. Tema 2. Ingeniería de Requisitos

2.1. Introducción

En 1995 se realizó el informe *CHAOS* sobre los resultados que se obtuvieron en diversos proyectos software:

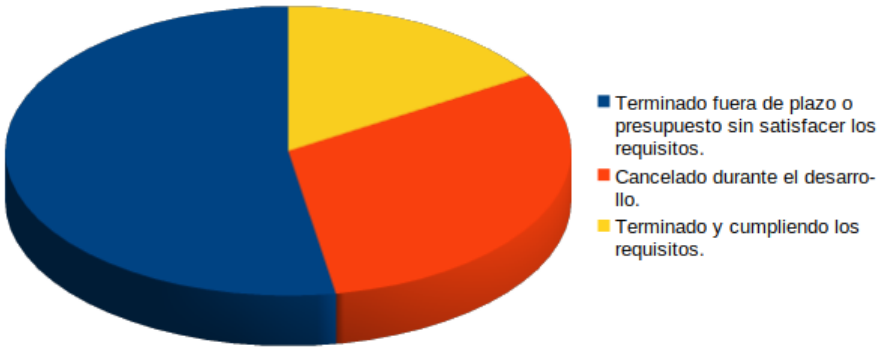


Figura 17: Resultados del informe *CHAOS*.

Los principales factores de fracaso son:

1. Falta de información por parte de los usuarios.
2. Especificación de requisitos incompleta.
3. Continuos cambios de los requisitos.
4. Pobres habilidades técnicas en la especificación de requisitos.

La **Ingeniería de Requisitos** cubre las tareas y proporciona los mecanismos adecuados para:

- Entender y analizar las necesidades del cliente.
- Evaluar la viabilidad de las necesidades.
- Negociar una solución razonable.
- Especificar la solución sin ambigüedades, confeccionando un documento que describa la solución acordada.
- Validar y analizar la especificación reflejada en el documento de especificación de requisitos, obteniendo el *modelo de análisis*.
- Administrar y desarrollar los requisitos a lo largo del proceso de desarrollo.

El proceso de construcción de una *especificación de requisitos* es iterativo. En él, partimos de las especificaciones iniciales imcompletas, poco claras y ambiguas y llegamos a especificaciones finales **completas, claras, documentadas y validadas**.

2.1.1. Concepto de requisito y tipos

Un requisito se puede definir de varias formas:

- Condición o capacidad que debe tener un producto software para resolver la necesidad expresada por un usuario.
- Representación en forma de documento de una capacidad o condición que debe tener un producto software.
- Característica de un productor software que es condición para su aceptación por parte del cliente.
- Propiedad o restricción determinada con precisión que un producto software debe satisfacer.

Tipos de requisito Los requisitos se pueden clasificar según varios factores:

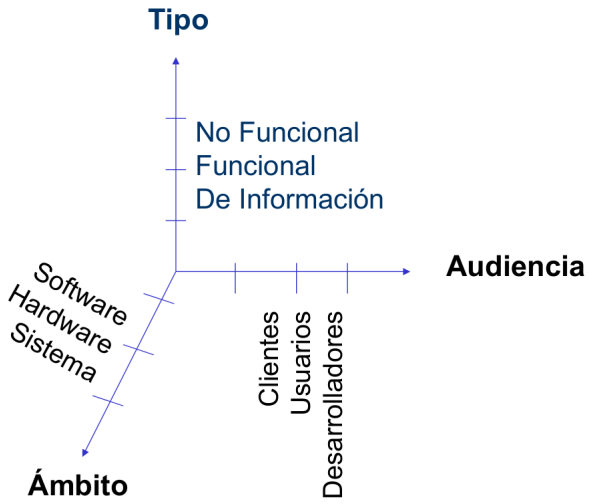


Figura 18: Tipos de requisito.

1. **Funcionales**. Describen la interacción entre el sistema y su entorno, proporcionando servicios que proveerá el sistema o indicando la forma en la que reaccionará ante determinados estímulos.
2. **No funcionales o atributo de calidad**. Describen cualidades o restricciones del sistema que no se relacionan de forma directa con el comportamiento funcional del mismo.
    - Restringen los tipos de soluciones que podemos tomar y suelen restringir el diseño que se realice.
    - No describen funciones, sino propiedades (rendimiento, fiabilidad, seguridad, etc.).
    - Son los que garantizan la calidad del software.
    - Pueden ser requisitos de producto, de organización o externos.
    - Son difíciles de determinar.

3. **De información**. Describen necesidades de almacenamiento de información en el sistema.

Clasificación FURPS+

- Funcionalidad (**F**uncionality): requisito funcional.
- Facilidad de uso (**U**sability): factores humanos, ayuda, documentación.
- Rendimiento (**P**erformance): tiempos de respuesta, productividad, etc.
- Soporte (**Supportability**): adaptabilidad, facilidad de mantenimiento, etc.
- Pseudorrequisitos o restricciones de diseño (+):
  - **Implementación**: limitación de recursos, lenguajes, etc.
  - **Interfaz**: restriccionesi mpuestas para la interacción con sistemas externos.
  - **Operación**: gestión del sistema en su puesta en marcha y a nivel operacional.
  - **Empaquetamiento**: formas de distribución, restricciones de instalación, etc.
  - **Legales**: licencias, derechos de autor, etc.

Ejemplos de requisitos

- El sistema debe validar la tarjeta en menos de tres segundos.
- El sistema debe contar el número de palabras procesadas.
- El sistema se diseñará para un terminal CRT monocromo.
- Los usuarios del sistema serán en su mayoría novatos.
- Deben producirse informes útiles

2.1.2. Propiedades de los requisitos

Para que los requisitos sean de calidad deben satisfacer las siguientes propiedades:

- **Completos**. Todos los aspectos del sistema deben estar representados en el modelo de requisitos.
- **Consistentes**. Los requisitos no deben contradecirse entre sí.
- **No ambiguos**. No se deben poder interpretar los requisitos de varias formas distintas.
- **Correctos**. Deben representar exactamente el sistema que el cliente necesita y que el desarrollador construirá.
- **Realistas**. Los requisitos se deben poder implementar con la tecnología y presupuesto disponible.
- **Verificables**. Se deben poder diseñar pruebas para demostrar que el sistema satisface los requisitos.
- **Trazables**. Cada requisito debe poder rastrearse a través del desarrollo del software hasta su conveniente funcionalidad del sistema.

2.1.3. Tareas de la Ingeniería de Requisitos

1. **Estudio de viabilidad**. ¿Es conveniente desarrollar el sistema?

   - ¿Soluciona el software los problemas existentes?
   - ¿Se puede desarrollar con la tecnología actual?
   - ¿Se puede desarrollar con las restricciones de costo y tiempo?
   - ¿Puede integrarse con otros existentes en la organización?

   Este paso finaliza con la obtención del *informe de viabilidad*.

2. **Obtención de requisitos**. Trabajo con clientes y usuarios para:

   - Estudiar el funcionamiento del sistema.
   - Descubrir las necesidades reales.
   - Consensuar los requisitos entre las distintas partes.

   Es un proceso difícil apoyado por diversas técnicas, como entrevistas, prototipados, casos de uso, etc.

3. **Análisis de requisitos**. Es la actividad más importante. Sus objetivos son:

   - Detectar conflictos entre requisitos.
   - Profundizar en el conocimiento del sistema.
   - Establecer las bases para el diseño.
   - Construir modelos abstractos.

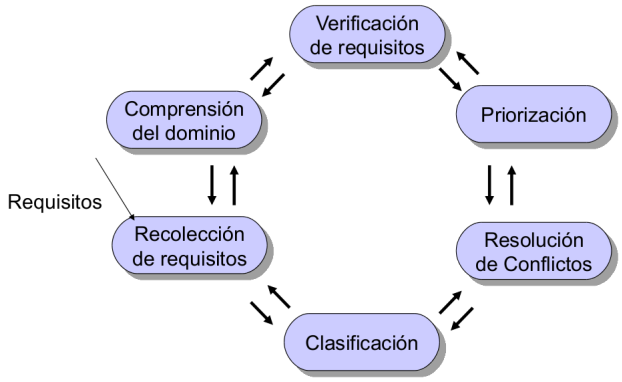


Figura 19: Actividades en el análisis de requisitos.

4. **Especificación de requisitos**. Consiste en representar los requisitos en base al modelo creado en el análisis.

5. **Revisión de requisitos**

   - **Validación**[1]. Consiste en ver que los requisitos reflejan el problema a solucionar.
   - **Verificación**[2]. Consiste en comprobar que la representación sea correcta.

   Es un proceso continuo durante todo el desarrollo.

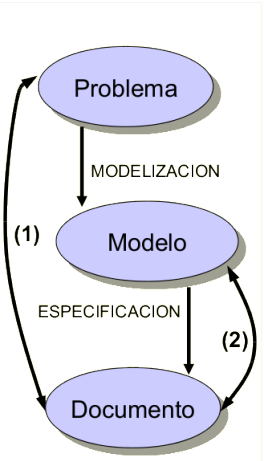


Figura 20: Revisión de requisitos.

En cada fase obtenemos una serie de productos:

- En la **obtención de requisitos**.
  1. Documentos de entrevistas.
  2. Lista estructurada de requisitos.
  3. Diagramas de casos de uso, plantillas de casos de uso y diagramas de actividad.
- En la **especificación de requisitos**.
  - Modelo arquitectónico (subsistemas) → Diagrama de paquetes.
  - Modelo estático (conceptual) → Diagrama de clases.
  - Modelo dinámico (funcional) → Diagrama de secuencia del sistema y contratos.

2.1.4. Roles

En la ingeniería de requisitos podemos distinguir varios roles:

- Stakeholder (personas que tienen relación con el sistema).
- Ingeniero de requisitos.
- Analista de sistemas.
- Arquitecto de software (diseño).
- Documentalista.

- Diseñador de Interfaces de Usuario.
- Gestor de proyecto,
- Revisor.

2.1.5. Problemas de la Ingeniería de Requisitos

Podemos agruparlos en 3 áreas:

- Dificultades para **obtener información**.
- Manejo de la **complejidad del problema**.
- Dificultades para la **integración de los cambios**.

Pueden estar causados por:

- Pobre comunicación
- Uso de técnicas inapropiadas.
- Tendencias a acortar el análisis.
- No considerar alternativas.

2.2. Obtención de Requisitos

2.2.1. Proceso de obtención de requisitos

La obtención de requisitos es la fase inicial de la ingeniería de requisitos. Necesitamos obtener:

- Necesidades y características del sistema.
- Informe del alcance del sistema o producto.
- Lista de participantes.
- Descripción del entorno técnico.
- Lista de los requisitos agrupados por su funcionalidad junto a las correspondientes restricciones que se aplicarán a cada uno.

Las tareas que se deben realizar son las siguientes:

1. Obtener información sobre el dominio del problema y el sistema actual.

   - Conocer el vocabulario propio.
   - Conocer las características principales del dominio.
   - Recopilar información sobre el dominio (consultas con expertos, libros, etc.)
   - Facilitar la comprensión de las necesidades del sistema.
   - Favorecer la confianza del cliente.

   Se ha de entregar la *introducción al sistema* y el *glosario de términos*.

2. Preparar las reuniones de elicitación y negociación.

   - Identificar a los implicados, realizando una descripción general de todos y del perfil de cada uno de ellos.
   - Conocer las necesidades de los clientes y usuarios.
   - Resolver los posibles conflictos

3. Identificar y revisar los objetivos del sistema. Si el sistema es muy complejo, podemos organizarlos mediante una jerarquía. De cada objetivo podemos describir:

   - Su **importancia** (vital, importante o "quedaría bien").
   - Su **urgencia** (inmediatamente, hay presión o puede esperar).
   - Su **estado** durante el desarrollo (en construcción, pendiente de solución, validado, etc.)
   - Su **estabilidad** (alta, media o baja).

4. Identificar y revisar los requisitos de información. De cada requisito podemos describir:

   - Objetivos y otros requisitos asociados.
   - Descripción del requisito.
   - Contenido.
   - Tiempo de vida (medio y máximo).
   - Ocurrencias simultáneas (medio y máximo).
   - Importancia, urgencia, etc.

5. Identificar y revisar los requisitos funcionales. Determinan lo que debe hacer el sistema. De cada requisito podemos describir:

   - Objetivos y requisitos asociados.
   - Secuencia de acciones.
   - Frecuencia.
   - Rendimiento.
   - Importancia, urgencia, etc.

6. Identificar y revisar los requisitos no funcionales. Son las restricciones aplicables a los requisitos funcionales y de información. De cada requisito podemos definir:
    - Descripción.
    - Objetivos y requisitos asociados.
    - Importancia, urgencia, etc.

Tras estos pasos se genera la *lista estructurada de requisitos*.

2.2.2. Técnicas de obtención

- Por **métodos tradicionales**: entrevistas, cuestionarios, análisis de protocolos, etc.
- Por **otros métodos**: técnicas orientadas a puntos de vista, escenarios y casos de uso, etc.

La información la poseen los implicados (*stakeholders*). Un implicado puede ser todo aquel que se beneficia del sistema a construir directa o indirectamente o bien que posea información sobre su funcionamiento o desarrollo, como los responsables del mismo, el cliente, los responsables de la gestión, etc.

2.2.3. Técnicas de entrevista

Tienen el objetivo de obtener información sobre el sistema mediante el dialogo con los expertos en el dominio del problema. Las entrevistas pueden ser de varios tipos: estructuradas o no estructuradas y formales o informales.
Las fases de una entrevista son:

- **Preparación**. Se estudia el dominio del problema, selecciona a los entrevistados y se planifican las entrevistas.
- **Realización**. Consta de:
    - Apertura: presentación e informe sobre los objetivos de la entrevista.
    - Desarrollo.
    - Terminación: recapitulación de la información obtenida.
- **Análisis**. Consiste en reorganizar la información, constatarla con otras fuentes , documentar la entrevista y enviar una copia al entrevistado.

Las principales limitaciones de una entrevista son que lo que los usuarios dicen no es siempre lo que hacen, la timidez y la interpretación de las preguntas. Sin embargo, aporta beneficios como la localización de las áreas en las que profundizar, la involucración de los clientes en el desarrollo o que es una técnica muy conocida y aceptada.

2.2.4. Técnicas de análisis etnográfico

Consiste en observar el contexto del sistema que afecta a los requisitos, es decir, observar la forma en la que las personas trabajan y no como el sistema las hace trabajar. Hay dos tipos de observaciones:

- **Directa**. El observador está inmerso en el sistema.
- **Indirecta**. Se utilizan entornos de observación.

Se utilizan fundamentalmente para dos tipos de requisitos:

- Los que derivan de la forma en la que trabajan realmente y no de cómo se han definido los procesos.
- Los que derivan de la cooperación y el conocimiento de las actividades de la gente.

No es un enfoque completo, sino que tiene que apoyarse en otras técnicas (entrevistas, prototipado, etc.)

2.3. Modelado de Casos de Uso

2.3.1. Introducción

2.3.2. Diagramas de Casos de Uso

2.3.3. Actor

2.3.4. Caso de Uso

2.3.5. Descripción del actor

2.3.6. Descripción del Caso de Uso

2.3.7. Relaciones de los Casos de Uso

2.3.8. Proceso de construcción del modelo de Casos de Uso

2.3.9. Otros aspectos del modelo de Casos de Uso