# Requerimientos y Análisis Inicial

¿*Wyatt Earp Syndrome*?

*" The "Wyatt Earp" is the indispensible programmer: the one who carries the critical program logic and design decisions around in his head, never*

*documenting anything he does..”*

Software engineering, Barry Boehm

Cuando se comienza el proceso de desarrollo de software, surge la pregunta de si es necesario invertir tiempo y esfuerzo en el diseño, o simplemente programar de manera directa. Sin embargo, según Barry Boehm, el diseño de un sistema de software permite descomponerlo en elementos, asignarles funciones y establecer relaciones entre ellos. Esta actividad es esencial para la producción de software porque permite reducir los costos de mantenimiento, que representan la mayor parte del esfuerzo total de producción. Además, evitará que cada cambio degrada la integridad conceptual del sistema y permitirá incorporarlos con el menor costo posible.

No obstante, existe un factor que puede inhibir el progreso de la práctica disciplinada de la ingeniería de software: el Síndrome de Wyatt Earp. Este síndrome se refiere a aquellos programadores indispensables que llevan la lógica crítica del programa y las decisiones de diseño en su cabeza, sin documentar nada de lo que hacen. Este hábito puede llevar a que la organización se vuelva aún más dependiente de estos programadores y que se les considere como héroes o modelos a seguir. Gerald Weinberg, un autor altamente respetado en la industria del software, sugiere que "If a programmer is indispensable, get rid of him as quickly as possible." (Weinberg, 1971, p. 100)

Por lo tanto, antes de comenzar a programar, es fundamental dedicar tiempo y esfuerzo al diseño de software. En este texto de Requerimientos y Análisis Inicial, se abordarán los aspectos fundamentales que deben ser considerados al inicio del proceso de desarrollo de software para garantizar una buena planificación y diseño del sistema. De esta manera, se podrá garantizar un proceso de producción de software más eficiente y económico, evitando la dependencia de programadores indispensables y permitiendo incorporar los cambios necesarios con el menor costo posible.

## Como obtener información

La obtención de información es un proceso fundamental en el desarrollo de software. La información obtenida en esta etapa es la base para la toma de decisiones y para el desarrollo del software. En este artículo se abordará el tema de cómo obtener información para el desarrollo de software y las fuentes de información que se pueden utilizar.

*Especificación inicial del cliente*

La especificación inicial del cliente es una fuente de información para el desarrollo de software. Sin embargo, esta especificación no siempre es precisa y puede tener omisiones. Es importante tener en cuenta que el cliente puede no saber exactamente lo que necesita y, por lo tanto, la especificación inicial puede ser insuficiente.

*Expertos en la materia*

Otra fuente de información para el desarrollo de software son los expertos en la materia. Estos expertos pueden proporcionar información sobre el problema a resolver y pueden ser de gran ayuda en el desarrollo del software. Los expertos en la materia no necesariamente tienen que saber de informática, pero tienen conocimiento sobre el negocio y pueden proporcionar información práctica.

*Usuarios*

Los usuarios son las personas que van a usar el software y pueden proporcionar información sobre el uso que se le dará al software. Es importante tener en cuenta que los usuarios pueden no tener conocimientos técnicos, pero tienen una idea clara de cómo quieren utilizar el software. Los usuarios pueden ser una fuente valiosa de información sobre la funcionalidad del software.

*Gerentes*

Para obtener información acerca de un sistema de software, es necesario considerar diversas fuentes de información. Una de ellas son los gerentes, quienes son los responsables del sistema y definen la importancia de cada parte del mismo. Además, pueden ayudarnos a investigar si hay mercado para productos similares, lo cual nos puede llevar a considerar la idea de hacer un framework o componentes reusables para proyectos similares.

*Departamento de Marketing*

Otra fuente de información son los departamentos de marketing, los cuales deben ser consultados en caso de que el proyecto sea interno. También es importante revisar proyectos previos, ya que pueden brindarnos información valiosa tanto positiva como negativa. En este caso, es necesario tomar en cuenta las experiencias positivas y evitar los errores del pasado.

En el proceso de desarrollo de software es fundamental obtener información para garantizar el éxito del proyecto. En este sentido, es importante considerar diversas fuentes de información, desde el cliente hasta los expertos en la materia, pasando por los usuarios, gerentes y el departamento de marketing. Cada uno de estos actores puede proporcionar información valiosa que permitirá entender las necesidades del software y garantizar su correcto funcionamiento. A continuación, se presentarán algunas preguntas que podrían hacer los diferentes actores mencionados, algunas de las cuales podrían representar requerimientos y otras no necesariamente lo serán.

Cliente:

¿Cuáles son los servicios que ofrece la póliza de seguro?

¿Cuál es la cobertura de la póliza?

¿Cuáles son los requisitos para acceder a la póliza?

¿Cómo es el proceso de renovación de la póliza?

Expertos en la materia:

¿Cuáles son las mejores opciones de cobertura según mi perfil?

¿Qué factores pueden afectar mi prima de seguro?

¿Cuál es el proceso de evaluación de riesgo para la póliza de seguro?

¿Cuál es el tiempo de respuesta ante un siniestro?

Usuarios:

¿Cómo puedo hacer efectiva mi póliza de seguro?

¿Cómo puedo hacer seguimiento a mi siniestro?

¿Cómo puedo reportar un siniestro?

Gerentes:

¿Cuál es la rentabilidad de la póliza de seguro?

¿Cómo podemos mejorar el proceso de venta de la póliza?

¿Qué coberturas son las más rentables? Marketing:

¿Qué canales de venta son más efectivos para la póliza de seguro?

¿Qué segmento de mercado tiene mayor potencial para la póliza?

¿Cómo podemos mejorar la imagen de la póliza de seguro en el mercado?

No todas las respuestas son requerimientos, por ejemplo, "¿Cuál es la rentabilidad de la póliza de seguro?" es una pregunta que no implica un requerimiento. La respuesta es importante para los gerentes para tomar decisiones, pero no es un requisito específico para el desarrollo del software. Por otro lado, preguntas como "¿Cuáles son los requisitos para acceder a la póliza?" o "¿Cómo puedo hacer efectiva mi póliza de seguro?" sí son requerimientos, ya que son aspectos específicos que deben ser considerados para el desarrollo del software. Es importante diferenciar entre preguntas que son necesarias para la toma de decisiones y aquellas que son necesarias para el desarrollo del software en sí.

## Suposiciones que debemos combatir

En el proceso de desarrollo de software, es común caer en suposiciones que pueden afectar negativamente el resultado final del proyecto. En particular, hay tres suposiciones que deben ser combatidas: la idea de que los usuarios son ingenuos y que los desarrolladores saben más, la creencia de que los requerimientos son estáticos y la suposición de que se puede construir una solución correcta en el primer intento. Para combatir estas suposiciones, es necesario identificar claramente los requerimientos de los usuarios, asegurarse de que el modelo se pueda adaptar a cambios en los requerimientos y también asegurarse de que el modelo pueda adaptar a inexactitudes en el entendimiento del dominio del problema. De esta manera, se pueden prevenir errores y mejorar la eficacia del proceso de desarrollo de software.



*Imagen disponible en https://pin.it/2pRytjm*

## Documento de Requerimientos

El Documento de Requerimientos es un documento esencial en el proceso de desarrollo de software que describe de manera clara y detallada los requerimientos funcionales y no funcionales del cliente y del sistema. Es importante que el cliente aporte su conocimiento y experiencia en el desarrollo del documento, lo que garantiza que se describan adecuadamente los requerimientos del proyecto. Además, el documento debe usar la terminología del negocio y evitar la jerga informática, para que cualquier persona involucrada en el proyecto pueda entenderlo.

El documento de requerimientos debe tener oraciones claras, completas y ser fácil de leer y entender, lo que facilita el análisis y diseño del proyecto. El alcance del proyecto y el contexto del problema deben ser claramente explicados y se deben incluir todas las restricciones necesarias.

Existen diferentes estándares para la creación del Documento de Requerimientos, incluyendo el IEEE 830, que establece un formato estándar para describir los requerimientos de software. Este estándar define los contenidos y la estructura del documento, y proporciona una guía detallada para la creación del mismo. Es importante seguir los estándares apropiados para garantizar que el documento sea completo y coherente con los estándares de la industria. En el curso directamente no se usará el estándar IEEE 830 para la creación del Documento de Requerimientos, pero si se consideran aspectos de él.

## Dominio del Problema

El dominio del problema es una sección crucial en el Documento de Requerimientos, ya que proporciona una comprensión clara y detallada del problema que se está abordando y las áreas que el nuevo sistema debe cubrir.

Por ejemplo, si estamos desarrollando un sistema de ventas en línea para una tienda de ropa, el dominio del problema podría incluir las áreas de la tienda que necesitan ser cubiertas por el sistema, como el catálogo de productos, el proceso de compra, la gestión de pagos, el seguimiento de envíos, entre otros. También podría incluir problemas específicos que el sistema debe resolver, como la necesidad de tener una interfaz amigable para el usuario, la necesidad de integrar el sistema con otros sistemas de la tienda, entre otros.

Es importante tener en cuenta que la vinculación del cliente con el proyecto no termina en esta sección del Documento de Requerimientos. Es fundamental mantener una comunicación constante con el cliente durante todo el proyecto para validar todas las áreas del dominio del problema y asegurarnos de que se están cumpliendo todos los requisitos.

## Clases y objetos candidatos

En la fase de análisis del desarrollo de software, una de las tareas importantes es identificar las clases y objetos que serán necesarios para implementar el sistema. Esta tarea se realiza a partir del documento de requerimientos, donde se subrayan los sustantivos y frases sustantivas que pueden ser candidatos a clases y objetos.

Luego de subrayar las palabras pertinentes, se debe hacer un análisis detallado de cada una, determinando si será una clase u objeto o si debe ser eliminada por alguna razón. Para esta tarea, se puede utilizar una planilla donde se registren los candidatos, la razón de eliminación y el nombre asignado en caso de ser una clase u objeto.

Por ejemplo, si estamos desarrollando un sistema de gestión de una biblioteca, podemos identificar palabras como "libro", "lector", "préstamo", "biblioteca", entre otras, como posibles candidatos a ser clases u objetos. Luego, se debe analizar cada una de ellas para determinar si son realmente necesarias y si es posible agruparlas en una clase.

A continuación, se presenta una tabla de ejemplo para ilustrar cómo se podría llevar a cabo el registro de los candidatos:

|  |  |  |
| --- | --- | --- |
| **Candidato** | **Razón de eliminación** | **Nombre de la clase u objeto** |
| Libro | Candidato válido | Libro |
| Lector | Candidato válido | Lector |
| Préstamo | Candidato válido | Préstamo |
| Biblioteca | Ya existe una clase para representar la biblioteca | - |
| Fecha de devolución | No es necesario un objeto especíﬁco para la fecha de devolución | - |
| Autor | No es necesario un objeto especíﬁco para el autor | - |

En este ejemplo, se identificaron cinco candidatos válidos y dos que no se consideraron necesarios. Además, se decidió no crear una clase adicional para la biblioteca, ya que se puede utilizar una existente para representarla. Es importante señalar que esta tarea requiere un análisis cuidadoso para evitar crear clases u objetos innecesarios o duplicados.

## Glosario

El glosario es un documento que describe todo el vocabulario utilizado en el proyecto. En el caso de la biblioteca, el glosario podría incluir nombres de productos como "sistema de préstamo", "sistema de catálogo", "sistema de devolución", etc. Además, el glosario podría contener definiciones de términos clave como "ISBN", "autor", "editorial", "prestamo", "multa", etc. También se pueden incluir sinónimos para términos que puedan ser confusos o que se utilicen de manera intercambiable. Por ejemplo, "usuario" y "cliente" pueden ser sinónimos en el contexto de la biblioteca. El glosario también puede incluir información sobre los tipos de datos y tamaños utilizados en el sistema, así como los dominios de valores válidos para los datos. En el caso de la biblioteca, esto podría incluir el tamaño máximo del nombre del autor o la longitud máxima del ISBN. Además, el glosario podría proporcionar información sobre el mapeo de tablas de bases de datos, transacciones legadas y etiquetas para la pantalla y los reportes.

# Tarea: "Lo que el analista olvidó"

**Objetivo**: Identificar errores u omisiones en un Documento de Requerimientos a partir de nuevas entrevistas, reescribir el ERS y extraer clases, objetos y casos de uso fundamentados.

### Escenario

Has sido incorporado al equipo de desarrollo de un sistema para una **librería online**. El analista anterior (que ya no está en el equipo) dejó un documento de especificación de requerimientos (ERS) que será tu punto de partida.

📄 **Recurso base**:

🔗 [ERS del sistema de librería online](https://rcasalla.gitbooks.io/libro-desarrollo-de-software/content/libro/temas/t_requerimientos/req_documentorequerimientos.html)

El gerente de proyecto te pidió que revises el ERS porque **hubo varios reclamos por parte de los usuarios del sistema** que no se sienten representados en el diseño. Por eso, realizarás entrevistas con los actores reales del sistema.

# Parte 1: Análisis inicial

1. Lee detenidamente el documento ERS.
2. Identifica los requerimientos actuales y anota los supuestos del analista original.

# Parte 2: Entrevistas reales con los actores del sistema

A continuación, encontrarás fragmentos de entrevistas realizadas con actores del sistema. Deberás leerlas como si fueran conversaciones **reales y recientes**, y detectar qué aspectos **no fueron tenidos en cuenta** en el documento original.

### 👤 Entrevista 1: Clara Benítez (lectora habitual)

“A veces busco novelas históricas y termino comprando algo de ciencia ficción. Me encantaría que el sistema me recomendara libros según mis compras anteriores.

Y también me gustaría dejar opiniones sobre los libros, como en Amazon.”

### 👤 Entrevista 2: Ricardo Gómez (encargado de atención al cliente)

“Nos llegan muchos correos preguntando si el libro está en stock. La web solo muestra si existe, pero no dice si hay disponibilidad. Y cuando alguien quiere pagar, no hay forma de guardar la dirección del cliente. Hay que escribirla de nuevo cada vez.”

### 👤 Entrevista 3: Mariana Costa (responsable de marketing)

“No tenemos datos sobre qué libros se buscan más. El buscador no registra nada. Sería muy útil tener estadísticas de búsqueda o de libros agregados al carrito, aunque no se compren. Eso ayudaría a nuestras campañas.”

### 👤 Entrevista 4: Tomás Viera (administrador del sistema)

“Yo puedo cargar libros nuevos, pero no puedo editar uno si cometí un error en el título o precio. Tampoco hay registro de ventas o estadísticas. Todo eso lo hacemos aparte en Excel.”

# Parte 3: Revisión crítica y modificación del ERS

📌 Actividad:

1. Anota todos los elementos **nuevos o contradictorios** que surgen en las entrevistas.
2. Modifica el ERS original agregando los nuevos requerimientos (como si fueras parte del equipo actual).
   * Mantené la estructura del documento original.
   * Usá lenguaje claro y alineado con los términos del negocio.
   * No borres nada del documento original, solo ampliá.

# Parte 4: Identificación de clases y objetos

1. A partir del ERS modificado, identifiquen candidatos a clases y objetos.
2. Completen una tabla como esta:

|  |  |  |
| --- | --- | --- |
| **Candidato** | **¿Clase u objeto?** | **Justificación técnica** |
| Recomendación | Clase | Se genera según historial del usuario |
| Opinión | Clase | Puede persistirse y ser mostrada públicamente |
| Dirección de envío | Objeto | Se usa como atributo en cada compra |

# Parte 5: Debate y fundamentación

📣 En Foro: "Lo que el analista olvidó"

* + ¿Qué elementos pasaron desapercibidos en el documento original?
  + ¿Qué supuestos peligrosos detectaron del analista anterior?
  + ¿Cómo afecta el **Síndrome de Wyatt Earp** esta situación?
  + ¿Qué buenas prácticas documentales surgieron de esta experiencia?

# Próximamente >> Casos de Uso

* *Introducción*
* *Identificación de Casos de Uso*
* *Relaciones entre Casos de Uso*
* *Diagrama de Casos de Uso*
* *Escenarios*
* *Desarrollo de Casos de Uso*
* *Clasificación de Casos de Uso*

## Bibliografía

Boehm, B. W. (1976). Software engineering: The future of systems development. IEEE Transactions on Computers, 25, 12, 1226-1241. Recuperado de

[https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=bb3752c1a38c071c430568b4cf3fee7f](https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=bb3752c1a38c071c430568b4cf3fee7f356d4033) [356d4033](https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=bb3752c1a38c071c430568b4cf3fee7f356d4033)

Weinberg, G. M. (1971). The Psychology Of Computer Programming. Recuperado de <https://djvu.online/file/j4zRDS1L7GQXm>