Universidad de Costa Rica

Informatica Empresarial, sede Guanacaste

Curso de Ingeniería de Software IF-7100

Exposiciones

Diseño de software

Autores: Rosales Mora Esteban B96967, Taisigüe Álvarez Cris B97785 & Vásquez Murillo Erick B98334

Profesor: Lic Iván Alonso Chavarría Cubero

Liberia, Costa Rica, mayo 2022

Rosales Mora Esteban, Taisigüe Álvarez Cris & Vásquez Murillo Erick

Diseño de software
Exposiciones, martes 11 de mayo
Profesor: Lic Iván Alonso Chavarría Cubero

Informatica Empresarial
Universidad de Costa Rica, sede Guancaste

Índice general

Índice de figuras

Puntos de evaluación para las exposiciones **Escala calificación de 1 a 10**	
Grupo #:	
Tema:	
Ítem a evaluar	**Nota**
1. Objetivos y metodología (relacionados con tema)	
2. Revisión bibliográfica (incluye trabajo escrito, empaste, orden, presentación)	
3. Preparación y organización del material en equipo	
4. Presentación y orden en el desarrollo del tema (incluye puntualidad, logística y apariencia personal)	
5. Utilización del tiempo de cada integrante del grupo	
6. Estímulo del interés por parte del auditorio	
7. Dominio del tema	
8. Evacuación de dudas propuestas por el auditorio	
9. Distribución equitativa del tema a exponer	
10. Aportes personales a la investigación y exposición	
Total puntos obtenidos:	

1. Introducción

Podría decirse que el diseño de software es el paso más importante y decisivo en el proceso de desarrollo de software, por lo que el producto resultante es uno de los más transparentes. Este paso generalmente implica el uso de diferentes técnicas y metodologías para producir un resultado lo suficientemente detallado como para que cualquiera que lo implemente pueda ejecutarlo físicamente o, en otras palabras, escribirlo en código. El diseño no solo se refiere a la interfaz gráfica del software, como suele equipararse con la palabra ”diseño”, sino que también se refiere al proceso específico mediante el cual se deben cumplir los requisitos del sistema generado.

El proceso de diseño de software es un concepto de alto nivel e independiente de la tecnología que describe un sistema que puede realizar las tareas identificadas en la fase de análisis de requisitos.

Según IEEE, el diseño se define como ”el proceso de definición de la arquitectura, los componentes, las interfaces y otras características de un sistema o componente, y el resultado de ese proceso”. El diseño de software proporciona el marco para la construcción de la arquitectura de software. Por lo tanto, es un paso importante. Precede a la creación o aplicación de un producto en todas las áreas de la tecnología. Por ejemplo, al construir un edificio, es impensable y poco práctico que los constructores lleguen al suelo y comiencen la construcción antes de desarrollar diseños detallados con los ingenieros.

El diseño es el primer y más importante paso en la metodología de desarrollo de software, ya que toda la aplicación se basará en las elecciones realizadas en esta etapa. Aquí, los diseñadores tendrán que elegir entre las soluciones propuestas y los recursos disponibles.

2. Objetivo General

Describir las diferentes fases que forma el diseño de software y sus distintas aplicaciones en el ámbito de explicación profesional en el mercado de TI.

2.1 Objetivos específicos

1. Evidenciar las características que contempla la etapa de entendimiento del proyecto, sus diferentes aplicaciones con las que cuenta.
2. Describir la etapa de investigación y la etapa de diseño de software que supone la recolección de datos e información que facilitan el desarrollo del mismo.
3. Explicar las etapas finales para el desarrollo de prototipo y evaluación para el análisis del conjunto de ideas que se representa por medio el prototipado.

3. Desarrollo

3.1 Etapa de Entendiendo los requerimientos del proyecto

Con el propósito de iniciar el ejercicio teórico que comprende la explicación de lo que abarca el diseño de software, es necesario conocer una de las etapas más tempranas en su estructuración; nos referimos a la etapa la cual se enfoca la labor de entender de forma certera los requerimientos de un proyecto. Esta etapa, así como las siguientes, son primordiales para determinar un buen diseño de software; mencionar que al tratarse de la primera fase su importancia es algo que se convierte en algo muy relevante, la razón de esto es que tiene repercusiones en las siguientes fases, por lo cual debe tratarse con seriedad y coherencia pues el proceso de diseño de software podría verse comprometido.

Para empezar con esta temática, comprendemos que para ofrecer o entregar una solución que se caracterice por poseer un alto valor de retorno para el usuario, debemos realizar un correcto y riguroso análisis de la problemática que presenta, por que al hacer esto logramos identificar las necesidades existentes para solventar, siempre considerando los diversos factores y variables que puedan de alguna manera influir. Al tener las necesidades que surgen a raíz de la problemática bien presentes, los requerimientos del proyecto son mas certeros en sus objetivos.

En uno de sus artículos el autor Akash Taykar nos describe la importancia de entender los requerimientos del proyecto, Akash expresa lo siguiente: “make sure that you know all the project requirements. These requirements can comprise your users’ pain points, their difficulties, and what issues you need to address while creating a software design specifically” [asegúrese de conocer todos los requisitos del proyecto. Estos requisitos pueden comprender los puntos débiles de sus usuarios, sus dificultades y los problemas que debe abordar al crear un diseño de software específicamente]. (Takyar, 2022).

Es necesario que al momento de realizar o llevar a cabo el proceso de diseño de software, tengamos una clara noción de los términos requerimiento y especificación, de esta manera podremos determinar las diferencias que existen entre ambos y podremos conocer que papel desempeñan en esta primera etapa de diseño de software.

Requerimientos & Especificaciones

Con la intención de realizar un mayor énfasis a la etapa compresión de los requerimientos dentro del diseño de software, manifestamos que en lo que respecta el desarrollo de software suelen manejarse terminologías, las cuales pueden poseer una cierta similitud conceptual, más, sin embargo, pertenecen a etapas o niveles diferentes. Dentro de la sintaxis y jerga de la ingeniería y desarrollo de software, existen dos términos, los cuales son requerimiento y especificación, dichos se relacionan con la primera etapa del diseño de software, por lo cual determinaremos a continuación qué significado tiene cada uno.

- **Requerimientos**: Los requerimientos pertenecen a etapas mas tempranas dentro del ciclo de desarrollo de un software, para la ingeniería y profesora de Aeronáutica y Astronáutica en MIT Nancy Leveson, los requerimientos dentro del desarrollo de software definen que funcionalidades podrá hacer un sistema, mas no describe el cómo se hará para ejecutar dichas, también menciona que los requerimientos son desde una perspectiva, una clase de contrato entre el proveedor del sistema y el equipo de desarrollo, por lo cual debe ser comprensible para ambos. (Leveson, s.f.).
- **Especificación de Diseño**: Las especificaciones son muy fundamentales en la etapa de diseño de software, pues como menciona la ingeniería y profesora de Aeronáutica y Astronáutica en MIT Nancy Leveson, las especificaciones son unas descripciones abstractas de las distintas conformantes de un software, generalmente van mayormente destinadas a diseñadores de software, también describe como se cumplirán los requerimientos. (Leveson, s.f.).

Abstracción

La abstracción es un proceso de síntesis de la información, en el caso del diseño de software y la etapa de entender los requerimientos sintetiza y abstrae requerimientos del sistema y brinda una representación que sea altamente estandarizable y en la medida de lo posible comprensible para los involucrados en el desarrollo del software.

El autor corporativo Icy Science, en uno de sus artículos nos indica que la abstracción es:

> La abstracción es el acto de representar características esenciales sin incluir los detalles de fondo o las explicaciones. En el dominio de la informática y la ingeniería de software, el principio de abstracción se utiliza para reducir la complejidad y permitir el diseño e implementación eficiente de sistemas de software complejos. (Icy Science, 1970).

Características

Las abstracciones dentro del diseño de software son de suma importancia, pues son la simplicidad de la información, que nos permiten tener objetivos y metas claras en el proceso de desarrollo de software. Por lo anterior, es indispensable que consideremos que características representan un buen proceso de abstracción.

- **Carácter abstracto** Lo que significa esta característica, es que en el proceso de abstracción debemos tomar en cuenta la información explicita que una abstracción pueda contener, las palabras mas abstractas tienen menos información por lo que pueden representar una estandarización. (Wagner & Deissenboeck, 2008, p.3).

  Esta característica puede ser ejemplificada mediante dos objetos. Establezcamos la situación en la que abstraemos dos objetos, el primero sera un ”Mueble” y el segundo sera una ”Silla”, la palabra ”Mueble” implica menos información explicita, mientras que contrariamente, la palabra ”Silla” se le pueden atribuir propiedades especificas, esto debe tomarse en cuenta dependiendo de la situación tratada.

- **Especificidad** La especificidad es una característica de abstracción que nos plantea ¿Que tan especifico o general debe representarse una solución? La respuesta para esta interrogante, es que el rango en el cual la abstracción va a variar entre los especifico y general va a depender del número de contextos en el cual es utilizado. (Wagner & Deissenboeck, 2008, p.3).

- **Complejidad** Uno de los propósitos de la abstracción es reducir la complejidad, sin embargo no se determinado una métrica estándar para la complejidad, por lo cual se habla de dos tipos de complejidad, la primera llamada complejidad de detalle la cual puede ser medida por métricas de complejidad, contrariamente, la complejidad dinámica puede describir las relaciones causa y efecto. (Wagner & Deissenboeck, 2008, p.3).

Tipos de Especificaciones

Al abordar en la etapa de diseño de software nos podremos encontrar con una diversidad de contextos en los cuales será necesario el uso de una representación valida y concisa para expresar una idea o abstracción respecto a los requerimientos del sistema en desarrollo. A continuación, veremos los tipos de especificaciones mas comunes, que nos ayudaran a comprender los requerimientos en la etapa de diseño de software.

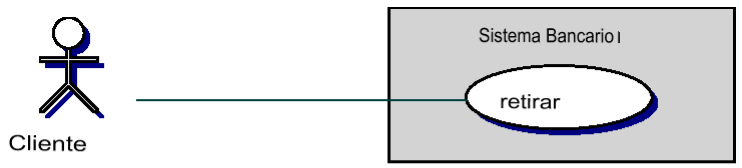
Informales

Las especificaciones informales se caracterizan por ser escritas de una manera libre y en un lenguaje natural. Existe una clara desventaja y es que la posible ambigüedad puede generar malentendido e inconsistencias

Podremos encontrar técnicas de representación informal tales como:

- Especificación de Lenguaje Natural: La representación mediante **lenguaje natural** nos permite describir una abstracción en un formato que a pesar de encontrarse poco depurado o sintetizado es de fácil asimilación. Véase la figura 3.1.

Figura 3.1: Especificación en **Lenguaje Natural**



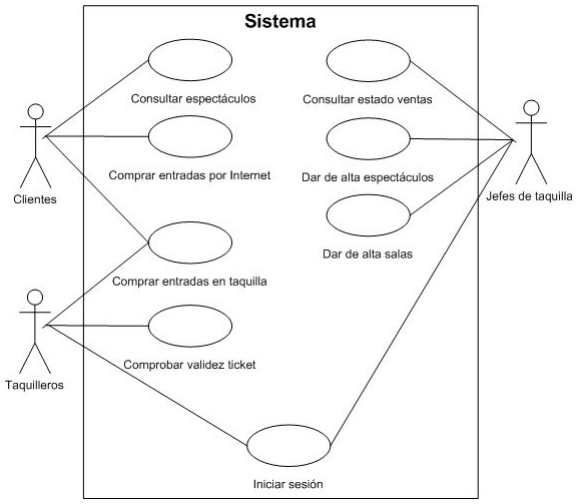
- Si el cliente ha introducido el PIN almacenado en la Tarjeta, y el saldo del cliente era mayor o igual al cantidad solicitada, luego el cliente obtuvo la cantidad solicitada cantidad y la cantidad se dedujo del saldo.
- Si el cliente ingresó el PIN incorrecto tres veces, el la tarjeta fue retenida.
- Si el cliente solicitaba demasiado dinero, la tarjeta devuelto al cliente.

[Fuente: Giese y Heldal, 2004 **Con traducción**]

- Casos de Uso: El propósito de los casos de uso es representar la abstracción en la cual el sistema da respuesta a la interacción realizada por un actor, de esta manera podremos ver

el comportamiento del sistema ante los eventos desencadenados por distintos entes. Véase la figura 3.2.

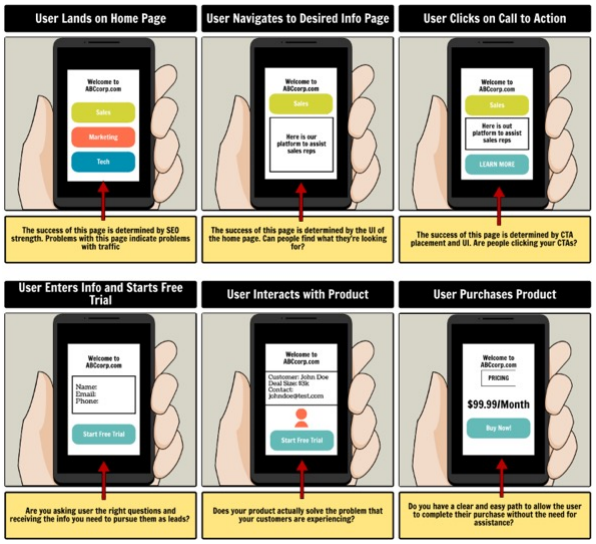
Figura 3.2: Especificación mediante **Casos de Uso**



[Fuente: de la Iglesia, 2022]

- Story Board: El guión gráfico lo que nos permite es expresar una abstracción mediante una secuencia de pasos, en esta secuencia lineal se puede representar el comportamiento conceptual u operacional del sistema. Véase la figura 3.3.

Figura 3.3: Especificación mediante **Story Board**



[Fuente: Bumcrot, s.f.]

Con Formato

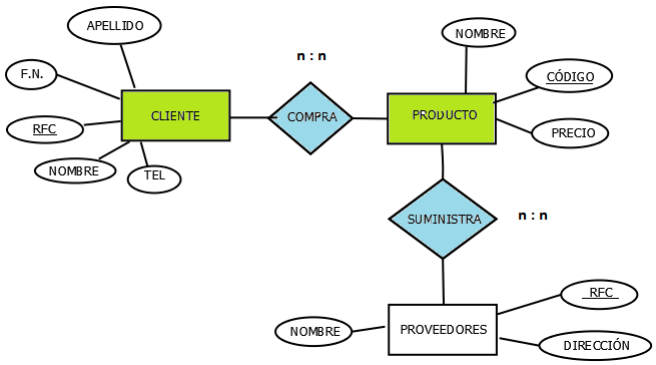
Las especificaciones con formato poseen una sintaxis mas estándar que las informales, por lo cual son mas fiables, ejemplos podrían ser representaciones como los diagramas UML.

En este tipo de especificaciones podemos encontrar:

- Diagramas de entidad-relación:

  El diagrama entidad relación nos permite representar un modelo en el cual se determinan las entidades relevantes de un sistema junto con las interrelaciones y propiedades. Véase la figura 3.4.

Figura 3.4: Diagrama Entidad Relación

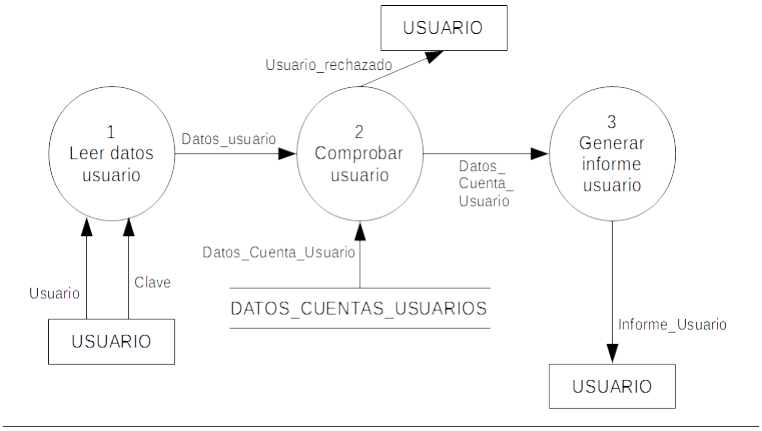


[Fuente: Sulbaran, 1970]

- Diagramas de flujo de datos:

  El diagrama de flujo de de datos nos facilita la expresión de los procesos del sistema, la relación que hay entre cada uno de ellos, con los datos y las entidades. (Domínguez Bejarano & Barrera Gómez, s.f.). Véase la figura 3.5.

Figura 3.5: Diagrama de Flujo de datos

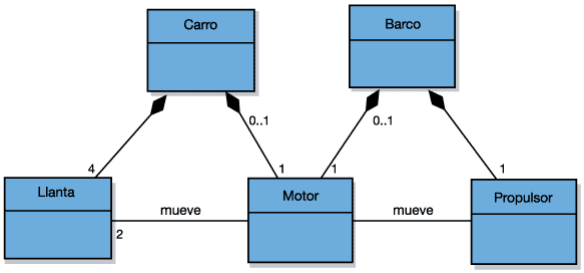


(Imagen de creación propia)

[Fuente: Domínguez Bejarano y Barrera Gómez, s.f.]

- Diagrama de Estructura: La finalidad del diagrama de Estructura es representar precisamente la estructura de un sistema o un componente en módulos, con esto vamos a poder representar una abstracción que de lo general va a los especifico. Véase la figura 3.6.

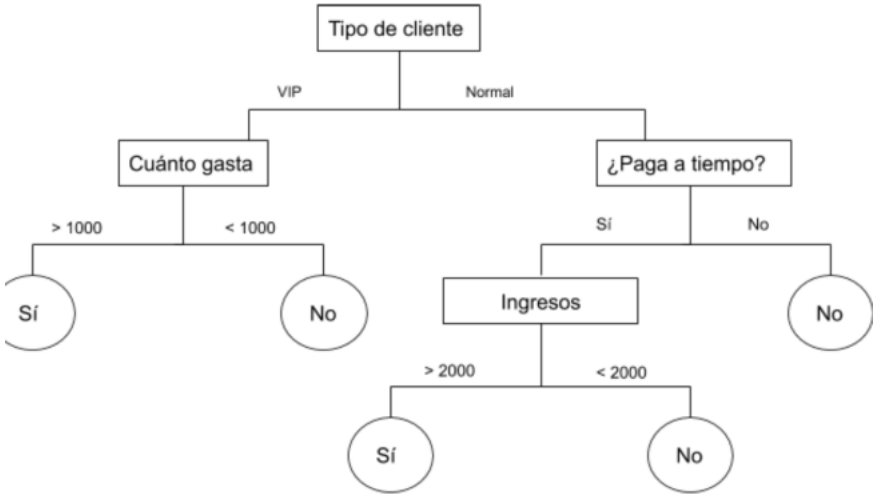
Figura 3.6: **Diagrama de Estructura**



[Fuente: Orozco, s.f.]

- Tabla de decisiones y árbol de decisiones: La importancia del árbol de decisiones es que podemos representar un modelo donde existan condiciones, con lo cual el flujo es caracterizado por seguir la lógica causa y efecto. La ventaja de este diagrama es que podemos ver todos los posibles casos y resultados. Véase la figura 3.7.

Figura 3.7: **Tabla de decisiones y árbol de decisiones**

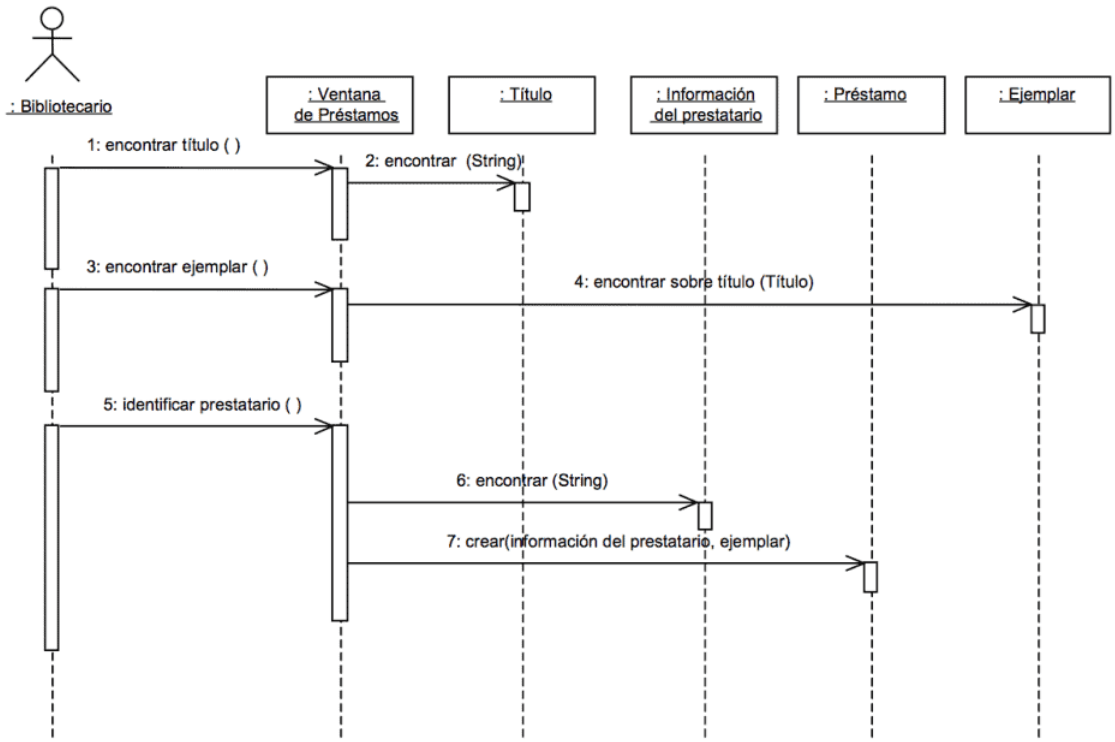


[Fuente: Software, 2021]

- Diagramas de Secuencia:

  El diagrama de secuencia representa como es que los componentes de un sistema se comportan entre si. (manuel.cillero.e, 2016). Véase la figura 3.8.

Figura 3.8: **Diagrama de Secuencia**

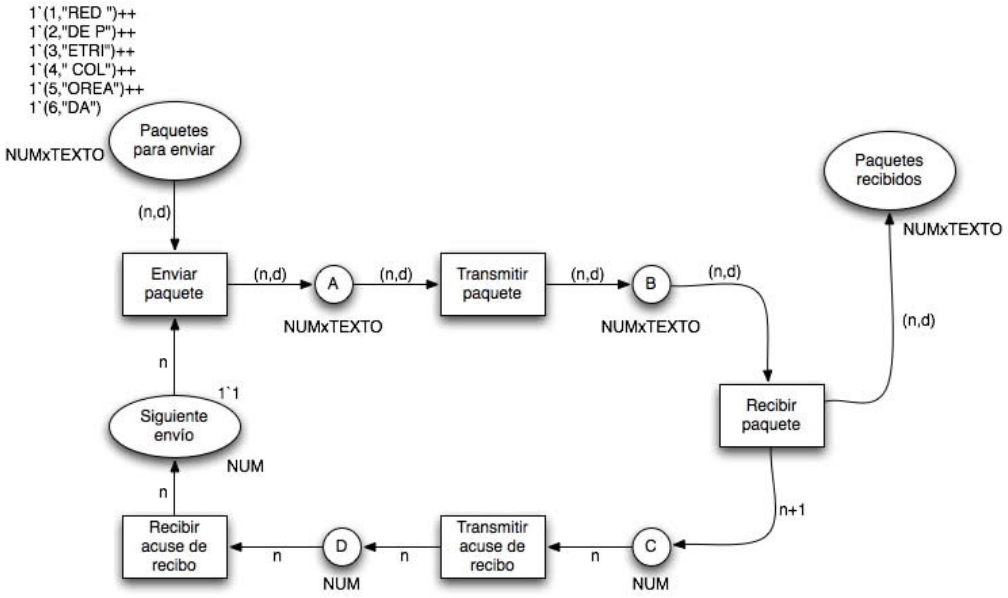


[Fuente: manuel.cillero.e, 2016]

- Redes de Petri:

  Según el autor Almicar Meneses, las Redes de Petri son "ideales para describir y estudiar sistemas que procesan información y con características concurrentes, asíncronas, distribuidas, paralelas, no determinísticas y/o estocásticas".(Meneses, 2002).

Figura 3.9: **Redes de Petri**



[Fuente: del Estado de Hidalgo, s.f.]

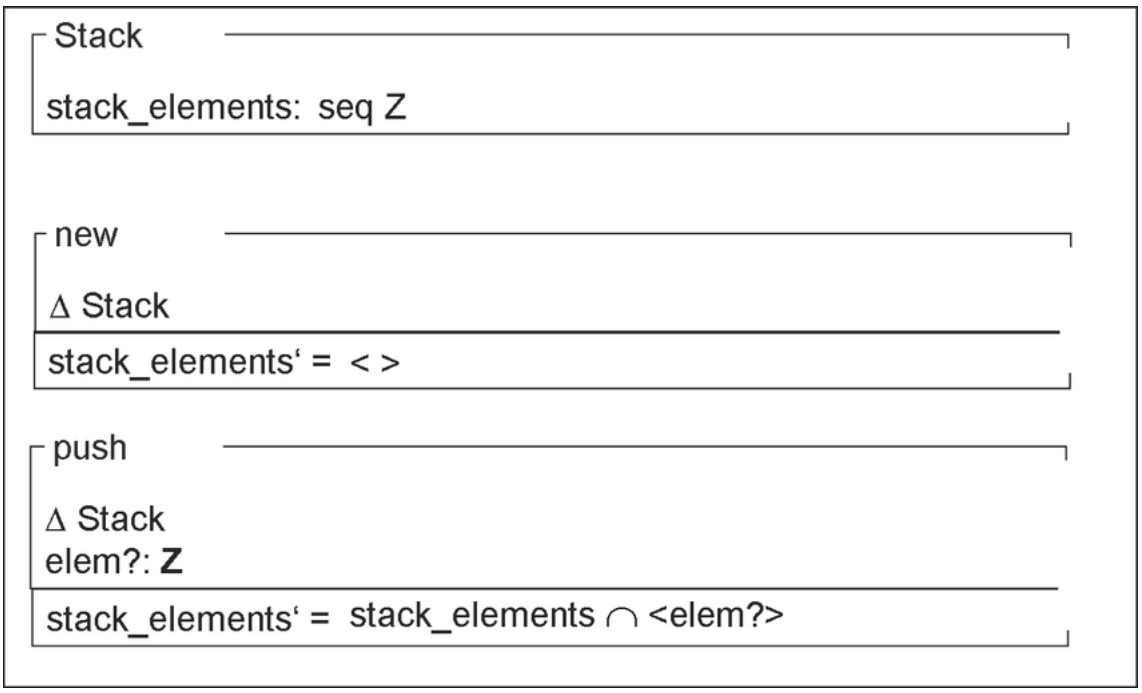
Formales

Las especificaciones formales se caracterizan por una sintaxis más definida, por lo cual elimina la ambigüedad e imprecisión, su desventaja es que puede ser compleja asimilar sin una capacitación previa.

- Lenguaje Z:

  El lenguaje Z es una notación para la expresión de una especificación formal de un sistema, según el auto Jesús Lobato el lenguaje Z: "Se sustenta en la teoría de conjuntos, cálculo de lambda y la lógica de primer orden, emplea esquemas para la representación de estados del sistema y las operaciones que se efectúan sobre el mismo, los cuales contienen variables y predicados que afectan los valores de las variables".(Lobato, 1970). Véase la figura 3.10.

Figura 3.10: **Notación Z**

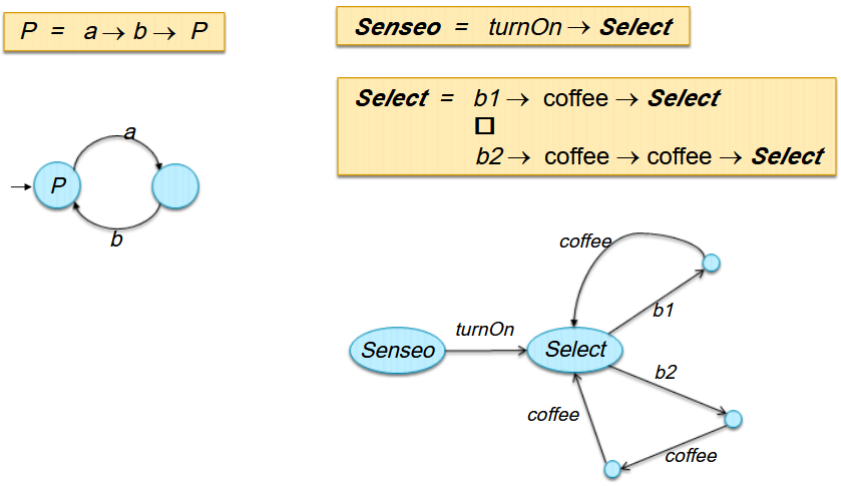


[Fuente: Rakić, 2016]

- Notación CSP:

  La notación CSP o en ingles "Communicating Sequential Processes" que en español seria *Comunicación de procesos secuenciales*, es una notación con sintaxis muy similar a expresiones matemáticas, la autora Jenna Phips nos dice que CSP es: "is a language that uses math and logic to analyze communication between systems. CSP describes concurrent processes, which operate simultaneously in a system but are not parallel" [es un lenguaje que utiliza las matemáticas y la lógica para analizar la comunicación entre sistemas. CSP describe procesos concurrentes, que operan simultáneamente en un sistema pero no son paralelos]. (Phips, 2021). Véase la figura 3.11.

Figura 3.11: **Notación CSP**



[Fuente: SlidePlayer, s.f.]

3.2 Etapa de Investigación

Prosiguiendo con el proceso de diseño de software, tenemos la segunda etapa la cual es la de Investigación. La etapa de la investigación en si, lo que involucra es un trabajo de campo en relación al contexto en donde se desarrolla el sistema, esta labor es importante porque determinaremos en que se deberá enfocara el diseño.

El autor Akash Takyar nos da una descripción de la etapa de investigación.

> This stage comprises research about the target audience to create user personas, analyze the data collected, and determine the user design basics user requirements. The decisions made in this stage become the foundation of building the entire software. [Esta etapa comprende la investigación sobre el público objetivo para crear personajes de usuario, analizar los datos recopilados y determinar los requisitos básicos de diseño del usuario. Las decisiones tomadas en esta etapa se convierten en la base para construir todo el software]. (Takyar, 2022).

Como se expreso en la anterior idea, el trabajo a realizar en la fase de análisis perteneciente al diseño de software, sera en palabras sencillas dirigir los objetivos y cambiar el énfasis del diseño de software conforme las necesidades de diseño investigadas en publico objetivo.

A continuación se recomiendan varios métodos de investigación:

- Entrevista: Uno de los métodos de investigación puede ser el de la entrevista, la finalidad de este método es entablar una comunicación honesta con usuarios, para obtener valiosa información como los problemas que los aquejan y que es lo que esperan para solucionarlos. (Takyar, 2022).
- Grupos de enfoque: Aparte del método de las entrevistas, también existe el de Grupos de Enfoque. Con este método buscamos observar y analizar como un grupo de personas usuarias, debatan sobre sus preferencias, sugerencias y opiniones respecto al diseño que el software debería tener. (Takyar, 2022).
- Encuestas: El método de encuestas, busca consultarle al publico usuario sobre un tema en especifico, al igual que los anteriores métodos la intención es poder obtener las preferencias de los usuarios respecto al diseño del software, el método de encuesta es el que recaba mayor información y datos en poco tiempo. (Takyar, 2022).

Cuando finalizamos la investigación siguiendo los diversos métodos, entramos en la situación en la que tenemos que decidir que enfoque o propiedades deberá optar el proceso de diseño de software, esto lo haremos fundamentándonos en la información recabada del publico usuario en el proceso de investigación.

3.3 Etapa de diseño de software

Tener una base solida para construir es esencial para que un edificio sea estructuralmente estable y los errores de construcción sean mínimos, de igual manera suele suceder con el diseño de software, si el diseño es pobre así sera el producto final, pero si por el contrario se le dedica el tiempo suficiente para crear una base solida, el producto final lo sera igualmente por eso la importancia de realizar diseños de calidad.

Importancia del diseño

Dentro del proceso de desarrollo de software la etapa de diseño es crucial para que el producto a desarrollar posea calidad y valla a poner en riesgo a ningún usuario u organización, ya que no es la primera ves que a través de la historia la mala planificación y falta de diseño en los proyectos de desarrollo de software han provocado grandes consecuencias (Gao y col., 2019), denotando

así lo delicado que puede llegar a ser el omitir ciertos detalles en las etapas de diseño, lo cual dependiendo el fin del software puede llegar tener consecuencias monetarias exorbitantes o hasta vidas humanas.

Un punto importante también es la compresión del problema, ya que en sus totalidad la solución a generar dependerá de cuan claro sea el problema, por ende comprenderlo adecuadamente se vuelve una de las actividades primordiales para los desarrolladores donde una vista gráfica es una herramienta de gran ayuda en el proceso de solución de problemas (Sonnentag, 1998), lo cual podría significar una reducción en el tiempo dedicado por los desarrolladores a la comprensión del problema, pudiendo llevar una ritmo de desarrollo mas fluido.

Ahora bien la etapa de diseño tiene múltiples propósitos, uno de los cuales come menciona Miller, 1989 es aportar un alto nivel de información acerca del software a desarrollar además de aportar gran valor a través de ayudas visuales, de manera que el equipo de desarrollo pueda comprender con mayor claridad los requerimientos de desarrollo, ya que a través del diseño se puede apreciar estructuralmente cada uno de los diferentes módulos que el software tendrá, evitando así la omisión de datos esenciales a la ahora de la implementación.

Los equipos de desarrollo por lo general a lo largo de un proyecto incluyen o sustituyen integrantes dependiendo de las necesidades, lo cual representa un gran reto, donde el correcto diseño de software cumple un papel importante debido a las buenas practicas de diseño, donde a la hora de incluir un nuevo integrante al equipo se reduce el tiempo que se necesita para que se familiarice con el desarrollo (Benedetto & Navon, 2020).

La gran importancia que posee el diseño de software, como se puede apreciar reside en la claridad que ofrece a los desarrolladores, permitiendo de esta manera no solo una apropiada comprensión y solución de problemas, sino que también de esta manera se logran optimizar los tiempos de desarrollo, debido a que cuan mas claro sea el diseño, menor sera el tiempo que se dedique a su comprensión.

Etapas a considerar para el diseño de software

El diseño de software por mas que parezca una etapa solida no lo es, ya que esta esta compuesta de una seria de componentes dentro de los cuales podemos mencionar el wireframing o los diagramas de flujo entre otros mas, los cuales tienen la finalidad de brindar un vista comprensible y estructurada del producto de software a desarrollar, en otras palabras el diseño sera el esqueleto del producto a partir del cual se construirá la solución, donde si este es firme así sera la calidad

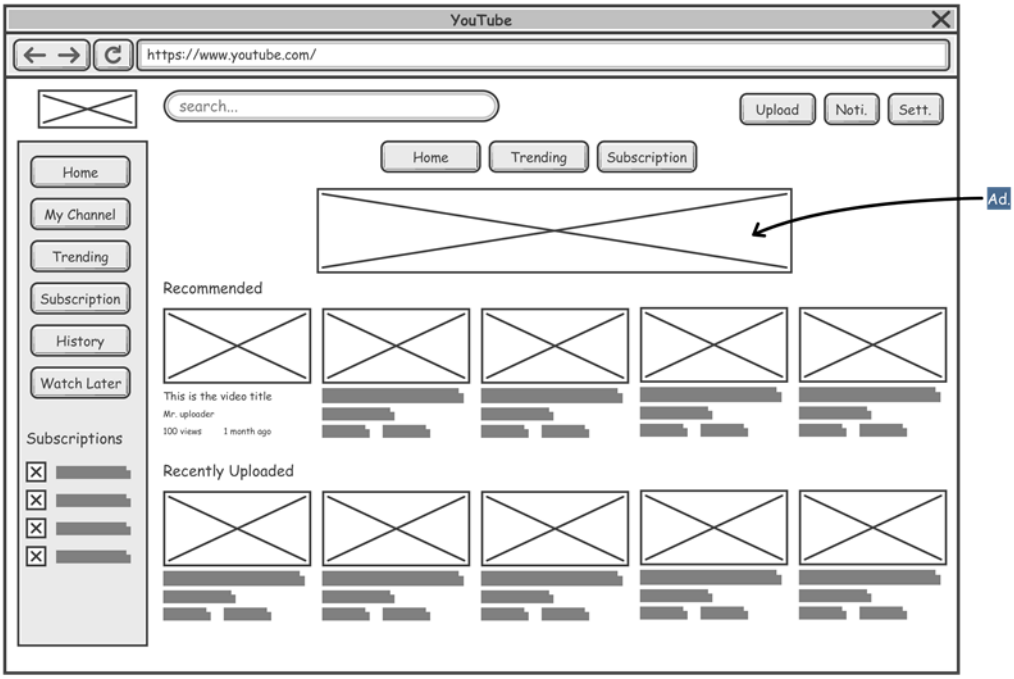
final del desarrollo final.

Wireframing

Wireframing es la parte donde el diseño es completamente estructural, cuyo propósito es llevar a que las partes interesadas y el equipo de desarrollo lo aprueben antes de pasar a una siguiente fase (Rees, 2021), de esta manera se puede ver el proyecto a desarrollar de una manera mas gráfica, lo cual facilita la comprensión del mismo, permitiendo realizar ajustes de así ser necesario en una etapa temprana del desarrollo antes de llegar aun punto donde el cambiar algo podría significar mucho.

La parte estructural, como se menciono anteriormente es la base del desarrollo, es el esqueleto que dará soporte a las siguientes etapas de diseño y desarrollo, además como se aprecia en la imagen 3.12 es un diseño que gráficamente no posee grandes rasgos, esta mas enfocado a brindar una idea sobre el la retrotracción de los módulos que llegara a tener el servicio, de esta manera se vuelve comprensible para todos los involucrados en el desarrollo.

Figura 3.12: Wireframing basada en youtube.com



[Fuente: Archimetric, s.f.]

Ventajas del wiframing

El uso de wireframing puede suponer grandes ventajas en la etapa de diseño donde Lynch, 2022 menciona algunas de estas a continuación:

1. **Visualización clara de la estructura:** permite que las ideas pasen de lo abstracto a algo mas tangible, permitiendo a una parte enviar sus opiniones a otra parte de manera clara, asegurando que todas las partes puedan entenderlo claramente.
2. **Aclarar las características de la interfaz:** un wireframe brinda una visión clara al cliente de la interfaz y las funciones que esta tendrá.
3. **Empuje la usabilidad a la venguardia:** la creación de wireframes lleva la usabilidad a la vanguardia al mostrar los diseños en su mas pura esencia, ya que elimina las imágenes y se enfoca en la estructura obligando a que sea visto de manera objetiva.
4. **Ayuda a afinar la navegación:** permite comprender si el esquema general de navegación es intuitivo, incomprensible o esta en un punto medio, lo cual permite mejorar la navegación.
5. **Proceso de diseño iterativo:** en ves de tratar de combinar las funcionalidad/diseño y los elementos creativos, los wireframes aseguran que estos elementos se tomen uno a la ves, permitiendo que los clientes u otros miembros del equipo brinden comentarios en una etapa temprana del proceso.
6. **Ahorro de tiempo y esfuerzo:** diseños mas claros, facilita la comunicación y evita mal entendidos, de esta manera se ahorra tiempo y trabajo extra por parte del equipo.

Desventajas de wireframing

Como se puede apreciar el el wireframing data la etapa de diseño de ventajas significativas, que ayudan tanto a equipo de desarrollo como las diversas partes interesadas los diversos elementos que poseerá el desarrollo final del software, pero aun sin embargo Malone, 2022 nos menciona una serie de desventajas a considerar:

1. **Los clientes se atascan con detalles minuciosos:** existe la dificultad de explicar porque los wireframes son importantes, ya que hay clientes que están mas interesados en el aspecto que en su comportamiento.
2. **Diseño excesivo:** cuando se dedica demasiado tiempo a el diseño de los wireframes estos pueden tener exceso de detalle, al final lo que se busca es una composición básica, no algo que retrase el proceso.

3. **Otro paso en el proceso:** dependiendo del el equipo de desarrollo puede ser otro paso en el proceso, por lo que creen que si lo omiten avanzaran más rápido.

4. **Limitación al diseño:** al usado wireframes deja muy pocos espacios creativos al diseñador, ya que este deberá adherirse al esqueleto de la página, omitiendo posibilidades creativas que pueden tener un flujo distinto.

A pesar los contrastes entre las ventajas o desventajas de usar wireframing, su uso constituye una herramienta de comunicación y orden en los inicios del proyecto que es más incertidumbre suele haber, organizando las ideas y permitiendo así traerlas al mundo físico a través de un diseño quizá un poco básico pero lo suficiente comprensible para arrancar con las demás etapas del desarrollo.

Historias de usuario

Las historias de usuario ha ganado terreno en el área de desarrollo de software, sobre todo en las metodologías ágiles, debido a que estas expresan los requerimientos funcionales del software desde un punto de vista del usuario (Raharjana y col., 2021), de esta manera es más sencillo comprender lo que el usuario realmente necesita, evitando malentendidos en él los procesos de desarrollo.

Por lo general las historias de usuario posen una estructura básica, que busca facilitar la comprensión de que es lo que se está solicitando, suele cambiar algunas veces la forma en que sé estructurar o como se organizan, pero eso dependerá del equipo de desarrollo, el aporte de Vela, 2020 muestra lo que por lo general tienen las historias de usuario:

- **Titulo:** debe ser claro y hacer referencia a lo que se necesita.
- **Como un:** tipo de usuario que usara la funcionalidad a desarrollar.
- **Quiero:** funcionalidad que el usuario desea.
- **Con el fin de:** finalidad que el usuario quiere lograr dentro del sistema.
- **Criterios de aceptación:** son todos aquellos elementos o restricciones a tomar en cuenta a la hora de realizar el desarrollo de dicha funcionalidad.

Dentro de las historias de usuario las palabras son claves para comprender que desea el cliente, por lo general se suelen realizar tablas para organizarlas como se muestra en el cuadro 3.1 de manera

que visualmente sean atractivas y fáciles de leer, dependiendo del tamaño del proyecto se pueden usar otro tipo de herramientas para una manera más adecuado de las mismas.

Como bien se aprecia primeramente las historias de usuario se centran en los usuarios del producto que se va a desarrollar, lo segundo que sucede es que esta explica desde el punto de vista de usuario como debería acontecer el evento, si haya nada técnico que lo describa, en la tercera parte se expresa el valor de lo que el usuario espera recibir, de manera que se ha mencionado la finalidad de las historias es el usuario, ya que estas suelen estar enfocadas en los distintos tipos de usuarios que usaran el producto final.

Cuadro 3.1: Ejemplo de una tabla de historias de usuario

Titulo:	
Como un:	
Quiero:	
Con el fin de:	
Citerios de aceptacion:	

[Fuente: Elaboración propia]

Ahora bien, hay que analizar muy bien el porqué usarlas, que ventajas o desventajas pueden tener, ya que la finalidad es que a la hora de realizar un desarrollo el equipo pueda realmente comprender que es lo que el cliente espera y que se debe respetar para que el usuario no tenga ningún inconveniente a la hora de usar la herramienta o software diseñado.

Ventajas de las historias de usuario

Dentro de las ventajas podemos encontrar según Stephan, 2016 las siguientes:

1. **Apoya el manifiesto Agile:** las historias de usuario apoyan directamente el enfoque funcional del software sobre la documentación excesiva.
2. **Mejor comunicación verbal:** las historias de usuario enfatizan la comunicación verbal, fomentan la conversación en ves de comunicación escrita.
3. **Aplazar detalles:** las historias de usuario permite aplazar los detalles hasta el punto que haya mejor comprensión de lo que realmente es esencial.
4. **Priorización:** las historias de usuario permiten que las mismas sean priorizadas, basándose en su nivel y buscando el mayor valor para el desarrollo.

5. **Comprensibles:** las historias de usuario están redactadas de manera coloquial, sin usar palabras técnicas del oficio, de esta manera suelen sé muy claras para todo el equipo.

6. **Apoyan el desarrollo iterativo:** el modelo de las historias de usuario permite que seleccionar ”n” cantidad de historias para una iteración, y estas cada vez se vuelven más refinadas o pequeñas.

Ahora bien, respecto a las desventajas en realidad son casi nulas, en realidad estas dependerán de la metodología de desarrollo que se esté empleando, ya que en algunos casos se puede llegar a omitir información, ya que estás por lo general por el método de oral para trasmitir los detalles y lapsos de desarrollo a los interesados (Davu, 2020), pero al final puede no ser una desventaja todo estará en como trabaje el equipo de desarrollo.

Casos de uso

Los casos de uso proporcionan una estructura para expresar requisitos funcionales en el contexto de procesos empresariales y de sistemas, buscando entregar a los interesados(usuarios)a algo de valor (IBM, 2021), por ende logrando recopilar de manera textual los involucrados, condiciones, flujos normales y alternativos de un requerimiento, además de que estos pueden ser representados de manera gráfica mediante diagramas UML.

Diagramas de casos de uso

Con el fin de poder tener una visión clara de los individuos que interactúan con el sistema y viceversa, los diagramas de casos de uso buscan mostrar de manera clara estas interacciones, relacionando así los diversos autores y a los sistemas como los casos de uso que serian las acciones o funciones dentro del software.

De manera básica estos poseen una estructura con 4 elementos esenciales según explica Gaskin, 2022 siendo estos los siguientes:

1. **Sistema:** elemento que interactúa con los actores.

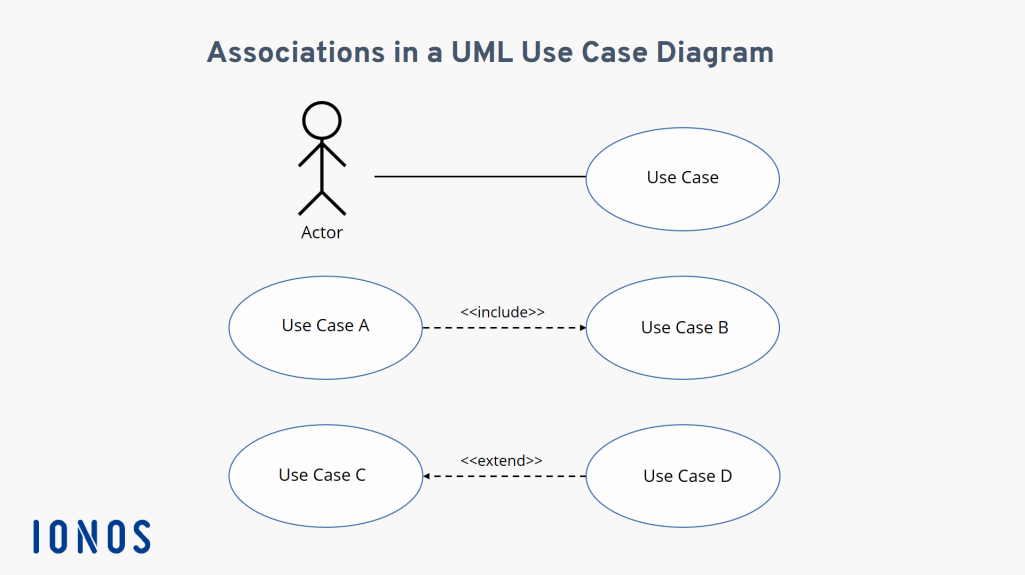
2. **Autores:** usuarios del sistema, que esperan algún fin de este.

3. **Casos de uso:** acción o respuesta generada a partir de las interacciones entre el sistema y los actores.

4. **Relaciones:** conectan los diversos casos de uso con los actores u otros casos de uso.

Las relaciones se dividen en dos tipos «include» o «extend», ya que estas denotan si donde *include* involucra de manera obligatoria otro caso de uso, en otras palabras este no se podrá ejecutar solo, y en el caso de *extend* es más lo opuesto, ya que el comportamiento suele ser más opcional respecto al otro caso de uso, no hay sentido de obligatoriedad.

Figura 3.13: Ejemplo casos de uso



[Fuente: IONOS Digital Guide, 2020]

Redacción de casos de uso

El desarrollo de los casos de uso muestra a detalle las relaciones y flujos del mismo con el actor, de manera que a la hora de leerse estos puedan comprenderse de manera clara, esto debido a que los distintos requerimientos y condiciones están desarrolladas de manera clara y ordenada para dar una mejor visión de lo que se necesita, en la imagen 3.14 Lund y col., 2010 nos muestran una plantilla que puede usarse como herramienta a la hora de desarrollar los casos de uso.

Como se puede apreciar en la imagen 3.14 el nivel de detalles que se muestra es esencial para una mayor comprensión de los requerimientos, ya que junta desde las condiciones iniciales para que el caso puede desarrollarse, su flujo de eventos, pos-condiciones y demás, haciendo así que la estimación o consideración a la hora del diseño sea mucho más precisa.

Figura 3.14: Platilla caso de uso

N°	Nombre del Campo	Descripción del Campo
1.	Nombre del CU:	< Nombre representativo de la función que realiza el CU.>
2.	Actor	<Rol de quién inicia el CU>
	Tipo de Asociación	<Asociación del CU con otro/s CU> [Incluido/ extendido/ generalización padre/ generalización hijo/-]. En el caso de generalización hijo, se debe indicar el caso de uso padre <Id. CU>y/o <nombre del caso de uso padre>, si forma parte de más de una asociación, se deben indicar todas.
4	Breve Descripción:	<Descripción resumida y clara de la funcionalidad del CU. Puede hacer referencia a documento/s que contenga/n información relacionada al CU >
5	Precondiciones:	[<Condiciones que se deben cumplir o estado en el cual se debe encontrar el sistema antes de que el CU pueda iniciarse>]
6	Flujo de Eventos:	<Descripción detallada de las acciones de los actores y las respuestas del sistema que se llevarán a cabo durante la ejecución del CU. >
7	Poscondiciones:	[<Estado del sistema a la conclusión de la ejecución del CU>]
8	Extensión (extend):	[<Id. Paquete> y/o <nombre de paquete> si corresponde] [< Id. CU> y/o <nombre del CU llamado desde este CU como excepción (CU al que se extiende)>]
9	Inclusión (include):	[<Id. Paquete> y/o <nombre de paquete> si corresponde] [< Id. CU> y/o <nombre del CU llamado desde este CU como inclusión (CU que se incluye)>]
10	Requisitos asociados	[<Enumeración de Id. de requisitos que satisface el CU> / <nombre del documento que los contiene>]
11	Consideraciones / Observaciones:	[<Comentarios, requisitos adicionales, no funcionales relacionados al CU a tener en cuenta en el diseño o la implementación>]
12	Suposiciones:	[<Son las condiciones o premisas que se tuvieron en cuenta en el análisis, que permitieron incluir a este CU en la descripción del sistema>]
13	Frecuencia de Uso:	[<Estimación del número de veces que el CU se ejecutará por los actores en una unidad de tiempo.>] [<N° de veces> / <unidad de tiempo]

[Fuente: Lund y col., 2010]

Ventajas y desventajas de los casos de uso

En su mayoría el uso de casos de uso suelen ser aplicados al desarrollo de software interactivo debido a que estos se muestran las diversas interacciones de los usuarios con el producto, también cabe destacar que como herramienta para la extracción de (Coronado, 2009), al final es aportar valor al producto en desarrollo.

Ventajas:

1. Muestran la intención del usuario(actor).
2. Permiten extraer los requerimientos del usuario y sistema.
3. Alto grado de descripción.
4. Tiene en cuenta los diferentes tipos de usuarios que usaran el sistema.

Desventajas:

1. No establecen requisitos funcionales o no funcionales.
2. Se deben complementar con las reglas del negocio, requisitos no funcionales, entre otros.
3. No permiten saber explícitamente lo que el cliente necesita.

Casos de uso 2.0

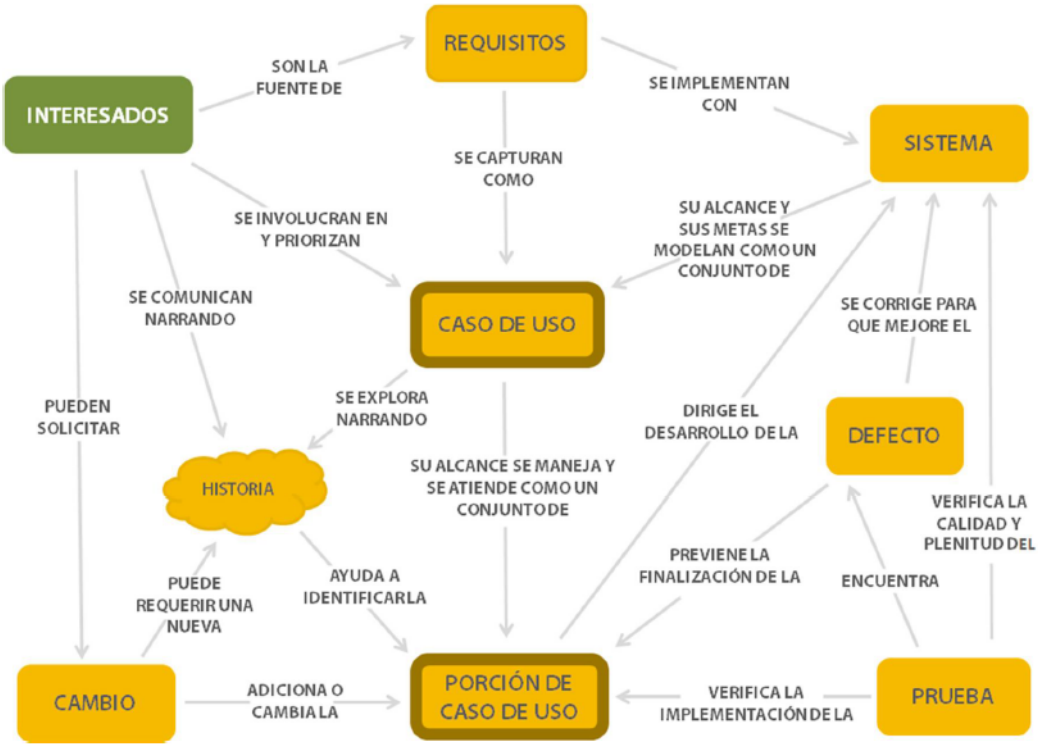
Los casos de uso 2.0 en realidad no difieren en su definición de los casos de uso normales, pero si en su manera de aplicación, ya que es una práctica escalable y ágil para realizar el levantamiento de requisitos y llevar a una metodología de desarrollo incremental (Jacobson y col., 2013). Por lo tanto, para entender un poco como se aplican estos casos de uso hay que entender que hay tres elementos fundamentales, *el caso de uso*, *la historia* y *la porción del caso de uso*, de manera que se cambia un poco respecto a los normales, ya que son elementos que no se suelen usar al desarrollarlos.

Algo que hay que tomar en cuenta es que como se aprecia en la figura 3.15 en estos intervienen:

- Los interesados.
- Los cambios.
- Las pruebas.
- Los defectos.

Este tipo de aplicación de los casos de uso se vuelve más interactiva desde distintos puntos de vista, ya que involucra a los interesados, hace uso de historias, tiene en cuenta cambios, sobre todo que tiende a dividir el caso de uso en porciones, dividiendo en pequeños avances, siempre pensando en la funcionalidad del producto y el cliente, por eso la manera de implementarlos tiende a clarificar y hacer menos complejo el desarrollo.

Figura 3.15: Diagrama casos de uso 2.0



[Fuente: Jacobson y col., 2013]

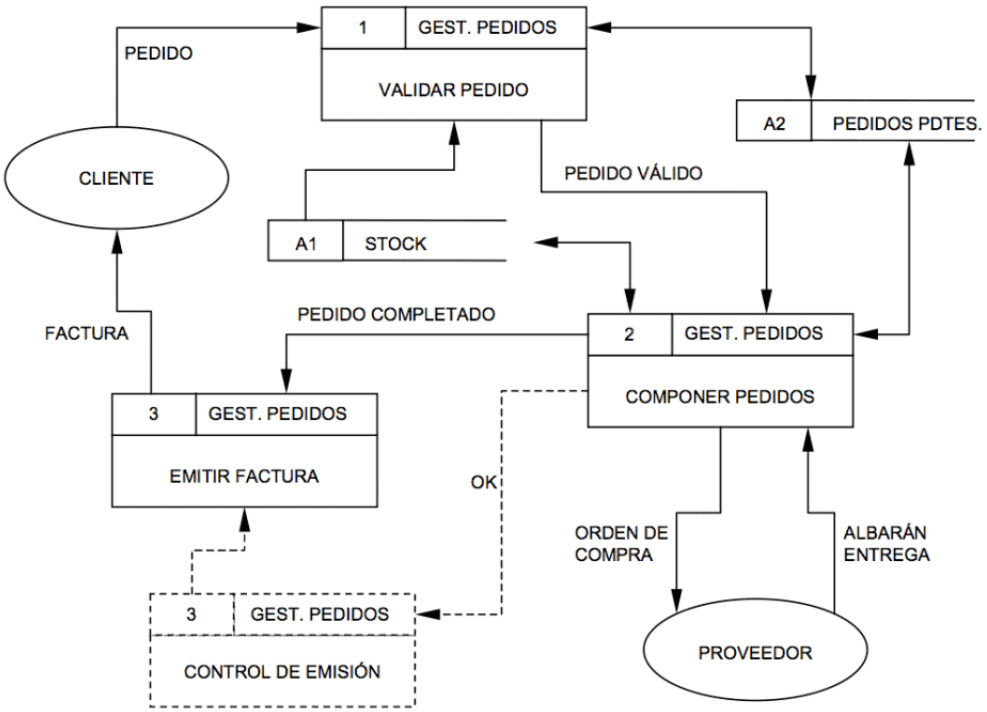
Diagramas de flujo

La complejidad que pueden llegar los flujos de información de un sistema por lo general suele ser algo difícil de explicar o ver a simple vista, por lo tanto tener una manera sencilla de captar su comportamiento viene bien a la hora de su desarrollo, que por lo general sera menos confuso que no tener un recurso gráfico que brinde una visión de el sistema y sus flujos.

> Un diagrama de flujo representa un sistema como una red de procesos funcionales interconectados a través de flujos de datos de esta manera un diagrama de flujo puede ser descompuesto en una jerarquía que representa la propuesta sistema en diferentes niveles de abstracción(Tao & Kung, 1991).

Como expresa Tao y Kung, 1991 anteriormente a través de un diagrama de flujos se puede ver de manera mas clara y sobre todo de forma gráfica los comportamientos que los datos tendrán dentro del sistema, proporcionando a los desarrolladores una herramienta de gran valor, que no solo simplifica el comprender lo que sistema hará sino que se puede usar para explicar de manera simple a cualquier miembro que se integre a lo lago de desarrollo.

Figura 3.16: Diagrama de flujo de datos



[Fuente: Cillero, 2016]

Para comprender mejor los diagramas de flujos de datos es importante conocer sus elementos (Cillero, 2016):

- **Entidad externa:** muestra un agente externo que puede tanto recibir como proporcionar información al sistema.
- **Proceso:** es aquella funcionalidad que mendicante valores de entrada, lo opera y genera valores de salida, transformando o manipulando los datos. Siempre sera el intermediario entre un agente externo y el sistema.
- **Almacén de datos:** es la información en reposo utilizada por el sistema de manera independiente del sistema de gestión de datos. No puede crear, transformar o eliminar datos, además de que no puede estar comunicado con otro almacén o unidad externa.
- **Flujo de datos:** muestra el movimiento de los datos, de manera que establece comunicación entre procesos y los almacenes de datos o unidades externas. Los flujos de datos que comunican procesos con almacenes pueden ser:
  - *De consulta:* muestra el uso de los valores de uno o mas campos de un almacén y si estos cumplen con ciertos criterios específicos.
  - *De actualización:* muestra la actualización de los datos de un almadén, a partir de un nuevo elemento, por eliminación o modificación de otros existentes.
  - *De dialogo:* es el flujo de un proceso y un almacén que representa un consulta y una modificación.
- **Proceso de control:** muestra los procesos que coordinan y sincronizan las actividades de otros procesos.
- **Flujo de control:** muestra el flujo entre un proceso de control y otro proceso, de esta manera el flujo de control que sale de un proceso de control activa al proceso que lo recibe y el que entra le informa de la situaciónn de un proceso.

Ventajas y desventajas

Ventajas

- Ayudan a ilustrar modelos y conectar ideas.

- Favorecen a la comprensión del modelo de forma gráfica.
- Permiten localizar problemas y las diferentes mejoras que se pueden realizar.
- Se pueden apreciar las interacciones de los datos.

Desventajas

- Entre mas grande es el diagrama mas complejidad gana.
- Confusos de seguir cuando existen diferentes caminos.

Interfaz de usuario

Dentro de el diseño de software hay que tener en cuanta muchos factores, tanto de back-end con las funcionalidades que el usuario desea como en el front-end donde esta la GUI(graphical user interface), a través de la cual el usuario debería realizar la interacciones con el sistema, siendo así una de las etapas de diseño que buscara dotar a usuario final de una ambiente cómodo y fácil de comprender.

De manera que como Workana, 2021 la interfaz gráfica de usuario son aquel contenido gráfico mediante el cual se visualiza la información del equipo o sistema a través de una pantalla, ya que por lo general todos los sistemas o programas poseen una interfaz visual, solo algunos que por su estructura no la poseen, pero la finalidad en simplificar esa interacción de usuario con el sistema.

A la hora de entrar en un sitio web, los gráficos, información y herramientas que se despliegan forman esta interfaz gráfica, que por lo general tiende a ser mas intuitiva que otros programas de escritorio, ademas de este tipo de interfaz Workana, 2021 menciona:

- **Interfaz de voz (VUI):** programas que son capases de identificar e interpretar el habla.
- **Interfaz de texto:** se desarrolla mas en el ámbito de los sistemas operativos y es la evolución de la interfaz de comandos primitiva.
- **Interfaz natural:** tipo de interfaz que interpretar las acciones naturales del ser humano, como movimientos y expresiones faciales.

- **Interfaz cerebro-ordenador:** una de la interfaces mas novedosas, no cuenta con muchas aplicaciones cotidianas pero se busca para el control de prótesis biónicas y dar instrucciones sencillas por medio de ondas cerebrales.

A la hora de entrar en el diseño de una interfaz de usuario es importante tener en cuenta que esta va a ser la manera en que el usuario podrá realizar uso de el sistema por lo tanto se deben considerar una serie de elementos o condiciones mencionadas por Zhou, 2018 a continuación:

- **Diseño agradable a la vista:** una interfaz de usuario con un diseño estético y refinado da una buena impresión, siendo agradable al usuario.
- **Interfaz clara:** la interfaz debe proporcionar una visión limpia, estructurada y lógica.
- **Comprensibilidad:** debe estar diseñada de manera que el usuario final no se le complique su uso, que sea intuitiva.

Pueden haber mas elementos a considerar sobre la UI, pero los mencionados anteriormente, muestran la esencia de que no se puede llegar a obviar a la hora de realizar un diseño, ya que este debe ser agradable, claro y comprensible para que las personas que usen el sistema puedan intuir el comportamiento del mismo, reduciendo la resistencia hacia el sistema, en caso contrario seria muy complicado para los usuarios lo que podría terminar com usuarios que no tienen idea de como funciona un sistema.

3.4 Prototipado

En esta sección, observaremos como se realizan el prototipado de los diseños de un Software y los diferentes tipos que existen. Esta etapa es de igual de importante que las anteriores, puesto que en esta se ilustra de forma creativa todo el desarrollo que se llevó en etapas anteriores. El prototipado nos ayuda a mostrarle al cliente, como se verá la aplicación final, con el objetivo que todos los relacionados en el proyecto den el visto bueno para dar en marcha con el proyecto completo. En esta etapa es donde el equipo de UX & UI inician el desarrollo de los diseños

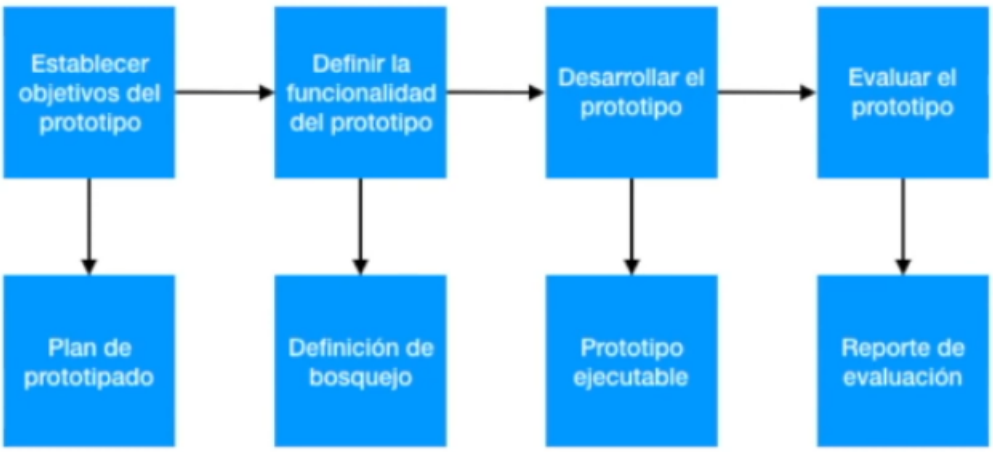
Los beneficios de utilizar los prototipos en nuestros diseños son, según el escritor (Takyar, 2022), son:

1. Probar diseños

2. Analizar si coinciden con los requisitos
3. Recompilar opiniones del cliente
4. Ahorra tiempo y dinero

Existe toda una rama que desarrolla el uso de prototipo, como los la metodología de desarrollo de prototipos.

Figura 3.17: Desarrollo de prototipos



[Fuente: IebSchool]

Fases

1. **Bocetos:** en esta fase, planteamos las ideas del software en bosquejos, con el fin de dar las bases de lo que será el prototipo. Es importante que en esta etapa, se discuta todas las ideas de como será el bosquejo.
2. **Wireframe:** El objetivo de esta fase, es conectar el diseño de la interfaz por medio interactividad, es decir, ver lo que pasa cuando se pulsa un botón, etc. Existen herramientas como **Figma** que facilitan esta labor.
3. **Prototipo no funcional:** En esta etapa se muestra los resultados anteriores. Se prueba el software sin funcionalidades, es decir, solo se muestran los aspectos y dinamismos.

Con la finalidad de facilitar en el desarrollo del prototipado, existen diferentes herramientas que facilitan la creación del prototipo, algunas de ellas son:

1. Marvel
2. Proto.io
3. Niinja Mock
4. Figma

A continuación, se presentan los tipos de prototipados.

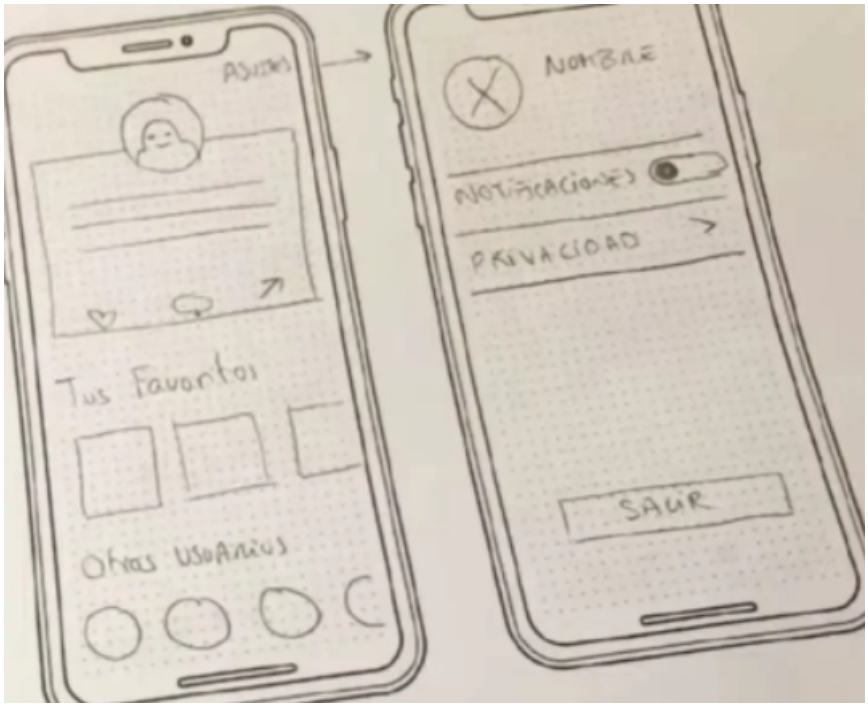
Tipos

Prototipado de baja fidelidad

Este tipo de prototipo, se incluyen las primeras ideas en bosquejos en papel o en tablets (dibujos digitales). Este tipo son ideales para dar las bases a un prototimo estilizado en un software para ello o bien, para dar una idea basica al cliente. Leeway Hert nos menciona que:

> El enfoque de creación de prototipos de baja fidelidad incluye bocetos en papel. Se utiliza principalmente durante las primeras etapas de diseño y es una forma rápida de crear un modelo de diseño de software aproximado. Le permite realizar cambios de forma rápida y conveniente. Ayuda en la lluvia de ideas para nuevas ideas y probar nuevas ideas antes de decidirse por el diseño final.(Takyar, 2022)

Figura 3.18: Ejemplo de un boceto



[Fuente: IebSchool]

Prototipado de fidelidad media

En este tipo, ya se puede tomar una decisión clara si el software cumple con las expectativas propuestas por el cliente. Leeway Hert nos explica que "se considera un trabajo en progreso. Los wireframes y los flujos de tareas se pueden hacer utilizando varias herramientas basadas en computadora para presentar un prototipo básico pero más formal al cliente." (Takyar, 2022)

Prototipado de alta fidelidad

Y por último, este tipo es de alta calidad y con gran detallismo; sin embargo, no es el producto final. Sin embargo, dada su complejidad, su desarrollo es muy tardío. Se puede decir que es como una "**beta**" sin todas las funcionalidades.

Nos explica Leeway Hert que "estos prototipos son necesarios cuando existe la necesidad de una precisión visual y funcional superior. A pesar de que pueden no convertirse fácilmente en código, actúan como una referencia para los diseñadores." (Takyar, 2022)

Figura 3.19: Ejemplo de un boceto

[Fuente: Blog]

3.5 Evaluación

En esta etapa, los prototipos realizados en la etapa anterior, son probados y testeados por el usuario y el equipo de desarrollo. Con esto, se evalúa el nivel de cumplimiento de los requisitos y se logra comprobar consecutivamente el diseño del software, antes de ser desarrollado por el equipo de programadores.

De aquí, se concluyen aspectos del diseño como:

1. Si es sencillo
2. Es directo
3. Facilita al usuario
4. Es accesible para todos los usuarios

A continuación se presentan algunos datos que nos hacen reflexionar sobre la importancia de una buena evaluación de los distintos prototipos:

- El 16,3% de los proyectos software tienen éxito. El proyecto es completado en tiempo y en presupuesto, con todas las características y funcionalidades especificadas al comienzo del proyecto.
- El 52,7% de los proyectos software cuestan más, tardan más o hacen menos.
- El proyecto es completado y operacional, pero con mayor presupuesto que el presupuestado (189% más)
- Tardando más del tiempo estimado y ofreciendo menos características y funcionalidades de las especificadas inicialmente (42%).
- El 31% son cancelados.

El proyecto es cancelado en cierto momento del desarrollo antes de poner el sistema en operación. Existen técnicas para reducir al mínimo los defectos remanentes. Existen técnicas como: Técnicas Estáticas (o Revisiones) y Técnicas Dinámicas.

Figura 3.20: Ejemplo de Evaluación

[Fuente: Platzi]

4. Conclusión

El proceso de diseño de software es el primer y principal paso que uno da hacia la construcción de software eficiente, por lo que es el paso más crucial. Es un proceso centrado en el usuario. Prioriza los requisitos, necesidades y limitaciones del usuario. Juega un papel importante en la atracción de usuarios al producto y la generación de lealtad del usuario. Usando varias herramientas y siguiendo diligentemente los múltiples pasos del proceso de diseño de software, uno puede construir diseños de software superiores, altamente eficientes y fáciles de usar.

El proceso tradicional de desarrollo de sistemas mecatrónicos es criticado por ser inapropiado para el desarrollo de sistemas caracterizados por la complejidad, la dinámica y la incertidumbre, como es el caso de los productos actuales. El software desempeña un papel cada vez más importante en el desarrollo de estos sistemas y se ha convertido en el motor evolutivo de las innovaciones. No sólo implementa una parte importante de parte de la funcionalidad de los sistemas mecatrónicos actuales, sino que también se utiliza para ventajas competitivas. Sin embargo, el proceso actual se divide tradicionalmente en software, electrónica y mecánica, y cada disciplina hace hincapié en sus propias metodologías y herramientas.

Las actividades de desarrollo de sistemas, como la especificación de los requisitos y el diseño, la implementación y la y verificación, están bien definidas en la ingeniería del software. En términos de ingeniería del software, el término En términos de ingeniería del software, el término ”especificación del diseño” se utiliza para referirse a los distintos modelos que se producen durante el proceso de diseño y describen los distintos modelos de las soluciones propuestas. Por tanto, las ”especificaciones de diseño” son descripciones del espacio de soluciones, mientras que las ”especificaciones de análisis” son descripciones del espacio de problemas.

Bibliografía

Archimetric. (s.f.). Wireframe based on youtube.com. https://www.archimetric.com/what-is-wireframe/

Benedetto, J. I. & Navon, J. (2020). Exploiting group shuffling dynamics to convey the importance of good software design. *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 193-196.

Bumcrot, J. (s.f.). How to use storyboards for product development. https://www.uxbooth.com/articles/how-to-use-storyboards-for-product-development/

Cillero, M. (2016). Diagrama de Flujo de Datos (DFD). https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-flujo-de-datos/

Coronado, C. (2009). VENTAJAS Y DESVENTAJAS DE UN CASO DE USO. https://umlmodeladodedatos.blogspot.com/2009/04/ventajas-y-desventajas-de-un-caso-de.html

Davu, S. (2020). User Stories vs. Use Cases: Pros and Cons for Agile Development. https://www.seguetech.com/user-stories-vs-use-cases-pros-cons-agile-development/

de la Iglesia, D. (2022). Generación automática de un índice de recursos bioinformáticos a partir de la literatura científica.

del Estado de Hidalgo, U. A. (s.f.). https://www.uaeh.edu.mx/scige/boletin/icbi/n1/e4.html

Domínguez Bejarano, J. L. & Barrera Gómez, F. J. (s.f.). 3.1. análisis de procesos. http://agrega.juntadeandalucia.es/repositorio/20022017/0a/es-an_2017022012_9122923/31_anlisis_de_procesos.html

Gao, J., Zhang, L., Zhao, F. & Zhai, Y. (2019). Research on software defect classification. *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 748-754.

Gaskin, J. (2022). Todo lo que necesitas saber sobre el diagrama de caso de uso. https://es.venngage.com/blog/diagrama-de-caso-de-uso/

Giese, M. & Heldal, R. (2004). From Informal to Formal Specifications in UML. *UML*.

IBM. (2021). Definición de casos de uso. https://www.ibm.com/docs/es/elm/6.0.3?topic=requirements-defining-use-cases

Icy Science. (1970). ¿Qué es la abstracción? - definición de techopedia - desarrollo - 2022. https://es.theastrologypage.com/abstraction

IONOS Digital Guide. (2020). Associations in a UML case diagram. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/

Jacobson, I., Spence, I., Bittner, K., Caraballo, L. A. S. & Jaramillo, C. M. Z. (2013). Casos de uso 2.0. *La gúıa para ser exitoso con los casos de uso*.

Leveson, N. (s.f.). Requirements specification - massachusetts institute of technology. http://sunnyday.mit.edu/16.355/requirements-notes.pdf

Lobato, J. (1970). Lenguaje de especificación formal z "en breve". http://jesuslobatobaez.blogspot.com/2013/12/lenguaje-de-especificacion-formal-z-en.html

Lund, M. I., Ferrarini Oliver, C., Aballay, L. N., Romagnano, M. G. & Meni, E. (2010). CUPIDo-Plantilla para documentar casos de uso. *V Congreso de Tecnología en Educación y Educación en Tecnología*.

Lynch, A. (2022). Benefits of Wireframe - Reasons to Use Wireframe - Edraw. https://www.edrawsoft.com/benefits-of-wireframe.html

Malone, E. (2022). The pros and cons of wireframing in a website redesign. https://www.growthdrivendesign.com/blog/the-pros-and-cons-of-wireframing-in-a-website-redesign

manuel.cillero.e. (2016). Diagrama de Secuencia. https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-interaccion/diagrama-de-secuencia/

Meneses, A. (2002). http://computacion.cs.cinvestav.mx/~ameneses/pub/tesis/mtesis/node5.html

Miller, A. (1989). Engineering design: its importance for software. *IEEE Potentials*, *8*(2), 14-16. https://doi.org/10.1109/45.31589

Orozco, S. (s.f.). Componiendo Lo Descompuesto: Diagrama de Estructura Compuesta. https://sg.com.mx/content/view/220

Phips, J. (2021). What is CSP? (communicating sequential processes). https://www.webopedia.com/definitions/communicating-sequential-processes/

Raharjana, I. K., Siahaan, D. & Fatichah, C. (2021). User stories and natural language processing: A systematic literature review. *IEEE Access*, *9*, 53811-53826.

Rakić, G. (2016). *SSQSA: Set of Software Quality Static Analysers*. https://doi.org/10.13140/RG.2.1.3975.0643

Rees, D. (2021). What is wireframing. https://www.experienceux.co.uk/faqs/what-is-wireframing/

SlidePlayer. (s.f.). CSP: Communicating sequential processes. overview computation model and CSP primitives refinement and trace semantics automaton view refinement checking. - ppt download. https://slideplayer.com/slide/8289898/

Software, C. (2021). Arboles de decisiones en la Minería de Datos. https://conectasoftware.com/analytics/arboles-de-decisiones-en-la-mineria-de-datos/

Sonnentag, S. (1998). Expertise in professional software design: A process study. *Journal of applied psychology*, *83*(5), 703.

Stephan, F. (2016). Top 5 Advantages Of User Stories. https://www.kaizenko.com/5-advantages-of-user-stories/

Sulbaran, H. (1970). Diagrama Entidad-Relación. https://helisulbaransistemas.blogspot.com/2016/11/modelo-entidad-relacion-el-modelo.html

Takyar, A. (2022). Software design process: UI/UX Design process. https://www.leewayhertz.com/software-design-process/

Tao, Y. & Kung, C. (1991). Formal definition and verification of data flow diagrams. *Journal of Systems and Software*, *16*(1), 29-36.

Vela, J. P. (2020). Los elementos de una buena historia de usuario. https://jeronimopalacios.com/product-delivery/los-elementos-una-buena-historia-de-usuario/#:%5C%7E:text=Los%5C%20elementos%5C%20fundamentales%5C%20de%5C%20una,de%5C%20la%5C%20historia%5C%20de%5C%20usuario.

Wagner, S. & Deissenboeck, F. (2008). Abstractness, specificity, and complexity in software design. *Proceedings of the 2nd international workshop on The role of abstraction in software engineering - ROA '08*. https://doi.org/10.1145/1370164.1370173

Workana. (2021). Interfaz de usuario: Qué es, Tipos de interfaces y más... | Workana. https://i.workana.com/glosario/interfaz-de-usuario/

Zhou, R. (2018). *User interface design principles for Finnish websites' localization in China based on cultural dimensions* (Tesis de maestría).

UCR

UNIVERSIDAD DE COSTA RICA