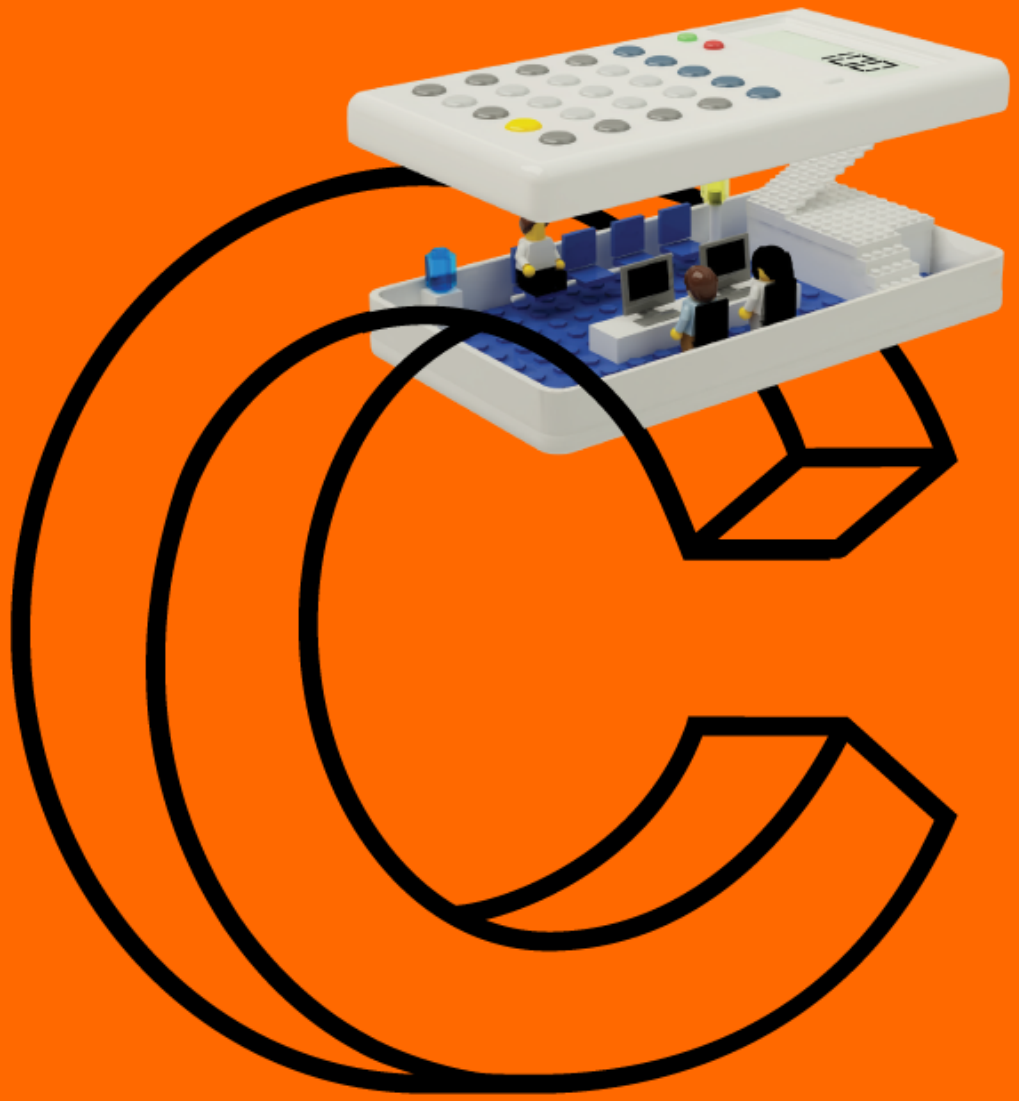
Ingeniería de Requerimientos de Software

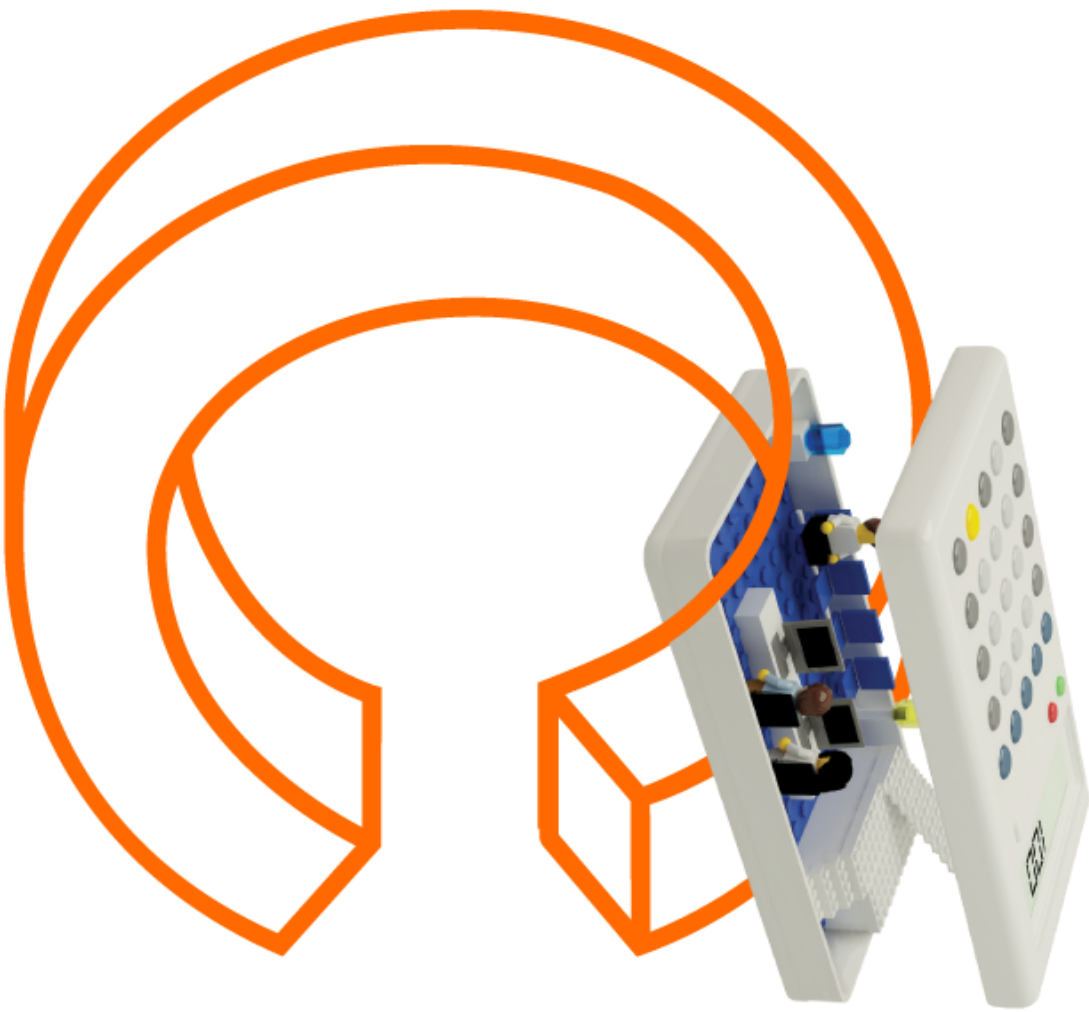


idat

Mg. Daniel Arias, **PMP™** y **Scaled Scrum Master™**

Validación de prototipos de los Requerimientos

Semana 9



Semana 9: Validación de prototipos de los Requerimientos

Logro de aprendizaje

- Planificar y probar el Diseño
- Planear la Prueba
- Diseñar la Prueba

Actividades

Actividad 09:

Elabora un Plan de Validación de prototipos de los requerimientos

Guía IR-09:

Validación de Prototipos

Aprendemos:

¿Qué aprendimos la clase pasada?

- Desarrollo de Interfaz Gráfica de Usuario (GUI).
- Requerimientos funcionales y no funcionales.
- Presentación Prototipo del Sistema

Aprendemos:

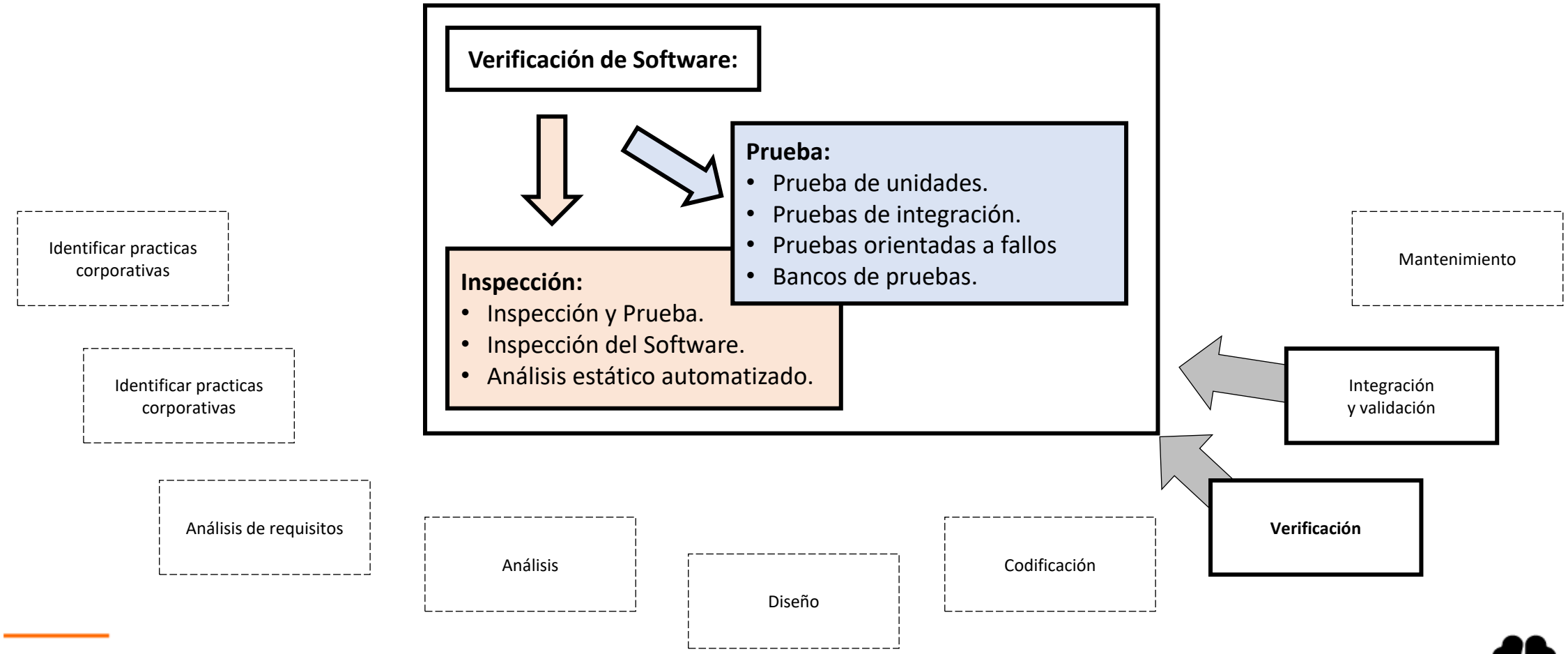
Verificación y Validación

El objetivo es introducir la verificación y validación del software con énfasis en las técnicas de verificación estática y en la prueba dinámica de código. Objetivo de este tema son:

- Comprender la diferencia entre verificación y validación del software.
- Valorar la inspección del software y el análisis estático como métodos de descubrir fallos y mejorar la calidad del software.
- Conocer las técnicas de pruebas para descubrir fallos en el código.
- Analizar las técnicas específicas para las pruebas de componentes y pruebas de sistemas orientados a objetos.
- Importancia de las herramientas CASE para la verificación de software y apoyar el desarrollo de las pruebas.

Aprendemos:

Situación dentro del proceso de desarrollo



Aprendemos: Verificación y Validación

Verificación:

¿Estamos construyendo el producto correctamente?

Se comprueba que el software cumple los requisitos funcionales y no funcionales de su especificación.

Validación:

¿Estamos construyendo el producto correcto?

Comprueba que el software cumple las expectativas que el cliente espera.

Con V&V se pretende certificar que el sistema desarrollado se ajusta a lo esperado de la forma mas satisfactoria posible. Nunca se puede demostrar que está libre de defectos

Aprendemos: Verificación y Validación

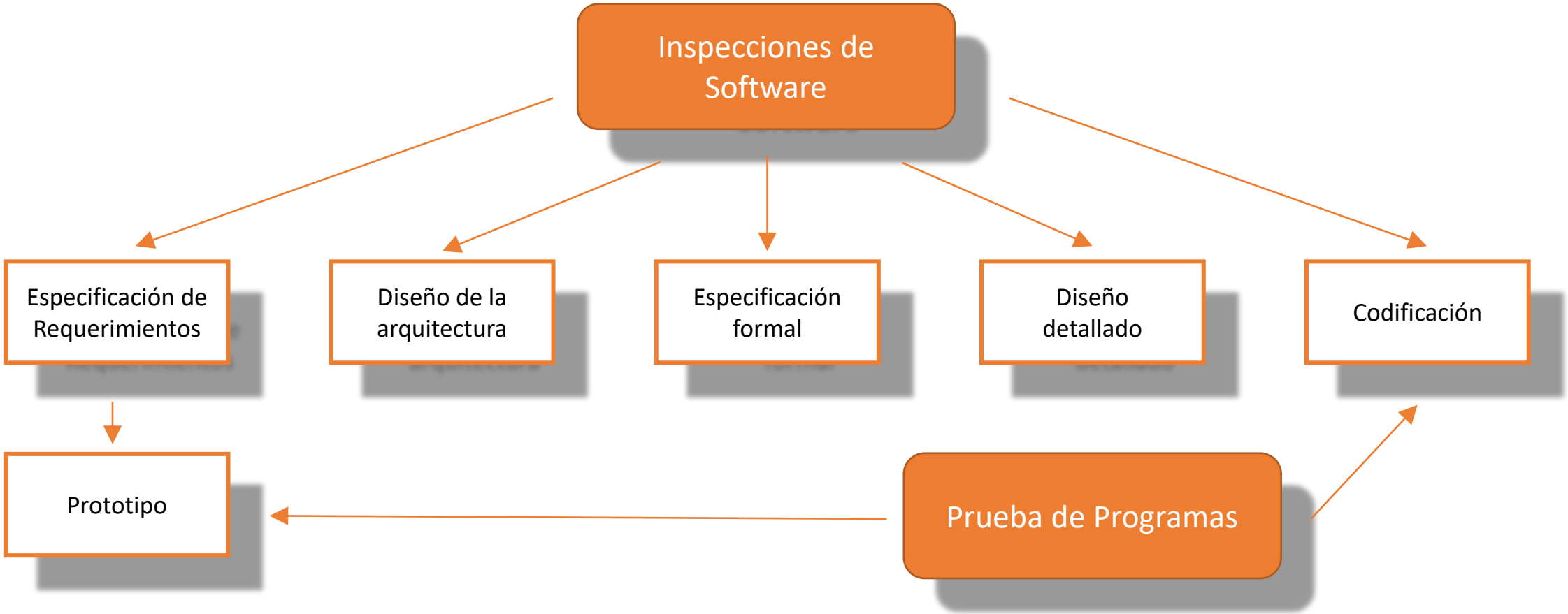
Técnicas de Verificación y Validación

En el proceso de Verificación y Validación se utilizan dos técnicas de comprobación y análisis:

- **Inspecciones del Software:**
  - Se contrasta estáticamente las diferentes representaciones del sistema (diagramas de requerimientos, diagramas de diseño y código fuente) en búsqueda de defectos.
  - No requiere que el código se ejecute
  - Debe realizarse durante todo el ciclo de desarrollo.
- **Las pruebas del Software:**
  - Se contrasta dinámicamente la respuesta de prototipos ejecutables del sistema con el comportamiento operacional esperado.
  - Requiere disponer de prototipo ejecutables y esto sólo es posible en las fases finales del proceso de desarrollo.

Aprendemos: Verificación y Validación

Verificación y validación estática y dinámica



Aprendemos:

Verificación y validación estática y dinámica

En el esquema se muestra el lugar que ocupan las inspecciones y las pruebas dentro del proceso de desarrollo de software. Las flechas indican las fases del proceso en las que se utilizan las técnicas. Las inspecciones de software se pueden utilizar en todas las etapas del proceso, mientras que las técnicas de prueba sólo se pueden cuando está disponible un prototipo o código ejecutable.

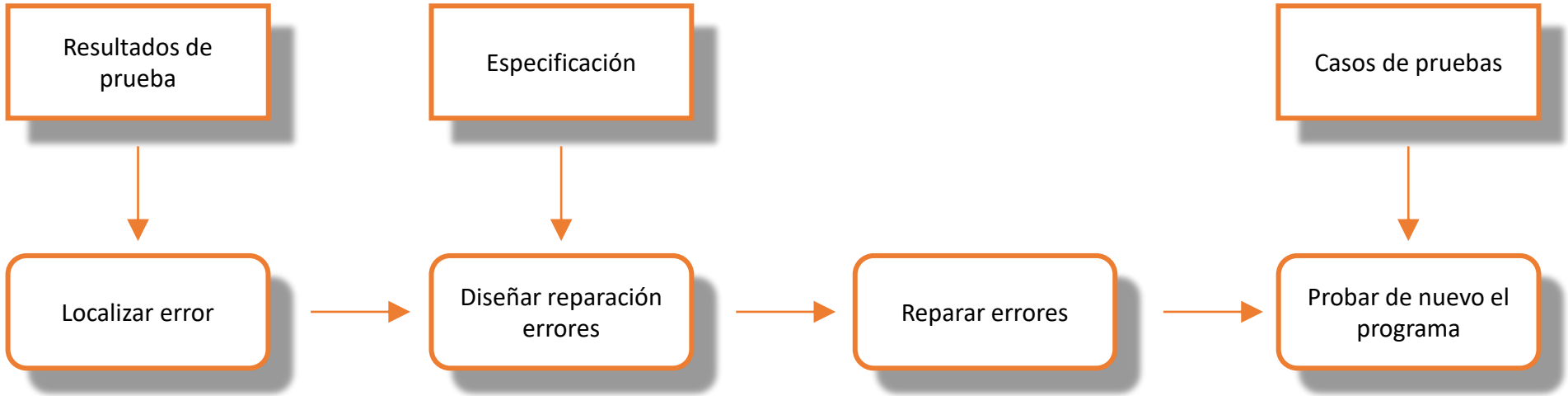
Las técnicas de inspección incluyen inspección de programas, análisis automatizado de código fuente y verificación formal. Sin embargo, las técnicas estáticas sólo pueden comprobar la correspondencia entre un programa y su especificación (verificación) y no puede probar que el software es de utilidad operacional, y mucho menos que las características no funcionales del software son las correctas. Por lo tanto, para validar un sistema de software, siempre se requieren llevar a cabo ciertas pruebas.

Aunque en la actualidad las inspecciones se utilizan ampliamente, las pruebas de los programas es aún la técnica de verificación y validación predominante.

Aprendemos:

Proceso de depuración

- Proceso de depuración: Proceso que localiza y corrige los errores descubiertos por las pruebas.
  - Es un proceso complicado pues no siempre los errores se detectan cerca del punto en que se generaron.
  - Se utilizan herramientas de depuración, que facilitan el proceso
- Después de reparar el error, hay que volver a probar el sistema (pruebas de regresión).
  - La solución del primer fallo puede dar lugar a nuevos fallos.



Aprendemos:

Proceso de depuración

- Al proceso de eliminación de los errores que se descubren durante las fases de prueba se denomina depuración. Es un proceso independiente que no tiene porqué estar integrado:
  - La verificación y validación establece la existencia de defectos en el programa.
  - La depuración es el proceso que localiza el origen y corrige estos defectos.
- No existe un proceso sencillo para la depuración de programas. Los mejores depuradores buscan patrones en los resultados de las pruebas donde el defecto se detecta, y para localizar el defecto utilizan el conocimiento que tienen sobre el tipo de defecto, el patrón de salida, así como del lenguaje y proceso de programación. El conocimiento del proceso es importante. Los depuradores conocen los errores de los programadores comunes (como olvidar incrementar un contador, errores de direccionamiento de punteros en lenguaje C, etc.) y los comparan contra los patrones observados.

Aprendemos:

Proceso de depuración

- Localizar los fallos es un proceso complejo porque los fallos no necesariamente se localizan cerca del punto en que se detectan. Para localizar un fallo de un programa el programador responsable de la depuración tiene que diseñar programas de prueba adicionales que repitan el fallo original y que ayudan a descubrir el origen del fallo. En estos casos las herramientas de depuración que permiten rastrear el programa y visualizar los resultados intermedios es de una gran ayuda.
- Las herramientas de depuración son habitualmente parte de las herramientas de apoyo al lenguaje y que sirven de base al compilador. Proporcionan un entorno especial de ejecución que permiten acceder a las tablas de símbolos del compilador, a través de ella a los valores de las variables del programa. Habitualmente, permiten controlar la ejecución paso a paso sobre el código del programa.
- Después de que se descubre el origen del fallo en el programa, este debe corregirse y entonces reevaluar el sistema. Esto implica repetir de nuevo las pruebas anteriores (pruebas de regresión). Estas pruebas se hacen para comprobar que los cambios introducidos resuelven definitivamente el fallo y no introducen nuevos fallos. La estadística muestra que la reparación de un fallo frecuentemente es incompleta y además introduce nuevos fallos.

Aprendemos:

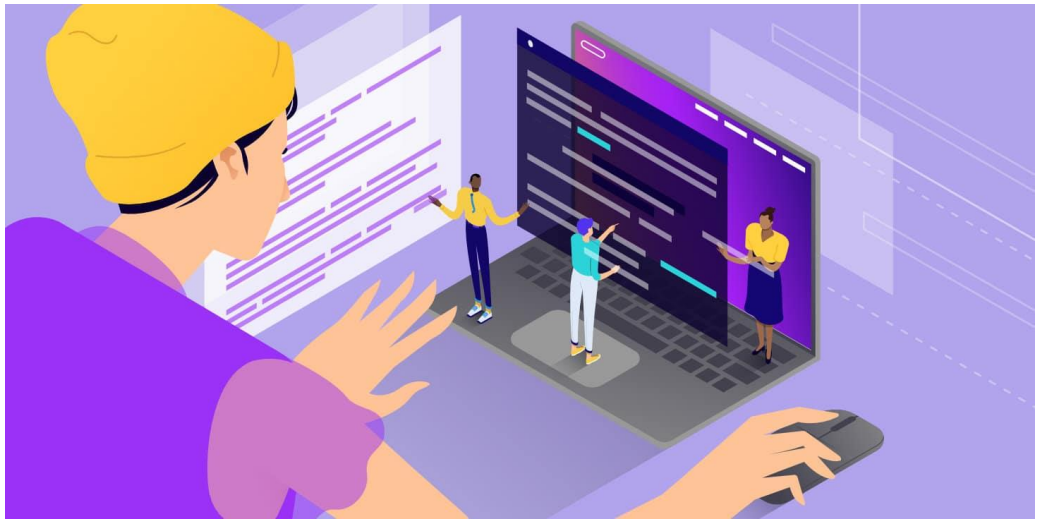
Inspecciones del software

- Consisten en revisiones sistemáticas tanto de los documentos generados como de los códigos fuentes con el único objetivo de detectar fallos.
  - Permiten detectar entre un 60 y un 90% de los fallos a unos costes mucho mas bajos que las pruebas dinámicas.
  - Permiten detectar múltiples defectos en una simple inspección, mientras que las pruebas solo suelen detectar un fallo por prueba.
  - Permite utilizar el conocimiento del dominio y del lenguaje, que determinan los principales tipos de fallos que se suelen cometer.
  - Las inspecciones son útiles para detectar los fallos de módulos, pero no detectan fallos a nivel de sistema, que ha de hacerse con pruebas.
  - Las inspecciones no son útiles para la detección de niveles de fiabilidad y evaluación de fallos no funcionales.
  - Las inspecciones se pueden aplicar a la detección de fallos en cualquiera de los documentos generados

Aprendemos:

Inspección del código fuente: Análisis estático automatizado

- Los analizadores estáticos de programas son herramientas de software que rastrean el texto fuente de un programa, en busca de errores no detectados por el compilador.
- En algunos lenguajes de programación no son muy útiles pues el compilador ofrece una gran información acerca de posibles errores (incluso en ejecución).



Aprendemos:

Inspección del código fuente: Análisis estático automatizado

- Aspectos habitualmente analizados son:
  - Análisis de flujo de control: Identifica y señala los bucles con múltiples puntos de salida y las secciones de código no alcanzable.
  - Análisis de utilización de datos: Señala como se utilizan las variables del programa: Variables sin utilización previa, que se declaran dos veces, declaradas y nunca utilizadas. Condiciones lógicas con valor invariante, etc.
  - Análisis de interfaces: Verifica la declaración de las operaciones y su invocación. Esto es inútil en lenguajes con tipado fuerte (Java, Ada) pero si en los restantes.
  - Análisis de flujo de información: se identifican las dependencias entre las variables de entrada y las de salida
  - Análisis de la trayectoria: Identifica todas las posibles trayectorias del programa y presenta las sentencias ejecutadas en cada trayectoria.

Aprendemos:

Lista de fallos

- Fallos de datos:
  - ¿Las variables se inicializan antes que de que se utilicen los valores?.
  - ¿Todas las constantes tienen nombre?
  - ¿El límite superior de los arrays es igual al tamaño de los mismos?
  - Las cadenas de caracteres tienen delimitadores explícitos.
  - ¿Existe posibilidad que el buffer se desborde?
- Fallos de control:
  - Para cada instrucción condicional ¿Es correcta la condición?
  - ¿Todos los ciclos terminan?
  - ¿Los bloques están delimitados correctamente?
  - En las instrucciones case ¿Se han tomado en cuenta todos los casos?
  - Si se requiere un break ¿Se ha incluido?.

Aprendemos:

Lista de fallos (Cont.)

- Fallos de entrada/salida:
  - ¿Se utilizan todas las variables de entrada?
  - Antes de que salgan ¿Se les han asignado valores a las variables de salida?
  - ¿Provocan corrupción de los datos las entradas no esperadas?
- Fallos de la interfaz:
  - ¿Las llamadas a funciones y métodos tienen el número correcto de parámetros?
  - ¿Concuerdan los tipos de los parámetros formales y reales?
  - ¿Están los parámetros en el orden adecuado?
  - Si los componentes acceden a memoria compartida, ¿Tienen el mismo modelo de la memoria compartida?

Aprendemos:

Lista de fallos (Cont.)

- Fallos de gestión de almacenamiento:
  - Si una estructura con punteros se modifica, ¿Se reasignan correctamente todos los punteros?
  - Si se utiliza almacenamiento dinámico, ¿se asigna correctamente el espacio?
  - ¿Se desasigna explícitamente la memoria después de que se libera?
- Fallos de gestión de las excepciones:
  - ¿Se toman en cuenta todas las condiciones de errores posibles?.

Aprendemos:

Pruebas del software

- Las pruebas se realizan en cuatro etapas:
  - Prueba de unidades (prueba de métodos y clases)
    - se prueba cada método y clase de forma independiente
  - Prueba de integración o de subsistemas
    - se prueban agrupaciones de clases relacionadas
  - Prueba de sistema
    - se prueba el sistema como un todo
  - Prueba de validación (o de aceptación)
    - prueba del sistema en el entorno real de trabajo con intervención del usuario final

El descubrimiento de un defecto en una etapa requerirá la repetición de las etapas de prueba anteriores

Aprendemos:

Tipos de pruebas del software

- Existen dos tipos diferentes de pruebas:
  - Las pruebas de defectos:
    - Buscan las inconsistencias entre un programa y su especificación.
    - Las pruebas se diseñan para buscar los errores en el código.
    - Demuestran la presencia, y no la ausencia, de defectos.
  - Las pruebas estadísticas:
    - Buscan demostrar que satisface la especificación operacional y su fiabilidad.
    - Se diseñan para reflejar la carga de trabajo habitual.
    - Sus resultados se procesan estadísticamente para estimar su fiabilidad (contando el número de caídas del sistema) y sus tiempos de respuesta.

No existe una separación clara entre estos dos tipos de pruebas. Durante las pruebas de defectos los desarrolladores obtienen una visión intuitiva de la fiabilidad, y durante las pruebas estadísticas, se descubren obviamente muchos fallos.

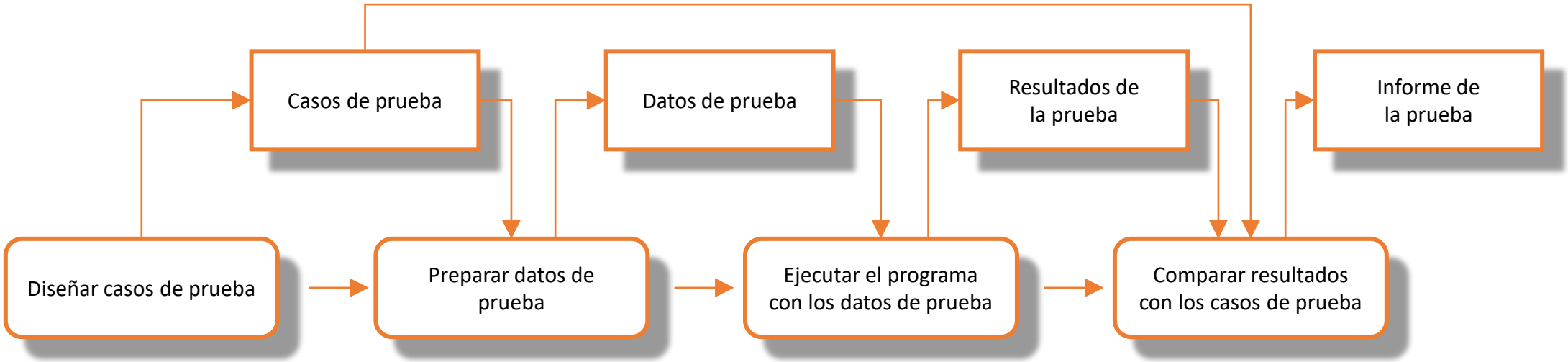
Aprendemos: Tipos de pruebas del software

Pruebas de defectos

- El objetivo de las pruebas de defecto es detectar los defectos latentes de un sistema software antes de entregar el producto.
  - Una prueba de defectos exitosa es aquella que descubre un fallo, esto es un comportamiento contrario a la especificación.
  - Las pruebas de defectos demuestran la existencia de un fallo, y no la ausencia de cualquier fallo.
- Las pruebas exhaustivas no son posibles y deben sustituirse por subconjunto basados en políticas de prueba. Por ejemplo:
  - Se deben probar todas las funciones del sistema que se acceden a través de menú.
  - Se deben probar las combinaciones de funciones que se acceden a través de menú.
  - Si la entrada es introducida por el operador, todas las funciones deben probarse con entradas correctas e incorrectas.

Aprendemos: Tipos de pruebas del software

Proceso de pruebas de defectos



En la figura se muestra un modelo general del proceso de prueba de defectos. Los casos de prueba son especificaciones de las entradas a la prueba y de la salida esperada del sistema, mas una declaración de lo que se prueba. Los datos de prueba son las entradas seleccionadas para probar el sistema. Dichos datos se generan, algunas veces, de forma automática, pero esto no es frecuente, ya que en tal caso no se dispone de las respuestas que hay que esperar del sistema (sin fallo).

Aprendemos: Tipos de pruebas del software

Pruebas funcionales (“Caja negra”)

- Las pruebas funcionales o de caja negras es una política de selección de casos de pruebas basado en la especificación del componente o programa.
- Las pruebas se seleccionan en función de la especificación y no de la estructura interna del software.

Aprendemos: Tipos de pruebas del software

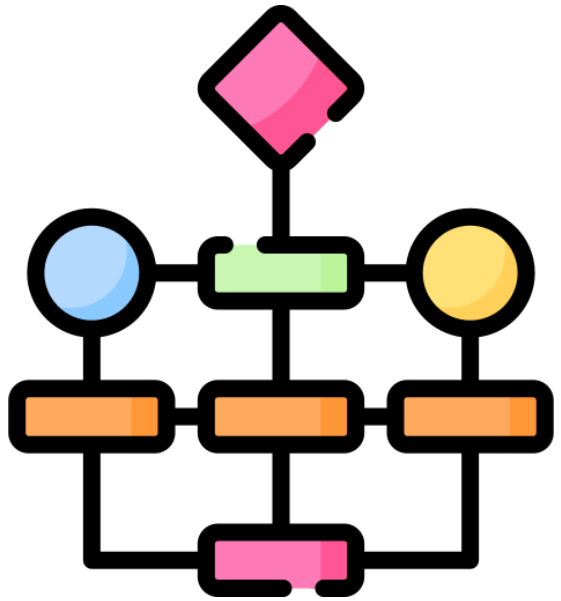
Pruebas funcionales o de caja negra

Son una estrategia para seleccionar las pruebas de fallos basándose en las especificaciones de los componentes y programas, y no del conocimiento de su implementación. El sistema se considera como una caja negra cuyo comportamiento sólo se puede determinar estudiando las entradas y de contrastarlas con las respuestas que proporciona el sistema.
Este enfoque se puede aplicar de igual forma a los sistemas que están organizados como librerías de funciones, o como objetos. El probador introduce las entradas en los componentes del sistema y examina las salidas correspondientes. Si las salidas no son las previstas, entonces la prueba ha detectado exitosamente un fallo en el software.
El problema clave para el probador de defectos es seleccionar la entrada que tienen una alta probabilidad de ser miembro del conjunto. En muchos casos la selección se basa en la experiencia previa de los ingenieros de pruebas. Ellos utilizan el conocimiento del dominio para identificar los casos de prueba que probablemente van a mostrar fallos. También se han propuesto enfoques sistemáticos de la selección de datos de prueba.

Aprendemos: Tipos de pruebas del software

Pruebas estructurales (“Caja blanca”)

- En las pruebas estructurales las pruebas se seleccionan en función del conocimiento que se tiene de la implementación del módulo.
- Se suelen aplicar a módulos pequeños.
- El probador analiza el código y deduce cuantos y que conjuntos de valores de entrada han de probarse para que al menos se ejecute una vez cada sentencia del código.
- Se pueden refinar los casos de prueba que se identifican con pruebas de caja negra.



Aprendemos: Pruebas estructurales (“Caja blanca”)

Ejemplo de prueba estructural

```
public static void search (int key, int[] elemArray, Result r) {
      int bottom= 0;
      int top =elemArray.length-1;
      int mid;
      r.found=false; r.index=-1;
      while (bottom <=top) {
         mid = (top+bottom)/2;
         if (elemArray[mid] == key) {
            r.index=mid; r.found=true;
            return;
         }
         else {
            if (elemArray[mid]<key)
               bottom =mid+1;
            else
               top=mid-1;
         }
      }
   }
```

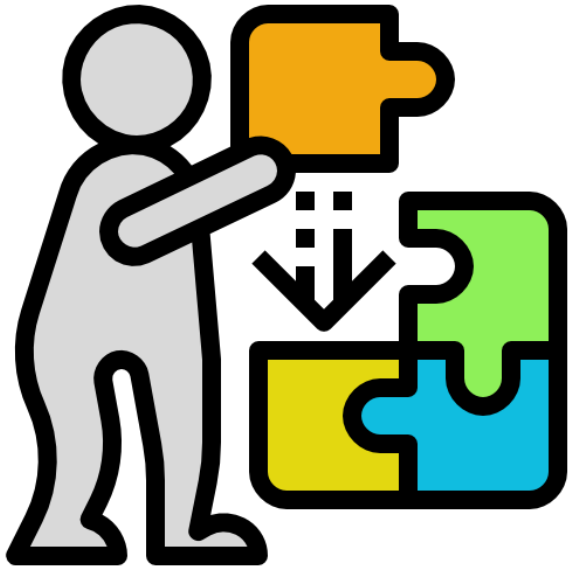
elemArray	key	r
17	17	(true,1)
17	0	(false,??)
17,21,23,29	17	(true,1)
9,16,18,30,31,41,45	45	(true,7)
17,18,21,23,29,38,41	23	(true,4)
17,18,21,23,29,33,38	21	(true,3)
12,18,21,23,32	23	(true,4)
21,23,29,33,38	25	(false,?)

El ejemplo Java, consiste en una función de búsqueda binaria que busca el entero key dentro del elemArray, retornando en el objeto r un campo booleano r.found que indica si ha sido encontrado, y un campo r.index que retorna la posición del elemento encontrado.
Las 8 pruebas que se proponen hacen que todas las sentencias del código se ejecuten al menos una vez, y que en todas las sentencias condicionales se elijan cada una de las dos opciones que tienen.

Aprendemos: Tipos de pruebas del software

Pruebas de integración

- Se prueban la respuesta de grupos de módulos interconectados a fin de detectar fallos resultantes de la interacción entre los componentes.
- Las pruebas de integración se realizan con referencia a las especificaciones del programa.
- La principal dificultad de las pruebas de integración es la localización de los fallos.
- Para facilitar la detección de los errores se utilizan técnicas incrementales.



Aprendemos: Tipos de pruebas del software

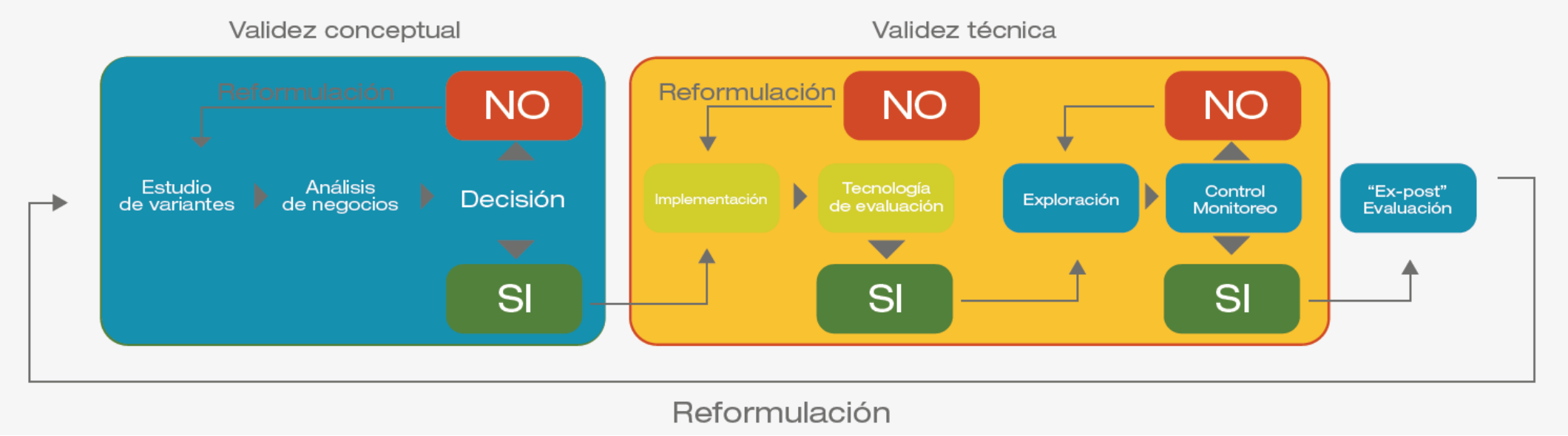
Pruebas de integración

- Una vez que se han probado los componentes individuales del programa, deben integrarse para crear un sistema parcial o completo. En el proceso de integración hay que probar el sistema resultante con respecto a los problemas que surjan de las interacciones de los componentes.
- Las pruebas de integración se desarrollan a partir de la especificación del sistema y se inician tan pronto como estén disponible versiones utilizables de algunos componentes del sistema.
- La principal dificultad que surge en las pruebas de integración es localizar los errores que se descubren durante el proceso. Existen interacciones complejas entre los componentes del sistema y cuando se descubre una salida anómala, es difícil encontrar la fuente de error.
- Para hacer mas fácil la localización de errores , siempre se utiliza un enfoque incremental para la integración y prueba del sistema. De forma inicial se deben integrar un conjunto operativo mínimo, y probarlo. Luego se agregan nuevos componentes a esta configuración mínima y se prueba después de que se agrega cada incremento.

Aprendemos: Evaluación de Prototipos

Metodología de evaluación de prototipos

En la actualidad existen diversas metodologías para evaluar prototipos, comenzando por la Metodología para la Evaluación y Análisis de Viabilidad de Proyectos de Investigación presentada por (Marcelino-Jesus E. S.-G., 2016). Dicha metodología comprende las etapas mencionadas en la siguiente gráfica:



Aprendemos: Evaluación de Prototipos

Metodología de evaluación de prototipos

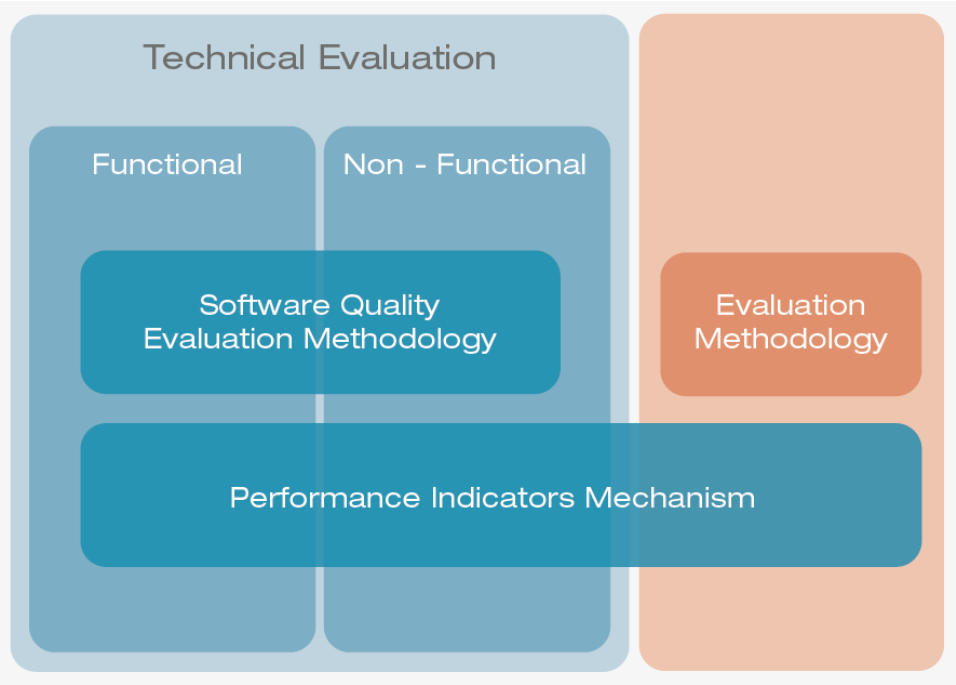
Como se puede observar, bajo mesta metodología se generan dos fases: la primera de ellas corresponde a una fase de validez conceptual y la segunda se refiere a una fase de validez técnica.
En la fase de **validez conceptual** se realiza (i) un estudio de variantes, con base en el cual se logra un primer acercamiento a la definición de la viabilidad técnico-comercial de la idea del proyecto, (ii) un análisis del negocio, en virtud del cual se definen los métodos de evaluación de la empresa y (iii) finalmente se toma una decisión, estableciendo si se debe reformular o no el proyecto, además, en este punto, debe considerarse que, si el proyecto no es viable, se debe abandonar o se debe reformular.
Por su parte, en la fase de **validez técnica** se deben adelantar las siguientes actividades, habiéndose definido previamente la existencia de un resultado u objeto de investigación en concreto: (i) paso de implementación, consiste en la puesta en marcha de la idea del proyecto, siendo necesario desarrollar la solución propuesta en la fase conceptual y, a su vez, evaluar/analizar la tecnología y sus resultados, y establecer si la misma se debe reformular o no; (ii) paso de exploración, durante el cual se examina la tecnología y se registran los resultados de la misma y (iii) paso de control o monitoreo, en el que el proyecto regresa a la exploración para ser reformulado o avanzar hacia una fase de evaluación posterior.
Finalmente, con base en esta metodología se lleva a cabo una última fase, denominada evaluación “ex-post” durante la cual también se puede generar la reformulación de la idea. Es importante precisare que para poder sostener que existe una validez técnica de los resultados del proyecto, estos mismos pueden estar en la etapa de exploración.

Aprendemos: Evaluación de Prototipos

Evaluar Prototipos de Software

Expuesta en un documento titulado “Un acercamiento para Evaluar Prototipos de Software”: desarrollada y aplicada por la NASA (Church, 1986) la cual menciona que dicha evaluación se debe adelantar en 3 pasos:

1. Definición de los criterios de evaluación.
2. Identificación de las alternativas de diseño.
3. Evaluación de las alternativas.



Aprendemos: Evaluación de Prototipos

1. Definición de los criterios de evaluación

En este punto se deben establecer aproximadamente **10 criterios**, todos basándose en la percepción que tendrían los clientes con respecto al problema que se desea resolver. Cada criterio deberá incluir una descripción, de tal forma que quienes participen respondiendo la evaluación comprendan con claridad qué es lo que se busca.

Aprendemos: Evaluación de Prototipos

2. Identificación de las alternativas de diseño

Con este paso se busca que el prototipo pueda ser evaluado desde distintas perspectivas y enfoques, proporcionando alternativas diferentes.
La revisión de tales alternativas, que en principio pueden parecer distantes, permite que se aumente la confianza en la evaluación y probablemente proporcionará información útil para el eventual proceso de desarrollo.

Aprendemos: Evaluación de Prototipos

3. Evaluación de las alternativas

Luego que se establezcan los criterios de evaluación y las alternativas, se puede comenzar la etapa de evaluación, siendo importante realizar análisis individuales, los cuales también pueden conducir a decisiones grupales. En este punto se realiza una evaluación comparativa de las alternativas dentro de cada criterio y considerando que el prototipo está disponible para inspección y que las alternativas son explicadas con claridad a los potenciales usuarios, el consenso se logra fácilmente. Con la información que se obtiene, se toman decisiones sobre el desarrollo del Sistema en su totalidad.

Aprendemos: Evaluación de Prototipos

Evaluación de Prototipos (Cabaj, 2019)

Finalmente, y con enfoque hacia el emprendimiento, la metodología creada por Tamarack Institute en Canadá (Cabaj, 2019), denominada **Evaluación de Prototipos**, la cual se basa en los siguientes pasos, los cuales son iterativos:

- **Confirmación del prototipo:** En este punto se define si en efecto existe un prototipo, acorde con la definición previamente dada, también se determina cuáles son las partes del prototipo que se desea evaluar y de igual forma se establece la clase de prototipo frente a la que nos encontramos: si este es un prototipo rápidamente obtenible o si se requiere elaborar lentamente pues para ello se deben generar cambios culturales; si es un prototipo desechable o si es evolutivo e incremental.

Aprendemos: Evaluación de Prototipos

Evaluación de Prototipos (Cabaj, 2019) (Cont.)

- **Generación de preguntas:** al respecto se elaboran preguntas, determinando cuáles serán los indicadores para probar los prototipos. Estas preguntas recaen sobre el prototipo, con miras a establecer su efectividad, viabilidad, soporte y escalabilidad, también recaen sobre el reto a superar, para lo cual se piensa en qué se está aprendiendo del reto y se analiza si el mismo se puede extender a otro contexto y, a su vez, se estudian la enseñanza que obtiene el equipo interviniente en la generación del prototipo, en el sentido de definir si, por ejemplo, es un equipo adverso al riesgo o si tiene tolerancia a él.

Aprendemos: Evaluación de Prototipos

Evaluación de Prototipos (Cabaj, 2019) (Cont.)

- **Diseño del método de evaluación:** para evaluar un prototipo se deben tener en cuenta principios tales como el de relevancia, de tal manera que se respondan preguntas que den lineamientos y permitan que el equipo que está creándolo pueda realizar ajustes; también se debe aplicar el principio de credibilidad, para que el equipo que está creando el prototipo sienta confianza y tenga en cuenta los resultados de la retroalimentación, por último, se deben aplicar los principios de oportunidad y calidad, de tal forma que los resultados se proporcionen en tiempo real y puedan realizarse ajustes sobre el prototipo y que las evaluaciones cumplan con estándares de calidad.

Aprendemos: Evaluación de Prototipos

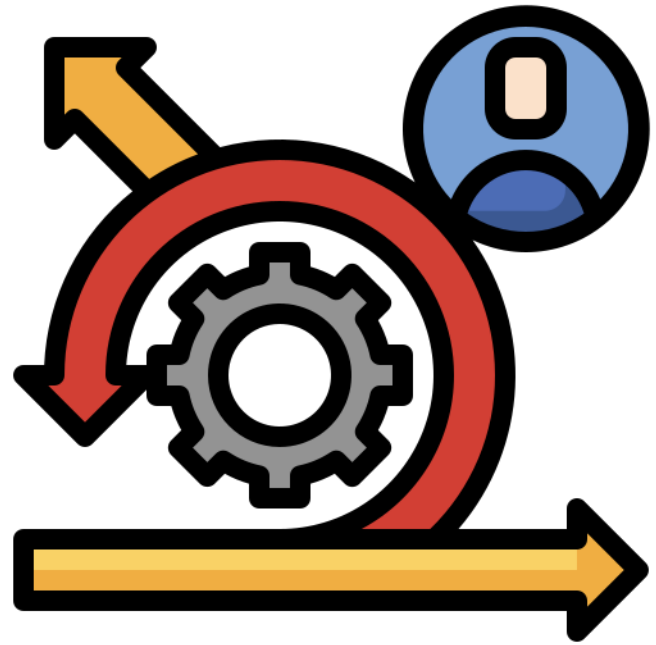
Evaluación de Prototipos (Cabaj, 2019) (Cont.)

- **Implementación y adaptación:** es importante tener en consideración que en esta etapa se pueden realizar ajustes sobre el método de evaluación, si por ejemplo se detecta que las preguntas no están profundizando sobre el prototipo o si, por ejemplo, el prototipo evoluciona.
- **Toma de decisiones:** una vez tienen lugar las anteriores etapas, el equipo puede tomar alguna de las siguientes decisiones: descartar el prototipo (en caso de que los resultados sean desfavorables y se determine que no vale la pena continuar invirtiendo en él); evolucionar y adaptar el prototipo (adaptándolo a un nuevo contexto); pasar a una fase piloto (si se determina que el prototipo se debe probar en un contexto más global); realizar el escalamiento (en cuyo caso se desarrollará la capacidad de operar con volúmenes significativos comercialmente y costos competitivos); continuar en las pruebas (si los resultados de la evaluación no permiten tomar una decisión).

Aprendemos: Evaluación de Prototipos

Importancia del establecimiento de una metodología para evaluar prototipos

Siempre que se desarrolle un nuevo producto, proceso, servicio o modelo de negocio, este debe pasar por un proceso de evaluación el cual se adelanta durante el diseño, construcción y transferencia de un prototipo innovador. Generalmente la evaluación se lleva a cabo a través de una metodología la cual permite determinar, mediante la aplicación de una serie de variables, qué requerimientos técnicos y de mercado se deben superar para disminuir los riesgos y garantizar la viabilidad de transferencia de una innovación.



Aprendemos: Evaluación de Prototipos

Importancia del establecimiento de una metodología para evaluar prototipos (Cont.)

La aplicación de una metodología de evaluación de prototipos permitirá a los emprendedores e investigadores garantizar el diseño, mejoramiento, entendimiento y adecuada transferencia de una innovación. Como punto de partida para aplicar una metodología de evaluación de prototipos es necesario caracterizar el mismo, definiendo qué problema resuelve, cuáles son sus características técnicas, ventajas, diferenciales, estado de desarrollo, quiénes son sus clientes potenciales, cuál es el mercado a impactar, entre otros; esta evaluación siempre se debe realizar de cara a las necesidades del usuario o beneficiario del prototipo a transferir, por tanto, desde que se concibe este debe ser testeado y pivoteado en el mercado.

Aprendemos:

Entendimiento y caracterización de un prototipo a evaluar

En un proceso de innovación es importante entender muy bien qué producto, proceso, servicio o modelo de negocio se quiere transferir o llevar al mercado, para lo cual los emprendedores o innovadores deben realizar un ejercicio de caracterización y entendimiento de la tecnología y del mercado.

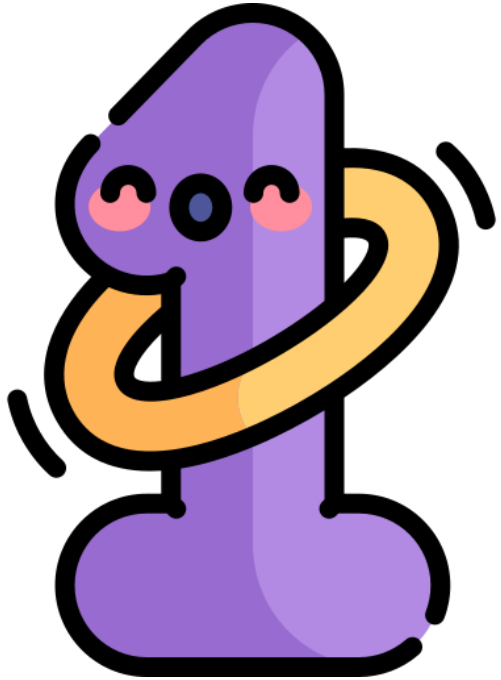
Se recomienda que este ejercicio se realice a través de la aplicación de una serie de preguntas que permiten medir alrededor de 5 variables, que se describen a continuación:

Aprendemos:

Entendimiento y caracterización de un prototipo a evaluar

Entender el prototipo innovador, aclarando de modo directo en qué es utilizado, cuál es su modo de funcionamiento. La caracterización y descripción debe evitar el uso de términos técnicos, para facilitar la apropiación del mismo en los mercados potenciales. Las preguntas que se pueden emplear en esta etapa son:

- ¿Cuál es la función del prototipo?
- ¿Qué hace el prototipo?
- ¿Qué problema resuelve?
- ¿Por qué ese es un problema?
- ¿Cuáles son los beneficios del prototipo?
- ¿Existen diferencias frente el estado del arte?



El estado del arte proviene originalmente del campo de la investigación técnica, científica e industrial y significa, en pocas palabras, la situación de una determinada tecnología.

Aprendemos:

Entendimiento y caracterización de un prototipo a evaluar

Describir brevemente las principales aplicaciones del prototipo, es decir, para qué puede ser utilizado, en qué mercado puede ser aplicado, qué nuevas formas de aplicación pueden ser identificadas, precisando que, si el prototipo cuenta con muchas aplicaciones, se recomienda usar viñetas y gráficos para que la información quede más explícita. Para este punto se sugiere aplicar preguntas como:

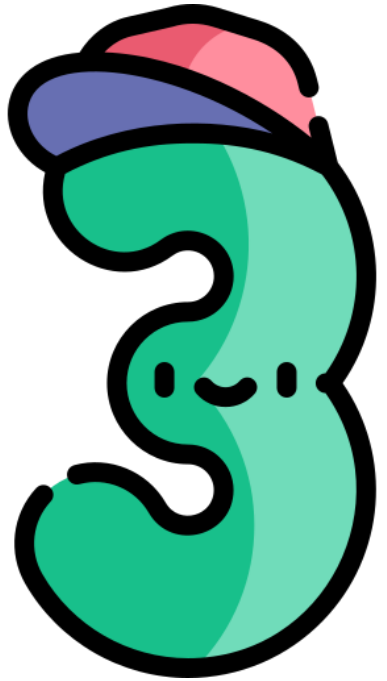
- ¿Cuáles son las posibles aplicaciones de la tecnología?
- ¿Estas aplicaciones ya fueron testeadas con el mercado?
- ¿Qué otras aplicaciones pueden darse y en que segmentos de mercado?

Aprendemos: Evaluación de Prototipos

Entendimiento y caracterización de un prototipo a evaluar

Describir cuales son las soluciones ya existentes en el mercado (según el mercado que se haya definido para la exploración. Para este punto se debe tener en cuenta que hay soluciones similares y sustitutas y que ambas pueden estar en el espectro de competencia del prototipo que se está evaluando. Para este punto se sugiere aplicar preguntas como:

- ¿Cuáles son las soluciones competidoras?
- ¿Qué empresas tienen estas soluciones competidoras?



Aprendemos:

Entendimiento y caracterización de un prototipo a evaluar

Describir los beneficios que ofrece el prototipo innovador al cliente objetivo, aclarando que el beneficio se genera desde la perspectiva del cliente. Posteriormente y basado en la información de los productos similares y sustitutos se deben identificar los diferenciales, los cuales se miden desde el punto de vista del desarrollador del prototipo innovador. Para ello se sugiere realizar preguntas como:

- ¿Cuáles son los beneficios y/o diferenciales de la tecnología?,
- ¿Estos diferenciales se han validado en entornos reales de operación?
- ¿El cliente percibe estos beneficios de manera real?

Aprendemos:

Entendimiento y caracterización de un prototipo a evaluar

Identificar las actividades que deben realizarse y los recursos necesarios para definir los próximo pasos y desafíos tecnológicos, detectando los resultados logrados en el desarrollo del prototipo, independientemente del estado de desarrollo.

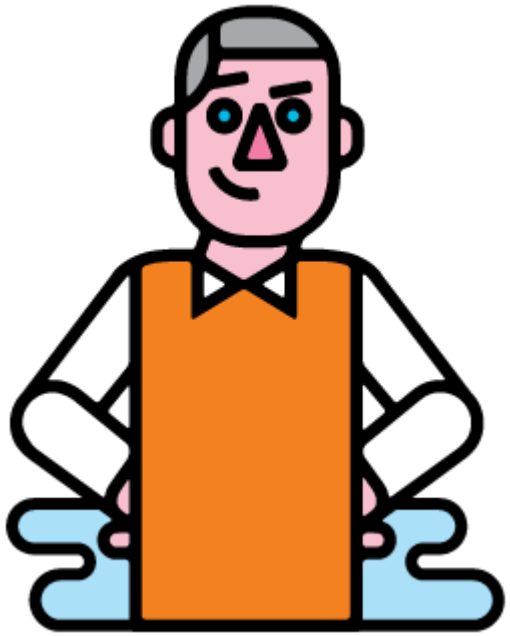
Para ello se sugiere aplicar preguntas como:

- ¿Qué falta para que el prototipo pueda estar disponible en el mercado?
- ¿Cuáles son las próximas pruebas y/o análisis a ser realizados?
- ¿Qué pruebas son necesarias para comprobar el funcionamiento/adherencia del prototipo?
- ¿Cuáles son los puntos críticos para tener un “prototipo” (uso de mercado)?
- Señale qué puede impedir o dificultar el uso del prototipo en el mercado.

Aprendemos:

Entendimiento y caracterización de un prototipo a evaluar

La metodología antes descrita se aplica generalmente al desarrollador o inventor del prototipo, sin embargo, es de suma importancia que las respuestas a esta serie de preguntas sean validadas con el cliente final o beneficiario del prototipo. El entendimiento y caracterización sirve como base para la construcción de la primera hipótesis de la propuesta de valor, la cual será testeada con el cliente final, así mismo, permite la construcción de las fichas de marketing que se convertirán en un instrumento para probar y validar con el potencial cliente la propuesta de valor, los diferenciales y las ventajas del prototipo innovador.



Aprendemos:

Entendimiento y caracterización de un prototipo a evaluar

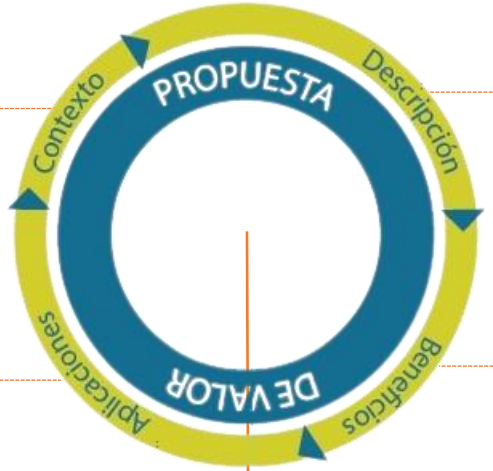
Dichas fichas son una herramienta para un primer acercamiento del prototipo con el sector o cliente de interés. Su finalidad es generar una primera conexión que resulte en una acción de relacionamiento y por ello debe ser contundente en mostrar el valor para que el cliente se interese en la posible innovación. Las fichas de marketing deben cumplir con una serie de características como:

- Debe ser un documento práctico y concreto.
- Debe establecer a quién se está dirigiendo para saber cómo orientar el prototipo, dependiendo del tipo de cliente, su orientación y construcción.
- Identificar cuáles son las necesidades del cliente que se quiere contactar, las cuales respondan directamente al prototipo desarrollado.
- Establecer cuál es el modelo de transferencia más indicado para el prototipo. Por ejemplo, si se ofrecerá como un servicio tecnológico, licenciamiento, cooperación tecnológica, spin off, entre otros.
- Debe contar con 5 secciones principales, como, por ejemplo, problema que resuelve el prototipo innovador, descripción técnica, propuesta de valor, aplicaciones y beneficios.

Aprendemos: Entendimiento y caracterización de un prototipo a evaluar

Ficha de marketing de un desarrollo tecnológico

Contexto
Es necesario redactar el problema u oportunidad que soluciona o aprovecha el prototipo innovador. Se debe plantear un contexto breve y destacar los problemas y dificultades de las principales soluciones que abordan el problema, o en el caso de la oportunidad, la magnitud de la brecha existente. Es necesario dejar claro el tamaño del problema u oportunidad que se quiere explotar. Por ejemplo, se han de definir los ingresos, eficiencia o rendimiento, costos, tasas de enfermedad, etc.



Descripción
En este punto, se debe hacer la presentación del prototipo innovador, usando términos de fácil comprensión, para indicar concretamente lo que tiene o lo que hace que se resuelva el problema identificado. Se deben destacar las pruebas/comparaciones que validen la superioridad del prototipo innovador con respecto a otras soluciones (que conecten directamente con el problema planteado).

Aplicaciones
En este punto se debe describir las principales aplicaciones del producto tecnológico o el sector de aplicación dónde esta puede ser utilizada. De acuerdo con el tipo de prototipo innovador y su contexto, se puede ser más o menos específico. Por ejemplo, para una enzima desarrollada en laboratorio se podrían identificar diferentes aplicaciones tecnológicas como 1. depuración 2. síntesis de otras moléculas o 3. destrucción de Radicales libres

La propuesta de valor es el corazón de la ficha y lo que primero capta la atención del cliente o público objetivo por lo cual su mensaje debe ser contundente. Debe ser una oración clara, concisa y que demuestre la esencia principal del prototipo innovador. La propuesta de valor es el enunciado de los beneficios que pueden esperar los clientes de una innovación.

Beneficios
Es necesario redactar el problema u oportunidad que soluciona o aprovecha el prototipo innovador, planteando un contexto breve y destacando los problemas/dificultades de las principales soluciones que abordan el problema o en el caso de la oportunidad, la magnitud de la brecha existente.
Es necesario dejar claro el tamaño del problema u oportunidad que se quiere explotar.

Aprendemos:

Niveles de pruebas en sistemas orientados a objetos

- En un sistema orientado a objetos se pueden identificar cuatro niveles de prueba:
    - Probar las operaciones individuales asociadas a los objetos.
    - Probar clases de objetos individualmente.
    - Probar grupos de objetos.
    - Probar el sistema.
- Para probar una clase se requiere:
    - Pruebas aisladas de cada operación de su interfaz.
    - La verificación de los valores que se asignan a los atributos de las clases.
    - La ejecución de los objetos en todos los estados posibles en la clase.

Aprendemos:

Niveles de pruebas en sistemas orientados a objetos

Las técnicas de pruebas para sistemas orientados a objetos han madurado rápidamente y hoy en día existe una gran cantidad de información disponible sobre ellas.
En un sistema orientado a objetos se pueden identificar cuatro niveles de prueba:

- **Probar las operaciones individuales asociadas a los objetos:** Es pruebas de funciones y se pueden utilizar las técnicas de caja negra y blanca.
- **Probar clases de objetos individualmente:** Dado que las diferentes funciones de una clase están interrelacionadas requiere un método especial.
- **Probar grupos de objetos:** En este caso nos son apropiadas las integraciones descendente o ascendente. Se requieren nuevos enfoques.
- **Probar el sistema:** La verificación y validación es la comparación con la especificación de requisitos disponibles, y se realiza igual que en el caso funcional estructurado.

Aprendemos:

Herramienta JUnit

- JUnit es un conjunto de clases que permite desarrollar y ejecutar conjuntos de pruebas de forma sencilla y sistemática.
  - Está integrada en el entorno Eclipse
  - Orientada a pruebas unitarias (prueba de clases en Java)
- Los casos de prueba que se generan son clases Java, que pueden ser almacenadas y re-ejecutadas cuando sea necesario
  => Se generan bancos de pruebas asociados al proyecto
- Configuración del proyecto para hacer uso de JUnit:
  - Botón derecho en el proyecto => *Configure Build Path* => *Add Library* => *seleccionar JUnit* => *next* => *Elegir JUnit 3.8.1*

Aprendemos:

Elementos del framework JUnit

- **Casos de prueba (Test Case):** Clases que incluyen una serie de métodos para probar los métodos de una clase concreta. Por cada clase que queramos crear podemos crear una clase de prueba
- **Métodos de prueba:** Son los métodos que la herramienta va a ejecutar automáticamente al ejecutar la clase de prueba.
- **Aserciones (Asserts):** Las pruebas en JUnit se basan en programación con aserciones
- **Grupos de pruebas (Test Suite):** Sirven para agrupar varias clases de prueba, y poder ejecutarlas todas seguidas.

Aprendemos:

Creación de una clase de prueba (Test Case)

- Pulsar con el botón derecho sobre la clase a probar y elegir New => Other => Java => JUnit => JUnit Test Case
- En la ventana que aparece a continuación marcar que deseamos que se cree el método setUp() y pulsar next
- Seleccionar los métodos que queremos probar y pulsar Finish
- Se crea la clase de prueba, de nombre *<NombreClase>Test*, que extiende a *junit.framework.TestCase*
- La clase incluye un método de prueba por cada método que se quiere probar, de nombre *test<nombreMetodo>*
- Podemos añadir más métodos de prueba (también deberán comenzar por la palabra “test”)

Aprendemos:

Ejemplo: Clase de prueba para la clase Moda

```
import junit.framework.TestCase;
public class ModaTest extends TestCase {
protected void setUp() throws Exception {
super.setUp();
}
public void testNuevoDato() {
fail("Not yet implemented");
}
public void testModa() {
fail("Not yet implemented");
}
public void testNumVecesValor() {
fail("Not yet implemented");
}
public void testNumDatos() {
fail("Not yet implemented");
}
```

Aprendemos:

Métodos de prueba

- Cuando se ejecute la clase de prueba, todos los métodos de prueba son ejecutados automáticamente.
- Debemos escribir en ellos los casos de prueba que queramos verificar.
- En general, la construcción de pruebas sigue siempre el mismo patrón:
    - Construcción de los objetos de prueba: El método ***setUp()*** de la clase de prueba se ejecuta antes de cualquier método de prueba, por lo que se emplea para inicializar variables u objetos en el estado que nos interese para cada prueba.
    - Verificación de propiedades de la clase a través de aserciones => Permiten verificar el correcto funcionamiento de los métodos invocados.
- La ejecución de un método de prueba puede tener tres resultados posibles:
    - Correcto: Finaliza sin producirse ningún fallo
    - Failure: Se ha producido un fallo esperado, de los que estamos comprobando a través del caso de prueba. Se lanza la excepción *AssertionFailedError*
    - Error: Se ha producido un error no esperado

Aprendemos:

Aserciones

- Una aserción es una sentencia que nos permite probar una proposición que debería ser siempre cierta de acuerdo con la lógica del programa.
    - Ejemplo: Si invoco el método clear() en una lista (elimina todos los elementos)
        - Aserción de comprobación: list.size() == 0
- Cada aserción evalúa una expresión booleana que se supone que será cierta si el programa ejecuta correctamente
    - Si no es cierta, el sistema lanza una excepción, poniendo de manifiesto un error en el programa
- Uso de aserciones en JUnit, a través de estos métodos:
    - void assertTrue (String msj, Boolean cond)
        - Si cond es evaluada a true no pasa nada
        - Si cond es evaluada a false, se lanza la excepción AssertionFailedError y se muestra el mensaje msj.
        - Ej de la lista
            - assertTrue(“Error en clear”, list.size()==0);
    - void fail(String msj): Lanza siempre la excepción AssertionFailedError y muestra el mensaje msj. (Para mostrar fallos en excepciones)
        - Equivalente a assertTrue (msj, false);

Aprendemos:

Validación de prototipos de los Requerimientos

¿Dudas?

¿Miedos?

¿Temores?

Notas

Aplicamos lo aprendido: Asignación para la clase siguiente

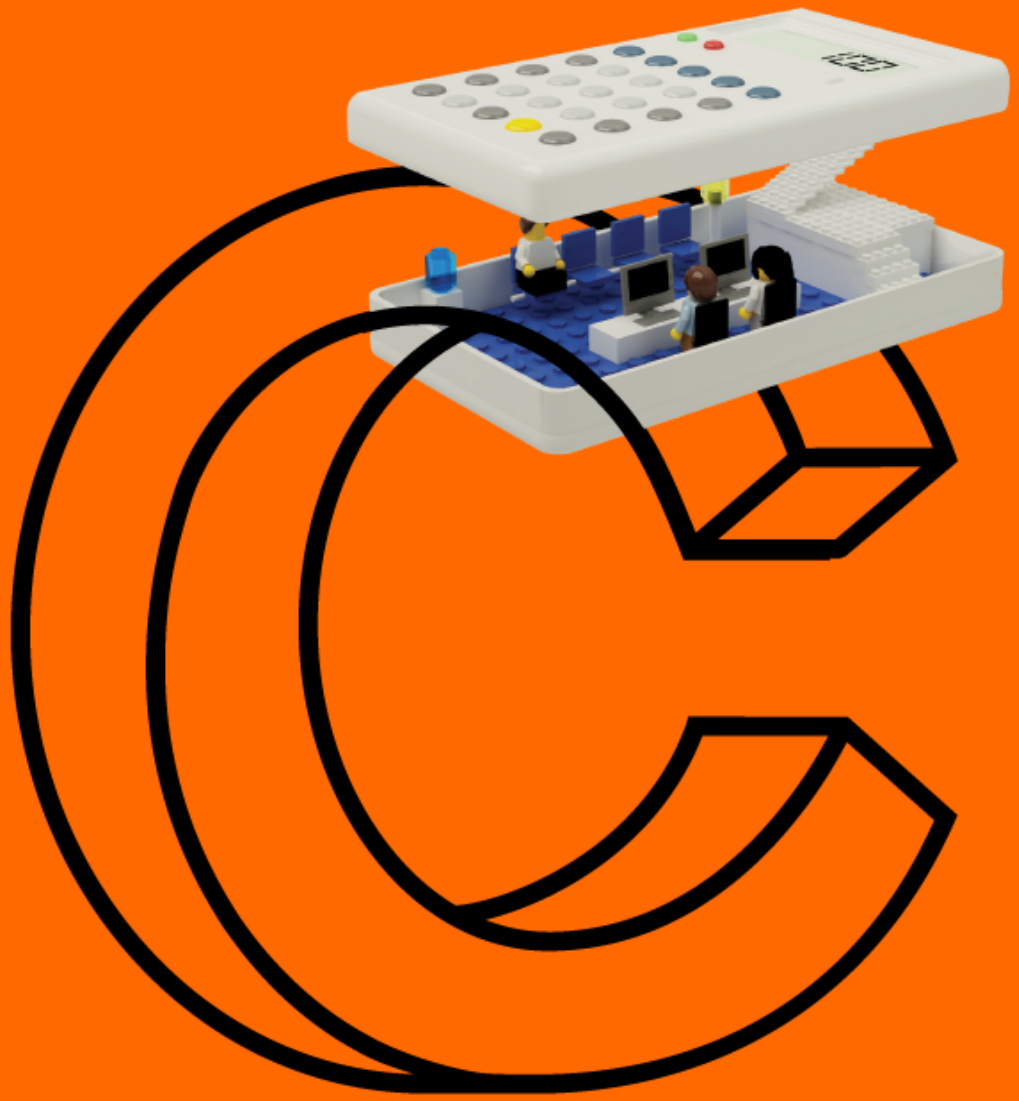
- Revisión de los trabajos (avance)

Verificamos lo aprendido:

Kahoot!

http://www.kahoot.it/

Gracias



idat