



# TALLER DE TESTING Y CALIDAD DE SOFTWARE

*Semana 3*





## **ESCUELA DE CONSTRUCCIÓN E INGENIERIA**

**Director:** Marcelo Lucero

### **ELABORACIÓN**

**Experto disciplinar:** Aída Villamar Gallardo.

**Diseño instruccional:** Carla Silva Alvarado.

**Editor instruccional:** David Villagrán.

### **VALIDACIÓN**

**Experto disciplinar:** Andrés del Alcázar

**Jefa de Diseño Instruccional:** Alejandra San Juan Reyes.

### **EQUIPO DE DESARROLLO**

AIEP

**AÑO**

2021



## Tabla de contenidos

<b>Aprendizaje esperado</b> .....	4
<b>Introducción</b> .....	4
<b>1. Identificación de procesos y técnicas de verificación y validación, considerando documentación de pruebas asociadas, según estándares internacionales</b> .....	5
1.1. Verificación y validación de software .....	5
1.2. Técnicas de validación y verificación de software .....	6
1.3. Estándares internacionales .....	9
1.4. Documentación de las pruebas de sistema .....	11
<b>2. Caracterización de problemas y errores frecuentes en proyecto de desarrollo de software</b> .....	12
2.1. Problemas frecuentes y errores en desarrollo de software .....	12
<b>3. Relación de estándares con proceso de verificación y validación, considerando estándar ISO/IEC y estándar IEEE</b> .....	14
3.1. Estándar ISO/IEC .....	14
3.2. Estándar IEEE.....	17
<b>4. Aplicación de la verificación y validación de procesos en proyectos de desarrollo de software, considerando análisis objetivo</b> .....	19
4.1. Estudio de casos: verificación y validación de software y procesos. ....	19
4.2. Aplicación de verificación y validación de procesos en proyectos reales externos.....	21
4.3. Análisis objetivo .....	23
<b>Conclusiones</b> .....	25
<b>Referencias bibliográficas</b> .....	26



## Aprendizaje esperado

Aplican verificación y validación de procesos en proyectos de desarrollo de software, considerando técnicas asociadas, de acuerdo con estándares internacionales.

## Introducción

- ¿Por qué los problemas de software no son detectados?
- ¿Qué hacemos para descubrir defectos en nuestros desarrollos?
- Para evaluar la calidad de nuestros productos, ¿cómo lo hacemos?

Estas son preguntas que debemos hacernos al momento de realizar nuestros desarrollos de software. Las respuestas, las estamos descubriendo a medida que nos vamos interiorizando en los conceptos asociados a calidad y métodos para establecerlos.

En este documento, veremos que la verificación, la validación y estándares internacionales juegan un rol muy importante, tan importante que si no los consideramos estamos afectando tanto el proceso como el resultado de nuestros proyectos.



# 1. Identificación de procesos y técnicas de verificación y validación, considerando documentación de pruebas asociadas, según estándares internacionales.

## 1.1. Verificación y validación de software

Para comenzar, revisaremos los conceptos de **verificación** y **validación**, que en las semanas anteriores estudiamos, pero que ahora veremos más en profundidad.

Dentro de todo el ciclo de desarrollo de software, encontramos actividades de verificación y validación que nos permiten reducir la cantidad de defectos. Sin embargo, lo que no podemos afirmar ni demostrar es que un software esté libre de ellos.

Establecimos que 'Verificación y Validación' (V&V) se define como un conjunto de procesos que nos permiten **comprobar y analizar si el desarrollo del software está de acuerdo con su especificación**, pero que, además, **cumple con las necesidades de los clientes**.

Además de lo anterior, podemos de manera independiente ver el significado de cada concepto por separado.

Podemos comenzar con la **verificación**, que está asociada a la pregunta: **¿estamos construyendo el producto correctamente?**, es decir, comprobamos que el software cumple los requisitos funcionales y no funcionales de su especificación.



Por otro lado, tenemos la **validación**, asociada a la pregunta: **¿estamos construyendo el producto correcto?** Por medio de la cual comprobamos que el software cumple las expectativas que el cliente espera.

Además, como plantean Drake y López (2009), es fundamental llevar a cabo la validación de los requerimientos del sistema de manera inicial, ya que es muy posible cometer errores u omisiones durante la fase de análisis de requerimientos del sistema; es decir, que definamos requerimientos incompletos o incorrectos, lo que va a generar que el software final no va a cumplir con las expectativas de los clientes.


No obstante, debemos considerar que la validación de los requerimientos no puede descubrir todos los problemas que presenta la aplicación y que algunos defectos en los requerimientos solo es posible descubrirlos cuando la implementación del sistema se encuentra completa, como hemos visto anteriormente lo que hacemos es **disminuirlos**.

## 1.2. Técnicas de validación y verificación de software

Dentro del proceso de validación y verificación, de acuerdo a Drake y López (2009), encontramos dos técnicas:

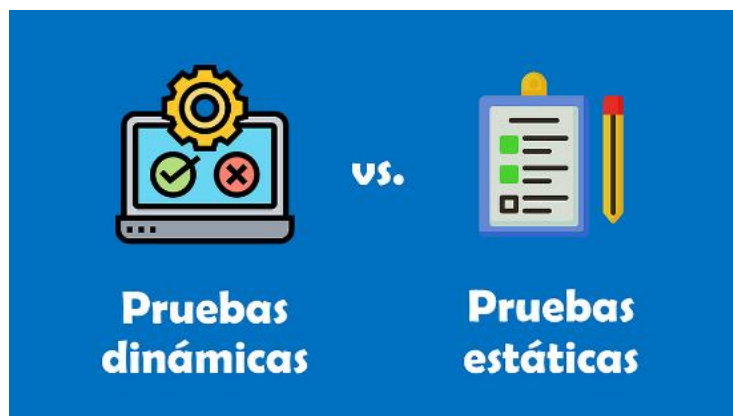
- Comprobación.
- Análisis de sistemas.

---



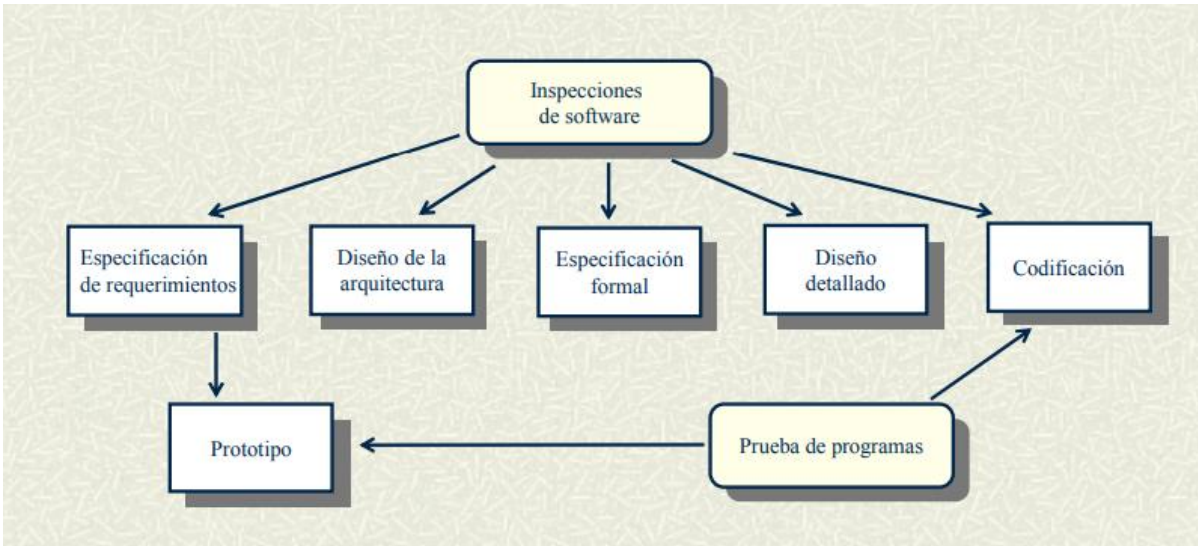
**Las inspecciones del software** analizan y comprueban las representaciones del sistema como el documento de requerimientos, los diagramas de diseño y el código fuente del programa. Se aplica a todas las etapas del proceso de desarrollo. Las inspecciones se complementan con algún tipo de análisis automático del texto fuente o de los documentos asociados. Las inspecciones del software y los análisis automatizados son técnicas de verificación y validación estáticas puesto que no requieren que el sistema se ejecute.

**Las pruebas del software**, en tanto, consisten en contrastar las respuestas de una implementación del software a series de datos de prueba y examinar las respuestas del software y su comportamiento operacional, para comprobar que se desempeñe conforme a lo requerido. Llevar a cabo las pruebas es una técnica dinámica de la verificación y validación ya que requiere disponer de un prototipo ejecutable del sistema. (p. 4)



**Figura 1.** Pruebas dinámicas y pruebas estáticas

**Fuente:** Gómez (2021)



**Figura 2.** Verificación y validación estática y dinámica

**Fuente:** Drake y López (2009, p. 5)

Seguendo a Drake y López (2009),

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 usar 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. (p. 5).





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

### 1.3. Estándares internacionales

Existen diversas metodologías, técnicas y herramientas actuales de ingeniería de software a nivel mundial que ayudan a mejorar la calidad de los productos de software desarrollados. Por ejemplo, está el Programa de Certificación de Pruebas de Software (CSTE, Certified Software Tester), los estándares de ingeniería de software del Institute of Electrical and Electronics Engineers (IEEE), el Modelo de Capacidad/Madurez (CMM) y el estándar ISO 9001.

De acuerdo a Sites.google.com. (s. f.),

El objetivo general de la creación del estándar ISO/IEC 25000 SQuaRE (System and Software Quality Requirements and Evaluation) es organizar, enriquecer y unificar las series que cubren dos procesos principales: especificación de requisitos de calidad del software y evaluación de la calidad del software, soportada por el proceso de medición de calidad del software.

Las características de calidad y sus mediciones asociadas pueden ser útiles no solamente para evaluar el producto software sino también para definir los requerimientos de calidad. La serie ISO/IEC 25000:2005 reemplaza a dos estándares relacionados: ISO/IEC 9126 (Software Product Quality) e ISO/IEC 14598 (Software Product Evaluation).



**ISO/IEC 2500n.** División de gestión de calidad. Los estándares que forman esta división definen todos los modelos comunes, términos y referencias a los que se alude en las demás divisiones de SQuaRE.

**ISO/IEC 2501n.** División del modelo de calidad. El estándar que conforma esta división presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso.

**ISO/IEC 2502n.** División de mediciones de calidad. Los estándares pertenecientes a esta división incluyen un modelo de referencia de calidad del producto software, definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación. Presenta aplicaciones de métricas para la calidad de software interna, externa y en uso.

**ISO/IEC 2503n.** División de requisitos de calidad. Los estándares que forman parte de esta división ayudan a especificar los requisitos de calidad. Estos requisitos pueden ser usados en el proceso de especificación de requisitos de calidad para un producto software que va a ser desarrollado o como entrada para un proceso de evaluación. El proceso de definición de requisitos se guía por el establecido en la norma ISO/IEC 15288 (ISO, 2003).

**ISO/IEC 2504n.** División de evaluación de la calidad. Estos estándares proporcionan requisitos, recomendaciones y guías para la evaluación de un producto software, tanto si la llevan a cabo evaluadores, como clientes o desarrolladores.

**ISO/IEC 25050–25099.** Estándares de extensión SQuaRE. Incluyen requisitos para la calidad de productos de software “Off-The-Self” y para el formato común de la industria (CIF) para informes de usabilidad.



## 1.4. Documentación de las pruebas de sistema

Durante el desarrollo de un sistema, en cada una de sus etapas hay un proceso de documentación asociado, en el caso de las pruebas de sistemas no es la excepción.

La documentación debe reflejar las entradas, como se cumplieron los objetivos y evidenciar que con los resultados obtenidos se cumplieron los objetivos esperados. Las pruebas son especialmente importantes ya que son la manera de demostrar de que lo que se ha implementado cumple con los requerimientos.

Las buenas practicas de la documentación, señalan que en primer lugar debe estar bien organizada y diseñadas para garantizar el control de la creación, revisión, aprobación, distribución y almacenamiento de esta.

Para la documentación se requieren algunos puntos, dentro de ellos podemos destacar algunos importantes:

- Resultados de la evaluación peligros y riesgos.
- Suposiciones usadas cuando se determinaron los niveles de integridad de seguridad.
- Especificaciones de requisitos de seguridad (SRS)
- Lógica de la aplicación incluyendo los módulos certificados usados.
- Registros de verificación y validación
- Procedimientos de pruebas de aceptación
- Información y/o documentación de las modificaciones
- Resultados de evaluaciones y auditorias



En general la documentación no se considera como algo prioritario, pero es un elemento muy importante en el desarrollo de un sistema y ayudará a definir ajustes necesarios o posibles mejoras.

## 2. Caracterización de problemas y errores frecuentes en proyecto de desarrollo de software

### 2.1. Problemas frecuentes y errores en desarrollo de software

En el libro *Rapid Development* (1996), Steve McConnell reúne los errores más frecuentes dentro del proceso de desarrollo de software.

El listado inicia con 36 errores desde la publicación del libro en el año 1996, y por medio de frecuentes actualizaciones se han ido agregando nuevos errores.

Dentro de los primeros diez errores más frecuentes se identificó los primeros cinco como: “Casi siempre” y los sub-siguientes como: “a menudo” son:


1. **Cronogramas demasiado optimistas:** Planificar un proyecto para dos meses, cuando en realidad abarcará un año.
2. **Expectativas irreales:** Pedirle a un proyecto algo imposible.
3. **Aseguramiento de calidad ínfimo.**



4. **Oficinas ruidosas y hacinadas:** en un entorno ruidoso o demasiado hacinado se evita la concentración y el estado necesario que se requiere para trabajar y obtener altos niveles de productividad.
5. **Confusión de estimados con objetivos:** el objetivo es tener el software en 3 meses, y de ahí se fija que el desarrollo serán 3 meses.
6. **Excesiva aplicación de multitarea:** cuando, por ejemplo, los desarrolladores están en muchos proyectos a la vez.
7. **Pesadilla de características:** durante la codificación se añaden características que no se solicitaron por parte del cliente.
8. **Pensamiento iluso:** cuando la gestión del proyecto se cierra totalmente en la idea de que algo va a funcionar cuando no se tiene base concreta o razonable para pensar que así será.
9. **Gestión de riesgo insuficiente.**
10. **Omisión de tareas necesarias para estimados:** no guardar históricos para realizar mejores estimaciones, al estimar obviar tareas como son las reuniones, etc.

Entre ellos se asignó alto impacto con un nivel de criticidad "Catastrófico" y "serio" como se indica a continuación:

- Expectativas irreales (83%)
- Personal inadecuado (78%)
- Cronogramas demasiado optimistas (78%)
- Pensamiento iluso (76%)
- Aseguramiento de calidad ínfimo (72%)
- Diseño inadecuado (72%)

- 
- 
- Falta de auspicio del proyecto (71%)
  - Confusión de estimados con objetivos (71%)
  - Excesiva aplicación de multitarea (71%)
  - Falta de involucramiento del usuario (70%)

Resulta interesante observar que 35 de los 42 errores clásicos se clasificaron como de impacto catastrófico o serio por más del 50% de participantes.

### **3. Relación de estándares con proceso de verificación y validación, considerando estándar ISO/IEC y estándar IEEE.**

#### **3.1. Estándar ISO/IEC**

Instituciones y organizaciones como la IEEE, la ISO, la IEC ('Comisión Electrónica Internacional'), entre otras, se han centrado en la definición de estándares que apoyen la V&V y pruebas.

La versión más antigua de este estándar data de 1986, y describía el contenido del plan V&V para software, con las subsecuentes versiones (1998 y 2004) el enfoque cambia desde el plan V&V de software hacia los procesos de software V&V. Esta revisión expande el alcance de los procesos V&V para incluir sistemas y hardware, así como también software. Adicionalmente, se alinea con la terminología y estructura para ser consistente con ISO/IEC 15288:2008 e ISO/IEC 12207:2008.



Al igual que la definición de estándares ISO/IEC 29119 Software Testing, también se han creado centros para la 'Certificación en Pruebas de Software' del International Software Testing Qualifications Board (ISTQB), que permitirán en conjunto satisfacer las necesidades apremiantes de calidad del software, que se ha tratado de alcanzar desde los comienzos de la computación.

Para cumplir y desarrollar software de calidad, se han creado estándares internacionales como la IEEE Std 1012TM-2012 que define los procesos de V&V en términos de actividades específicas y tareas relacionadas. Estas series de normativas cambiantes se han ido incrementando, sustituyendo y mejorando, tal es el caso de: ISO/IEC 15288:2008 reemplaza al estándar IEEE/EIA Std 12207.0:1996. ISO/IEC 15288:2008 es el estándar para los procesos del sistema. ISO/IEC 12207:2008 es el estándar para los procesos de software.

En los procesos de V&V, deben considerarse los estándares de desarrollo establecidos como por ejemplo la IEEE Std 1012TM-2012., que abarcan:

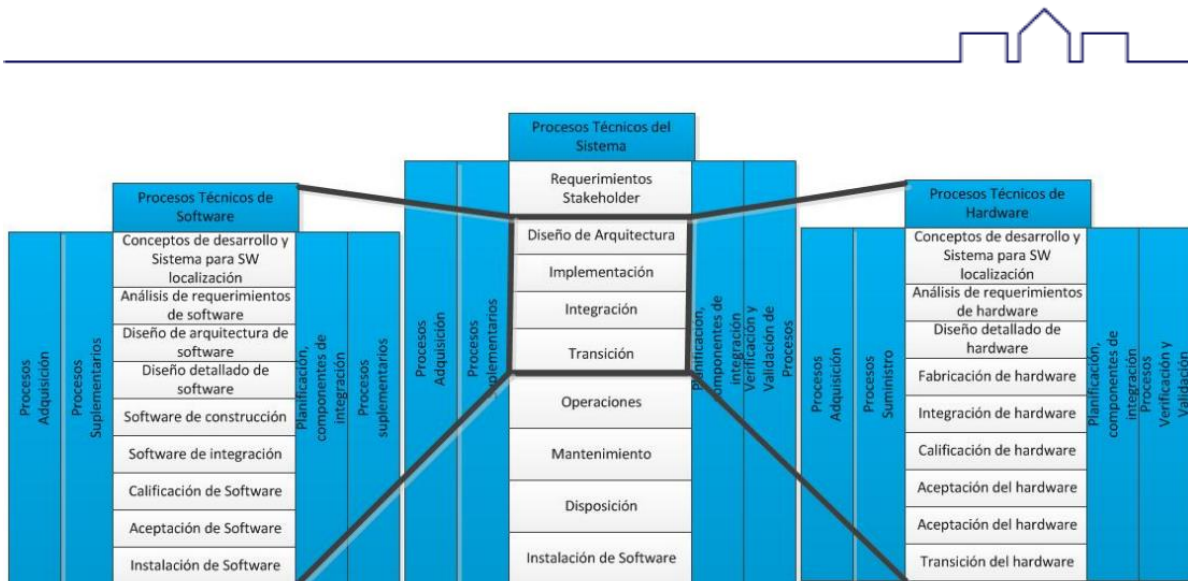
- a) Apoyo en adquisición V&V.
- b) Planificación de suministros V&V.
- c) Planificación del proyecto V&V.
- d) Administración de la configuración V&V.
- e) Definición de requerimientos del sistema (stakeholder) V&V.
- f) Sistema de análisis de requisitos V&V.



- g) Diseño de la arquitectura del sistema V&V, concepto de software V&V, concepto de hardware V&V.
- h) Implementación del sistema V&V, Actividades V&V de hardware y software.
- i) Diseño V&V, implementación/fabricación V&V, test de integración V&V, test de calificación V&V, test de aceptación V&V.
- j) Integración del sistema V&V.
- k) Todas las actividades del ciclo de vida del estándar IEEE 1012.
- l) Sistema de transición V&V, software de instalación y transición V&V, transición de hardware V&V.
- m) Todas las actividades de validación del ciclo de vida del estándar IEEE 1012.
- n) Funcionamiento del sistema V&V, funcionamiento de software V&V, funcionamiento de hardware V&V.
- o) Mantenimiento del sistema, hardware y software V&V.

El estándar ISO/IEC 12207:2008 adicionalmente tiene los siguientes procesos que corresponden a las actividades de implementación del sistema, hardware y software, como se indican en la figura 3, requisitos de software, análisis de procesos de software, proceso de diseño arquitectónico, proceso de diseño detallado de software, proceso de construcción de software, proceso de integración de software, proceso de calidad de pruebas del software.





**Figura 3.** V&V proceso de Sw, Sistema y Hw C. IEEE-1012, "IEEE std 1012 2012"

### 3.2. Estándar IEEE


Con respecto al estándar IEEE debemos considerar lo siguiente:

La visión del desarrollo de software como un conjunto de fases con posibles realimentaciones facilita la V&V.

Al inicio del proyecto es necesario hacer un plan de V&V del software (estructura del plan según el estándar IEEE 1012).

Las actividades de V&V se realizan de forma iterativa durante el desarrollo.

---



## **IEEE 1012-2004: IEEE Standard for Software Verification and Validation**

1. Propósito
2. Documentos de referencia
3. Definiciones
4. Visión general de la verificación y validación
  - 4.1 Organización
  - 4.2 Programa de tiempos
  - 4.3 Esquema de integridad de software
  - 4.4 Resumen de recursos
  - 4.5 Responsabilidades
  - 4.6 Herramientas, técnicas y metodologías
5. Verificación y validación en el ciclo de vida
  - 5.1 Gestión de la VV
  - 5.2 VV en el proceso de adquisición
  - 5.3 VV en el proceso de suministro
  - 5.4 VV en el proceso de desarrollo:
    - 5.4.1 VV de la fase de concepto
    - 5.4.2 VV de la fase de requisitos
    - 5.4.3 VV de la fase de diseño
    - 5.4.4 VV de la fase de implementación
    - 5.4.5 VV de la fase de pruebas
    - 5.4.6 VV de la fase de instalación
  - 5.5 VV de la fase de operación
  - 5.6 VV del mantenimiento
6. Informes de la VV del software
7. Procedimientos administrativos de la VV
  - 7.1 Informe y resolución de anomalías
  - 7.2 Política de iteración de tareas
  - 7.3 Política de desviación
  - 7.4 Procedimientos de control
  - 7.5 Estándares, prácticas y convenciones.
8. Requisitos de documentación para la VV

**Figura 4.** IEEE 1012-2004: IEEE Standard for Software Verification and Validation



## 4. Aplicación de la verificación y validación de procesos en proyectos de desarrollo de software, considerando análisis objetivo.

### 4.1. Estudio de casos: verificación y validación de software y procesos.

Debemos partir por establecer cuáles son los objetivos de la V&V.

En primer lugar, tenemos que uno de ellos es descubrir defectos para poder corregirlos y por otro lado evaluar la calidad de los productos.

Para descubrir los defectos, ¿qué hacemos?

- Provocamos fallas, ya que es una manera de poder detectarlos.
- Revisamos los productos, otra manera de detección.

Para evaluar la calidad de los productos, ¿qué hacemos?

- Probamos o revisamos el sistema, es una excelente manera de evaluar su calidad.

Una vez que los hemos detectado, corregimos y evaluamos nuevamente (pruebas de regresión).

A continuación, se presenta cada paso del proceso de un estudio de caso, de acuerdo a la publicación de Jiménez (2012).

1. **La selección y definición del caso:** se trata de seleccionar el caso apropiado y además definirlo. Se deben identificar los ámbitos en los que



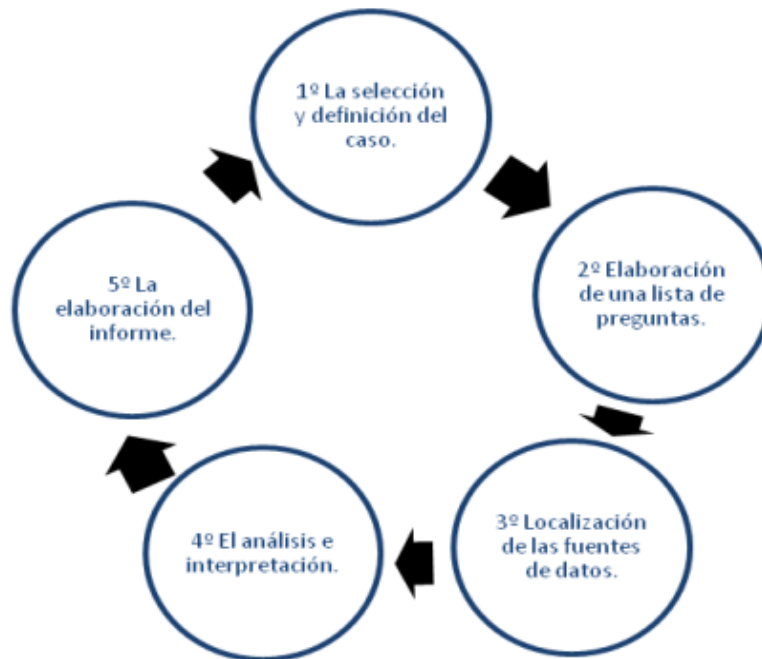
es relevante el estudio, los sujetos que pueden ser fuentes de información, el problema y los objetivos de investigación.

2. **Elaboración de una lista de preguntas:** después de identificar el problema, es fundamental realizar un conjunto de preguntas para guiar al investigador. Tras los primeros contactos con el caso, es conveniente realizar una pregunta global y desglosarla en preguntas más variadas, para orientar la recogida de datos, en el caso particular de un desarrollo de software, nuestros parámetros están enfocados a los requerimientos del sistema.

3. **Localización de las fuentes de datos:** los datos se obtienen mirando, preguntando o examinando. En este apartado se seleccionan las estrategias para la obtención de los datos, es decir, los sujetos a examinar, las entrevistas, el estudio de documentos personales y la observación, entre otras. Todo ello desde la perspectiva del investigador y la del caso.

4. **Análisis e interpretación:** se sigue la lógica de los análisis cualitativos. Se trata de la etapa más delicada del estudio de caso.

5. **Elaboración del informe:** se debe contar de manera cronológica, con descripciones minuciosas de los eventos y situaciones más relevantes. Además, se debe explicar cómo se ha conseguido toda la información (recogida de datos, elaboración de las preguntas, etc.). Todo ello para trasladar al lector a la situación que se cuenta y provocar su reflexión sobre el caso. (p. 147).



**Figura 5.** Fases para desarrollar el método

**Fuente:** Jiménez (2012, p. 147)

## 4.2. Aplicación de verificación y validación de procesos en proyectos reales externos

Hemos visto las definiciones de verificación y validación, pero es importante que podamos revisar como se realiza la aplicación, esto por medio de algunos casos que lo ejemplificarán.

Consideremos el siguiente ejemplo:

Se tiene un programa que lee tres números enteros, estos son interpretados como la representación de las medidas de los lados de un triángulo. El programa envía un mensaje indicando si el triángulo es escaleno, isósceles o equilátero.



Queremos detectar defectos testeando el programa. Posibles casos que probar:

- Lado\_1 = 1, Lado\_2 = 0, Lado\_3 = 0 Resultado = error
- Lado\_1 = 2, Lado\_2 = 3, Lado\_3 = 4 Resultado = escaleno

Lo que acabamos de revisar, son casos de prueba.

A continuación, comparamos el resultado esperado con el obtenido y si son diferentes lo más probable es que haya fallado el programa

De manera intuitiva sería bueno probar otros casos como, por ejemplo:

- Lado\_1 = 4, Lado\_2 = 2, Lado\_3 = 2 Resultado = isósceles
- Lado\_1 = 3, Lado\_2 = 3, Lado\_3 = 3 Resultado = equilátero

¿Por qué considerar estos casos para probar?

- Debemos por lo menos considerar el probar un caso por cada respuesta posible del programa (equilátero, escaleno, error, isósceles)

Otra manera utilizada para la detección de defectos es revisar el código:

```
If Lado_1 = Lado_2 or Lado_2 = Lado_3 then write ("Triángulo  
Equilátero") else ...
```

En este caso podemos detectar que en vez de un **or** debiese ir un operador **and**. Tenemos la posibilidad de realizar la revisión de forma individual o con varios grupos de código.



Otro ejemplo de aplicación:

Ejemplo de Verificación y Validación V&V de un programa que permite gestionar los estudiantes matriculados en los cursos de una academia **Verificación:**

- ¿las operaciones de gestión de las listas de estudiantes funcionan correctamente?
- ¿las operaciones de gestión de cursos funcionan correctamente?

**Validación:**

- ¿El programa proporciona todas las operaciones que el usuario necesita?
- ¿Las operaciones hacen lo que el usuario espera?
- ¿La interfaz con el usuario es la apropiada?

### 4.3. Análisis objetivo

El análisis objetivo se enfoca en determinar el informe final en base a las pruebas que se han efectuado, la satisfacción del usuario con relación al sistema probado y al proceso que se ha empleado, con la finalidad de realizar una mejora continua.

Cabe destacar que esta información es almacenada, a manera de histórico, para pruebas futuras, por lo general se recomienda no eliminar los resultados de las pruebas obtenidos ya que nos permiten



tener una trazabilidad muy importante de todo el proceso de desarrollo.

Como hemos visto, en el análisis objetivo se sigue la lógica de los análisis cualitativos. Jiménez (2012), como hemos citado anteriormente, describe el análisis objetivo como la etapa más delicada del estudio de caso

El objetivo es tratar la información recopilada durante la fase de recopilación y establecer relaciones causa-efecto tanto como sea posible respecto de lo observado. Contrariamente a las fases de diseño y de recopilación de datos, este análisis está menos sujeto a metodologías de trabajo, lo que de hecho constituye su relativa dificultad. Tras establecer una correlación entre los contenidos y los personajes, tareas, situaciones, etc., de nuestro análisis; cabe la posibilidad de plantearse su generalización o su exportación a otros casos. (p. 147).





## Conclusiones

En esta semana estudiamos verificación, validación y estándares internacionales que nos entregan los lineamientos para realizar nuestro trabajo pero, ¿qué obtenemos?

En primer lugar, podemos mejorar la competitividad, vamos a lograr una importante disminución de los errores, se les entregará a los clientes soluciones rápidas, todo lo anterior se verá reflejado en la mejora que obtendrá la organización.

Por otro lado, hemos visto que los estudios de casos ofrecen perspectivas interesantes en el proceso, extraen directamente la esencia de los temas estudiados que pueden ser de gran beneficio para los profesionales. Se constituye en un camino difícil de tomar en lo referente al investigador, un desafío teniendo en cuenta que ante todo el conocimiento y la capacidad de ser objetivos en cuanto a los resultados subjetivos deben ser evidenciados en todo lo natural del contexto y no incidir en los resultados.



## Referencias bibliográficas

Drake, J. y López, P. (2009). Ingeniería softwares. 4º de Físicas. Verificación y validación. [Documento PDF]. Recuperado de [https://www.ctr.unican.es/asignaturas/Ingenieria\\_Software\\_4\\_F/Doc/M7\\_09\\_VerificacionValidacion-2011.pdf](https://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf)

Gómez, C. (2021). Pruebas dinámicas versus pruebas estáticas. [Imagen]. Recuperado de <https://www.diariodeqa.com/post/pruebas-din%C3%A1micas-vs-pruebas-est%C3%A1ticas>

Jiménez, V. (2012). El estudio de caso y su implementación en la investigación. En: Rev. Int. Investig. Cienc. Soc, (8)1, pp. 141-150. Recuperado de <https://dialnet.unirioja.es/servlet/articulo?codigo=3999526>

Rachell, W. (s. f.). Ejemplos de programas para la administración de bases de datos. Recuperado de [https://techlandia.com/ejemplos-programas-administracion-bases-datos-lista\\_126285/](https://techlandia.com/ejemplos-programas-administracion-bases-datos-lista_126285/)

Sites.google.com. (s. f.). Filosofía de calidad. [Entrada de blog]. Recuperado de <https://sites.google.com/site/businesscontrolesi/estructura/filosofia-de-calidad?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1>