# 0 Contenido

[Contenido](#_heading=h.gjdgxs) **1**

[Fundamentos del testing](#_heading=h.30j0zll) **1**

[Introducción al Testing](#_heading=h.1fob9te) 1

[Lecciones no aprendidas](#_heading=h.3znysh7) 2

[Aspectos psicológicos del Testing](#_heading=h.2et92p0) 4

[Mesa de 3 patas](#_heading=h.tyjcwt) 6

[Ciclo de vida de las pruebas de software](#_heading=h.3dy6vkm) 7

[Clase Sincrónica](#_heading=h.1t3h5sf) 11

[Aspecto psicológico del testing](#_heading=h.4d34og8) 13

[7 principios del testing](#_heading=h.2s8eyo1) 15

[Ciclo de Deming](#_heading=h.17dp8vu) 18

[Ciclo de Vida de las pruebas de Software](#_heading=h.3rdcrjn) 18

[**Fundamentos del Testing**](#_heading=h.xv41hjx9g8vz) **20**

[Desde el análisis a la implementación](#_heading=h.a6g3zqp30r0t) 20

[Niveles de pruebas](#_heading=h.lfcpufkzow2h) 21

[Modelo V](#_heading=h.js7t9jat06mi) 21

[Atributos de niveles de pruebas](#_heading=h.l5vfb3tv1ckg) 24

[Prueba de componente](#_heading=h.qt77je8koh6f) 24

[Prueba de integración](#_heading=h.bnqbws3k2ral) 26

[Prueba de Sistema](#_heading=h.t8fnqzdsbf5h) 28

[Prueba de aceptación](#_heading=h.fvnzjbewuqav) 30

[Tipos de prueba](#_heading=h.xz1409ld06hn) 32

C1 – TESTING I

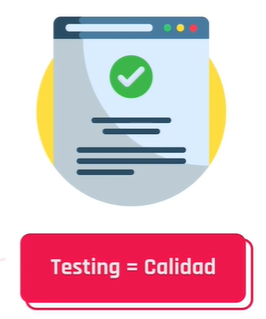
# Fundamentos del testing

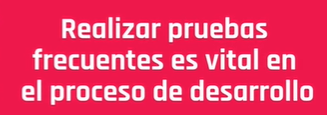
## Introducción al Testing

Todos los días hacemos testing.

Como testers probamos si todo funciona de acuerdo a lo esperado y en caso contrario, avisamos para que se pueda solucionar.

Un tester parte siempre de la suposición de que **el programa contiene errores**.

**La calidad es la satisfacción de un cliente ante un producto o servicio**. Con esta información se puede tomar decisiones, por ejemplo, saber cuándo el producto está listo para ser liberado.

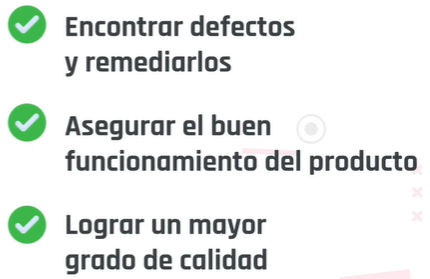


Sin embargo, hay factores externos que hacen que, más allá de haber realizado las pruebas adecuadas, se estén llevando a cabo sobre requerimientos equivocados 🡪 **Factor humano**.

Existen proyectos que en la fase final tienen que volver a corregirse por errores humanos:

* Requerimientos poco claros o mala comunicación durante el proceso
* Modificaciones a último momento
* Errores de diseño y testers no calificados.

Como conclusión, el testing permite:



## Lecciones no aprendidas

De lo visto anteriormente, se desprende que, si bien la ingeniería de software es una disciplina relativamente joven, durante su corta vida ha atravesado diferentes etapas de las cuales aprendió la lección: **la calidad es un valor en sí mismo y no un gasto que las empresas deben realizar para que su negocio prospere**.  
Sin embargo, se dice que esta ingeniería “es como una persona de la cual todos alguna vez se enamoran, pero con la que muy pocos están dispuestos a casarse” porque **es muy común encontrarse con proyectos que generan productos de dudosa calidad debido a que los responsables decidieron no realizar tareas de revisión de diseño y código** **o** determinadas **pruebas** porque dilatan el tiempo de salida al mercado, cuestan dinero y además no habrá diferencia entre uno y otro producto cuando el proyecto termine.



|  |
| --- |
| **La prueba muestra la presencia de defectos, no su ausencia** |
| No puede probar que no hay defectos. Reduce la probabilidad de que queden defectos no descubiertos en el software, pero, incluso si no se encuentran, el proceso de prueba no es una demostración de corrección. |

|  |
| --- |
| **La prueba exhaustiva es imposible** |
| No es posible probar todo —todas las combinaciones de entradas y precondiciones—, excepto en casos triviales. En lugar de intentar realizar pruebas exhaustivas se deberían utilizar el análisis de riesgos, las técnicas de prueba y las prioridades para centrar los esfuerzos de prueba. |

|  |
| --- |
|  |
| **La prueba temprana ahorra tiempo y dinero** |
| Para detectar defectos de forma temprana, las actividades de testing, tanto estáticas como dinámicas, deben iniciarse lo antes posible en el ciclo de vida de desarrollo de software para ayudar a reducir o eliminar cambios costosos. |

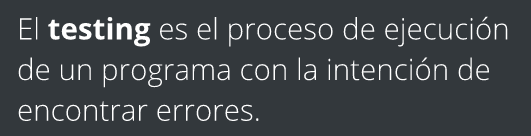
|  |
| --- |
| **Los defectos se agrupan** |
| En general, un pequeño número de módulos contiene la mayoría de los defectos descubiertos durante la prueba previa al lanzamiento o es responsable de la mayoría de los fallos operativos. |

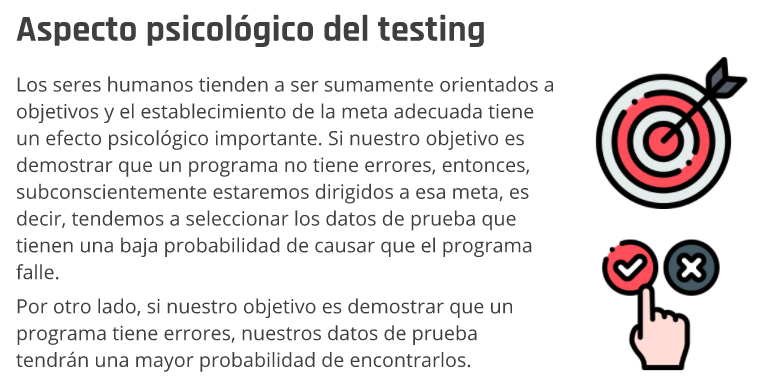
|  |
| --- |
| **Cuidado con la prueba del pesticida** |
| Si las mismas pruebas se repiten una y otra vez, eventualmente estas pruebas ya no encontrarán ningún defecto nuevo. Para detectarlo, es posible que sea necesario cambiar las pruebas y los datos de prueba existentes. |

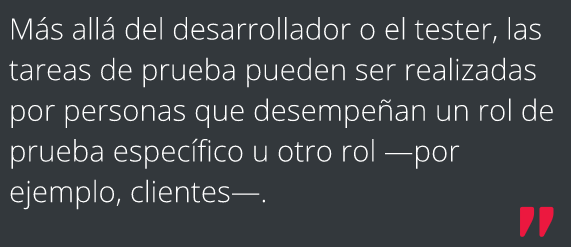
|  |
| --- |
| **La prueba se realiza de manera diferente según el contexto** |
| Por ejemplo, el software de control industrial de seguridad crítica se prueba de forma diferente a una aplicación móvil de comercio electrónico. |

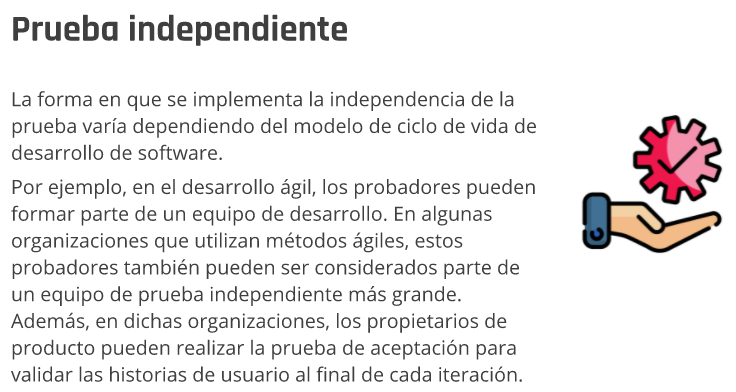
|  |
| --- |
| **La ausencia de errores es una falacia** |
| El éxito de un sistema no solo depende de encontrar errores y corregirlos hasta que desaparezcan ya que puede no haber errores, pero sí otros problemas. Existen otras variables a tener en cuenta al momento de medir el éxito. |

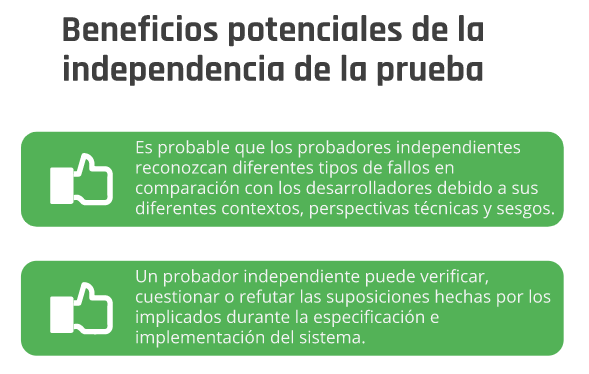
## Aspectos psicológicos del Testing







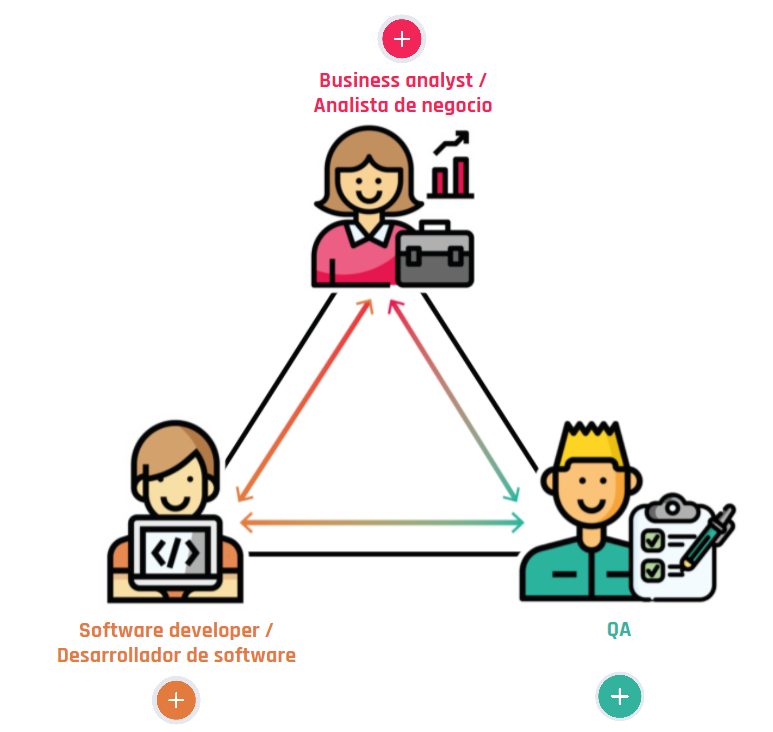






## Mesa de 3 patas

Si bien cada actor tiene un rol definido, es necesario un trabajo en comunión entre los 3 actores. Es decir, es necesario que trabajen como equipo. Por eso, utilizamos la analogía con una mesa de 3 patas, pues si falta alguna de ellas, la mesa no podría estar de pie.  
En algunas empresas de software pequeñas o “start up” es posible que una misma persona tenga más de un rol.

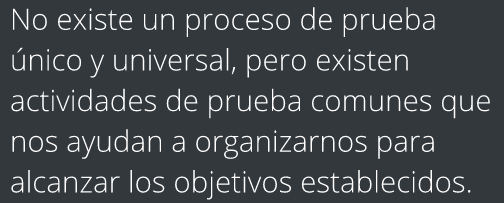


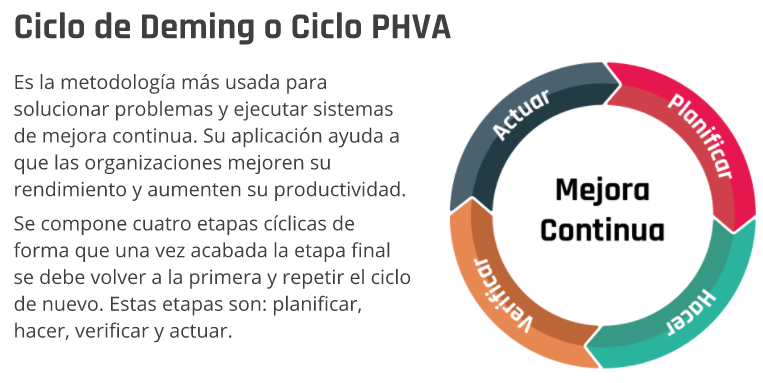
|  |
| --- |
| **Business analyst / Analista de negocio** |
| S  Se encarga de detectar los factores clave del negocio y es el intermediario entre el departamento de sistemas y el cliente final. |

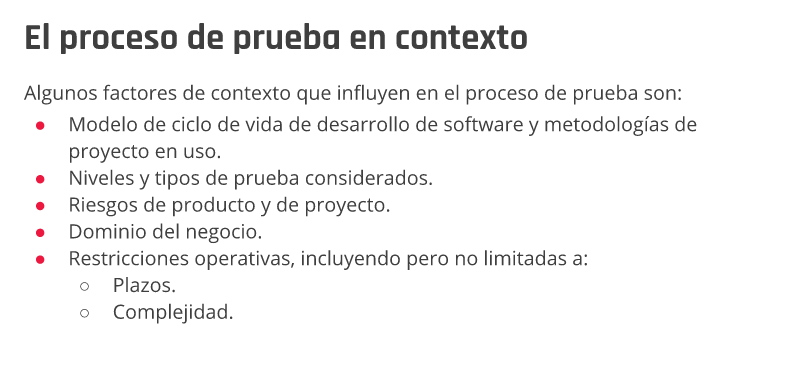
|  |
| --- |
| **Software developer / Desarrollador de software** |
| Su función es diseñar, producir, programar o mantener componentes o subconjuntos de software conforme a especificaciones funcionales y técnicas para ser integrados en aplicaciones. |

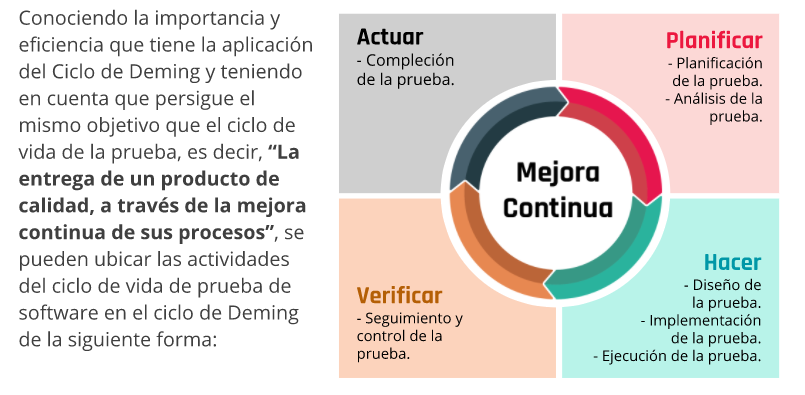
|  |
| --- |
| **QA** |
| La principal función es probar los sistemas informáticos para que funcionen correctamente de acuerdo a los requerimientos del cliente, documentar los errores encontrados y desarrollar procedimientos de prueba para hacer un seguimiento de los problemas de los productos de forma más eficaz y eficiente. |

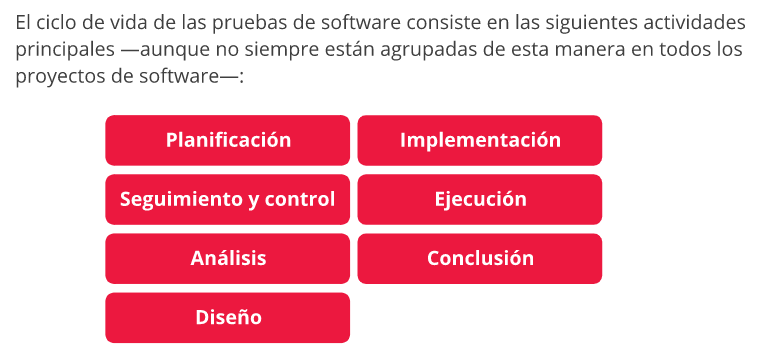
## Ciclo de vida de las pruebas de software

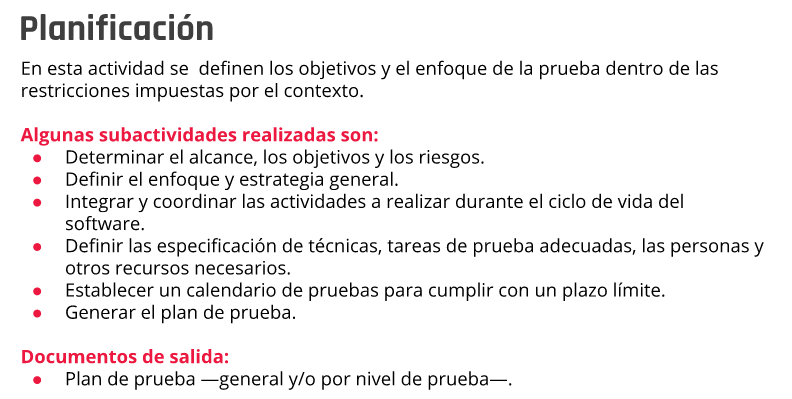


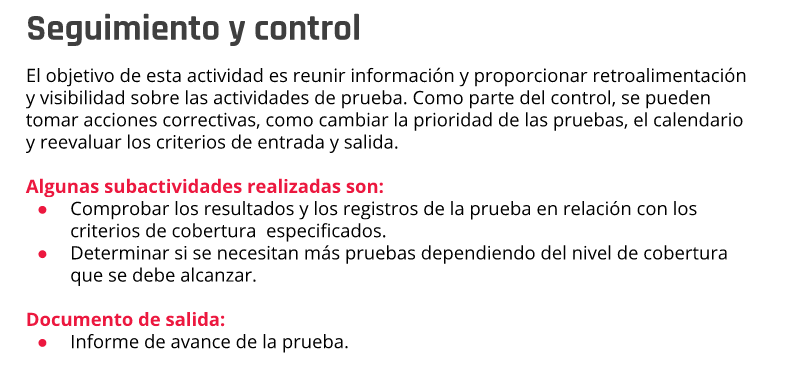


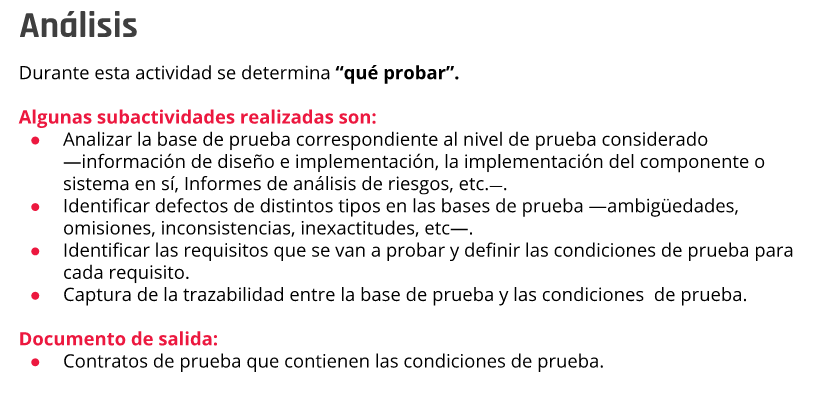


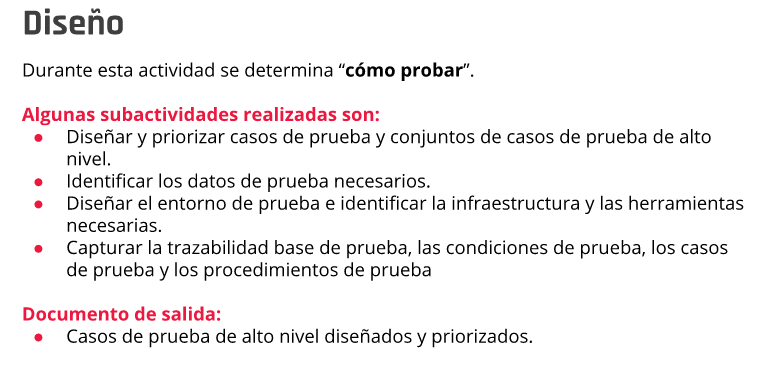


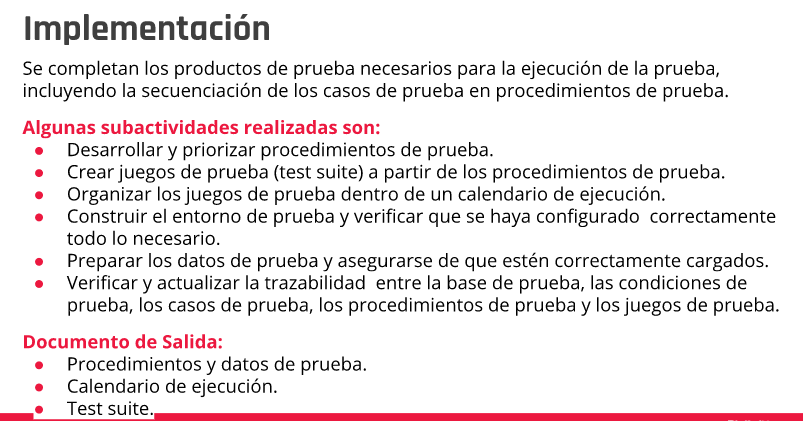


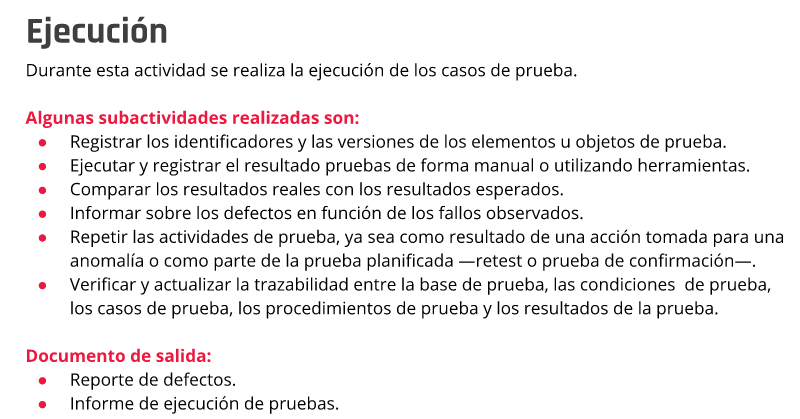


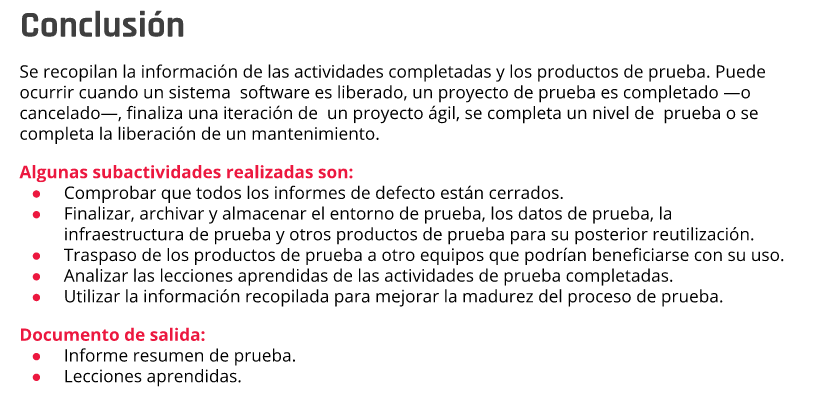












## Clase Sincrónica

Docentes:



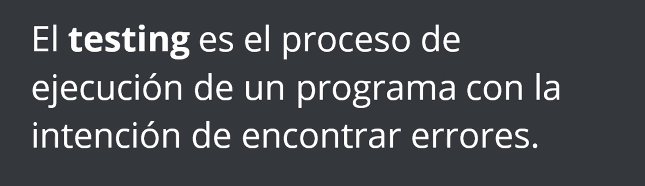


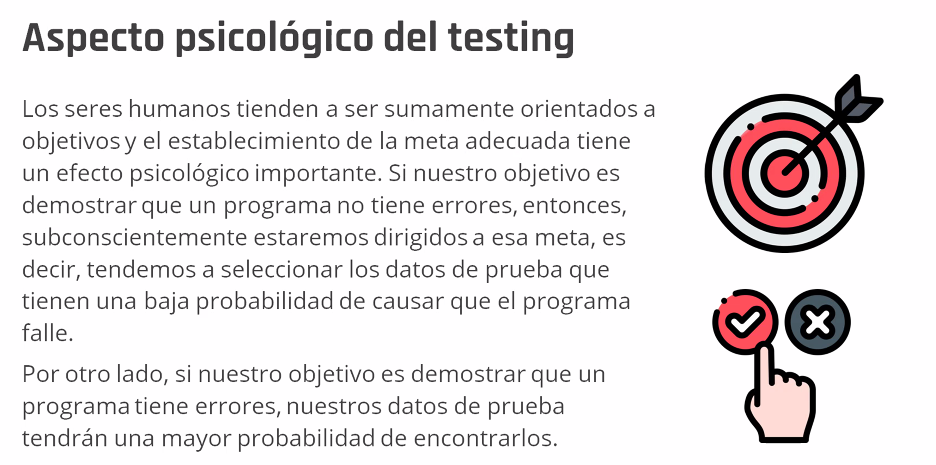


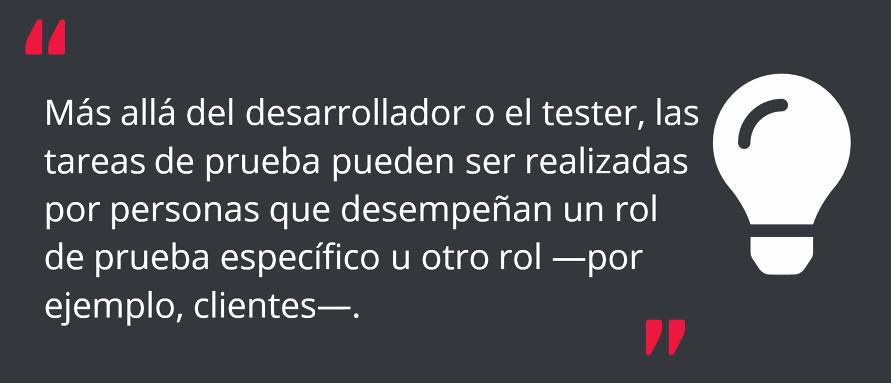


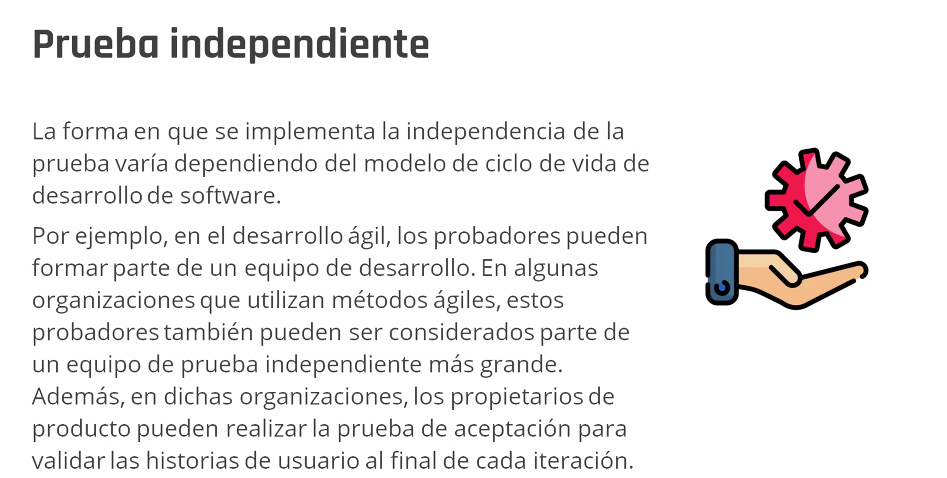


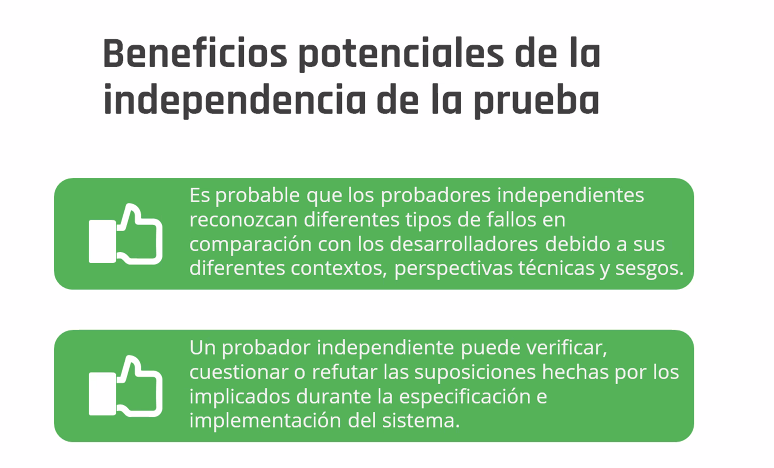
### Aspecto psicológico del testing





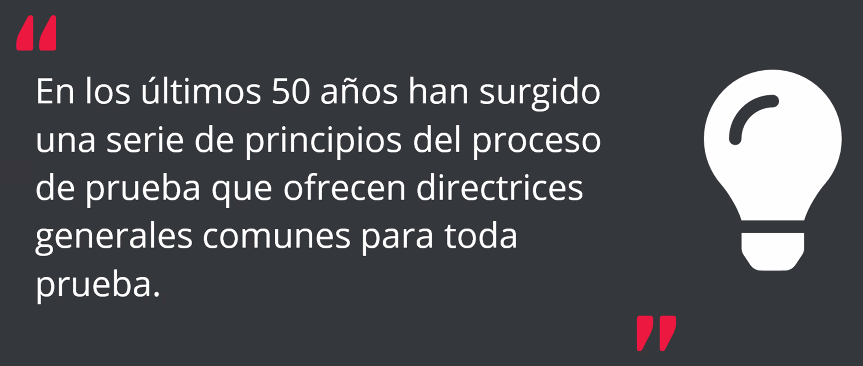


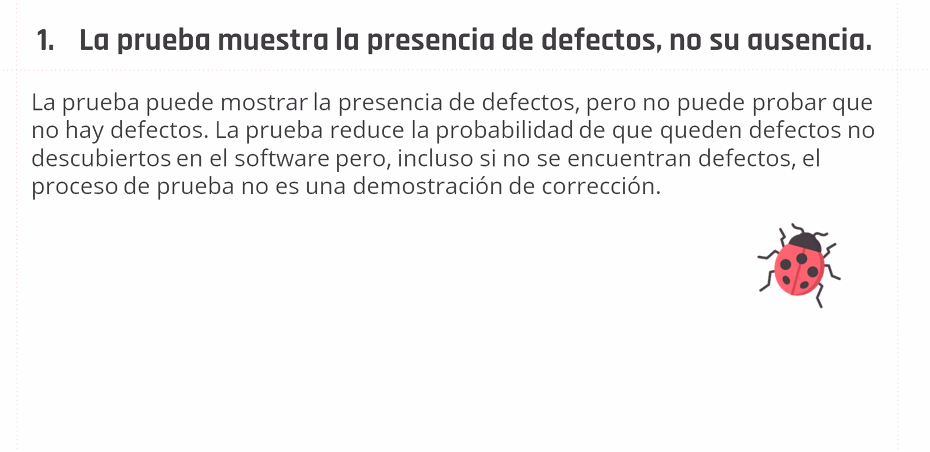


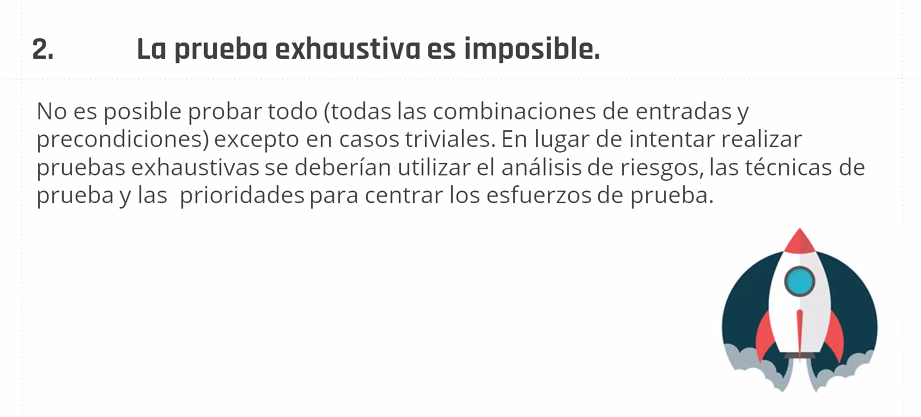


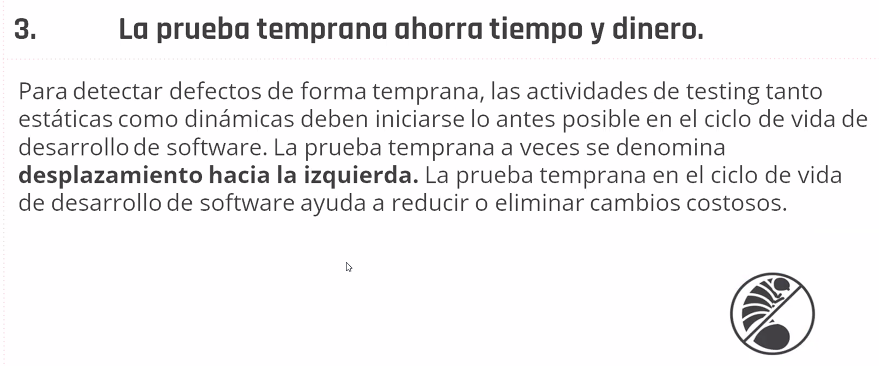


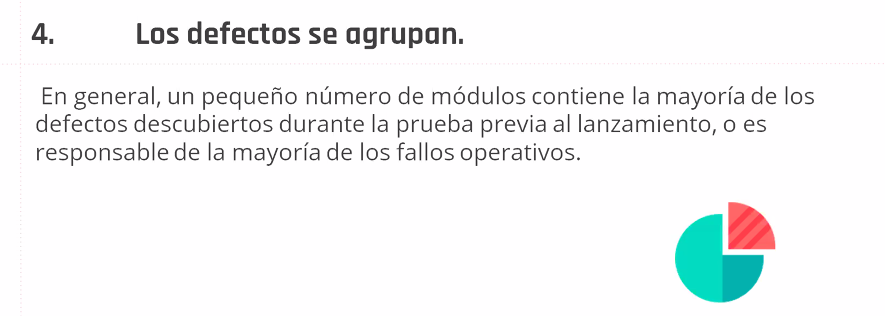
### 7 principios del testing

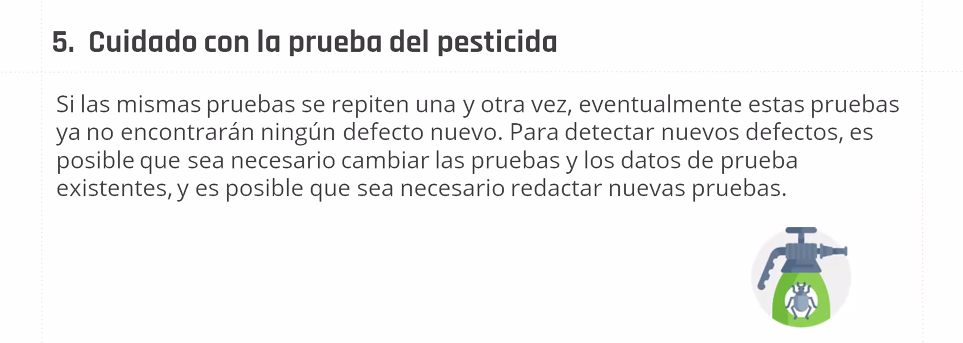


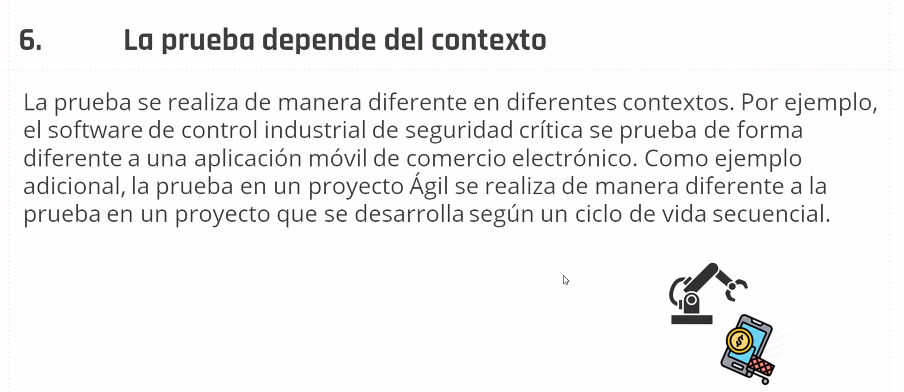


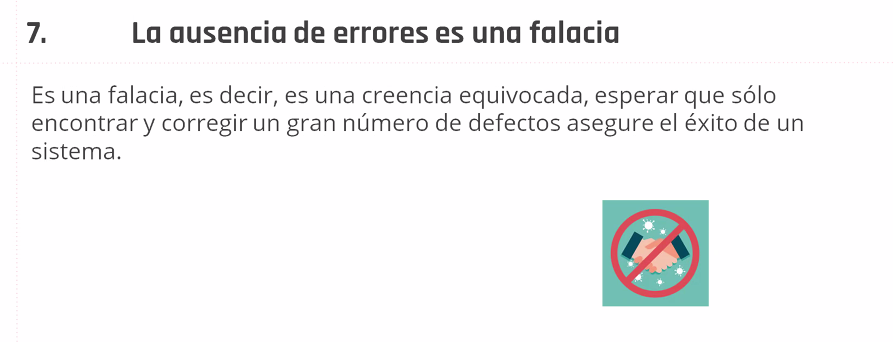


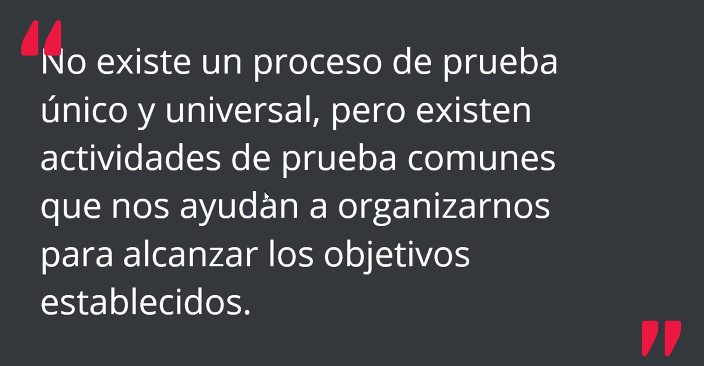




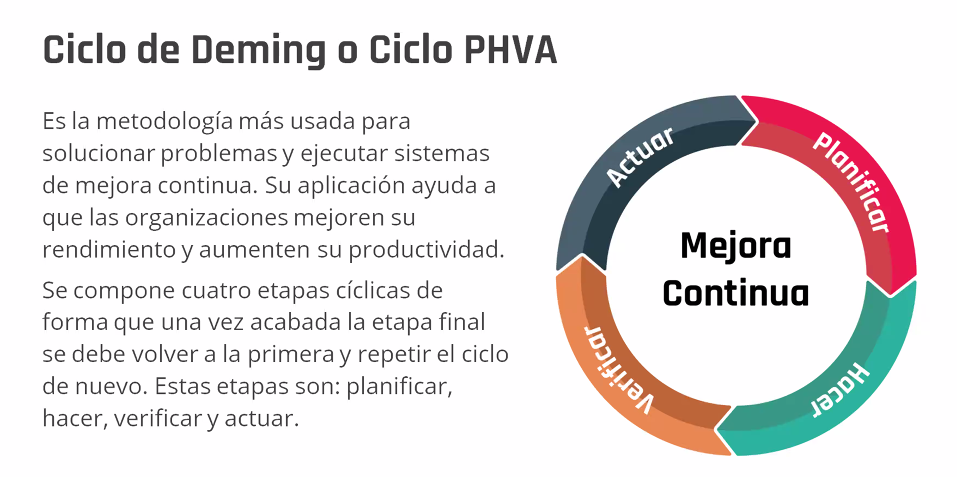




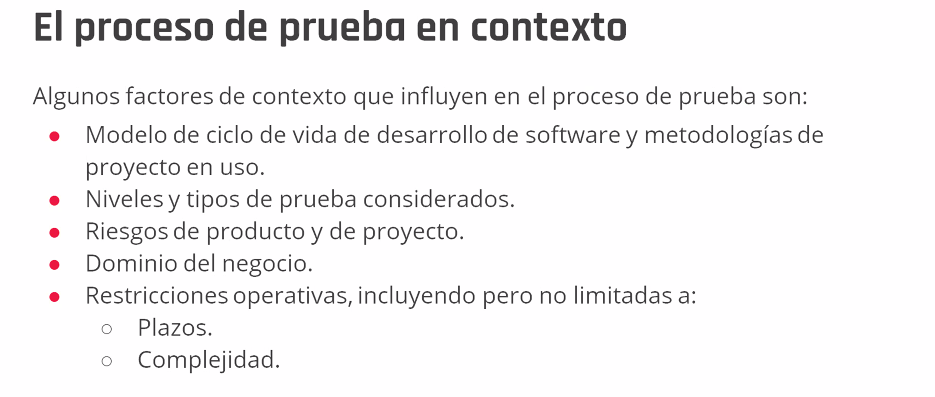


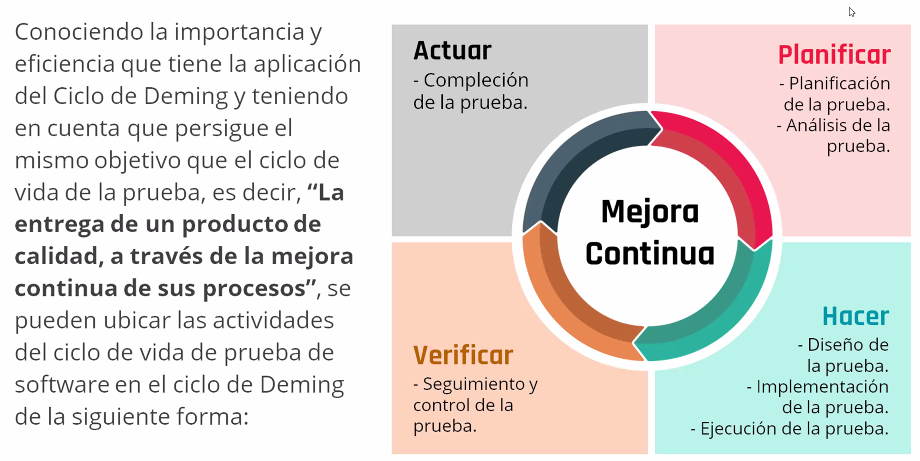


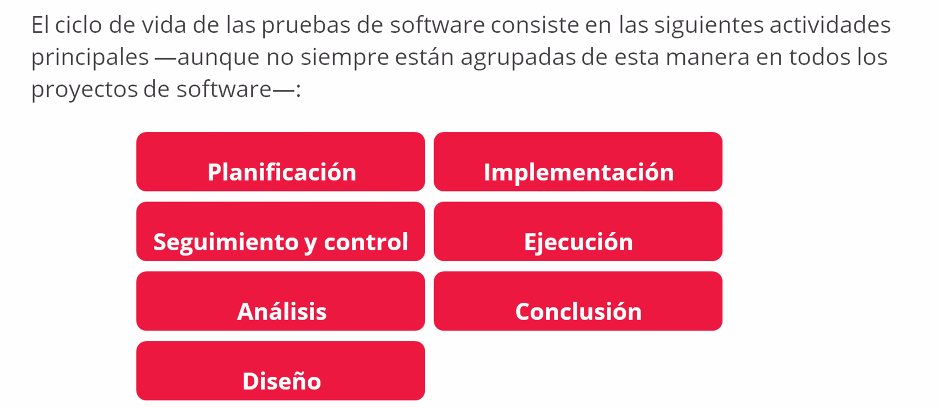
### Ciclo de Deming



### Ciclo de Vida de las pruebas de Software

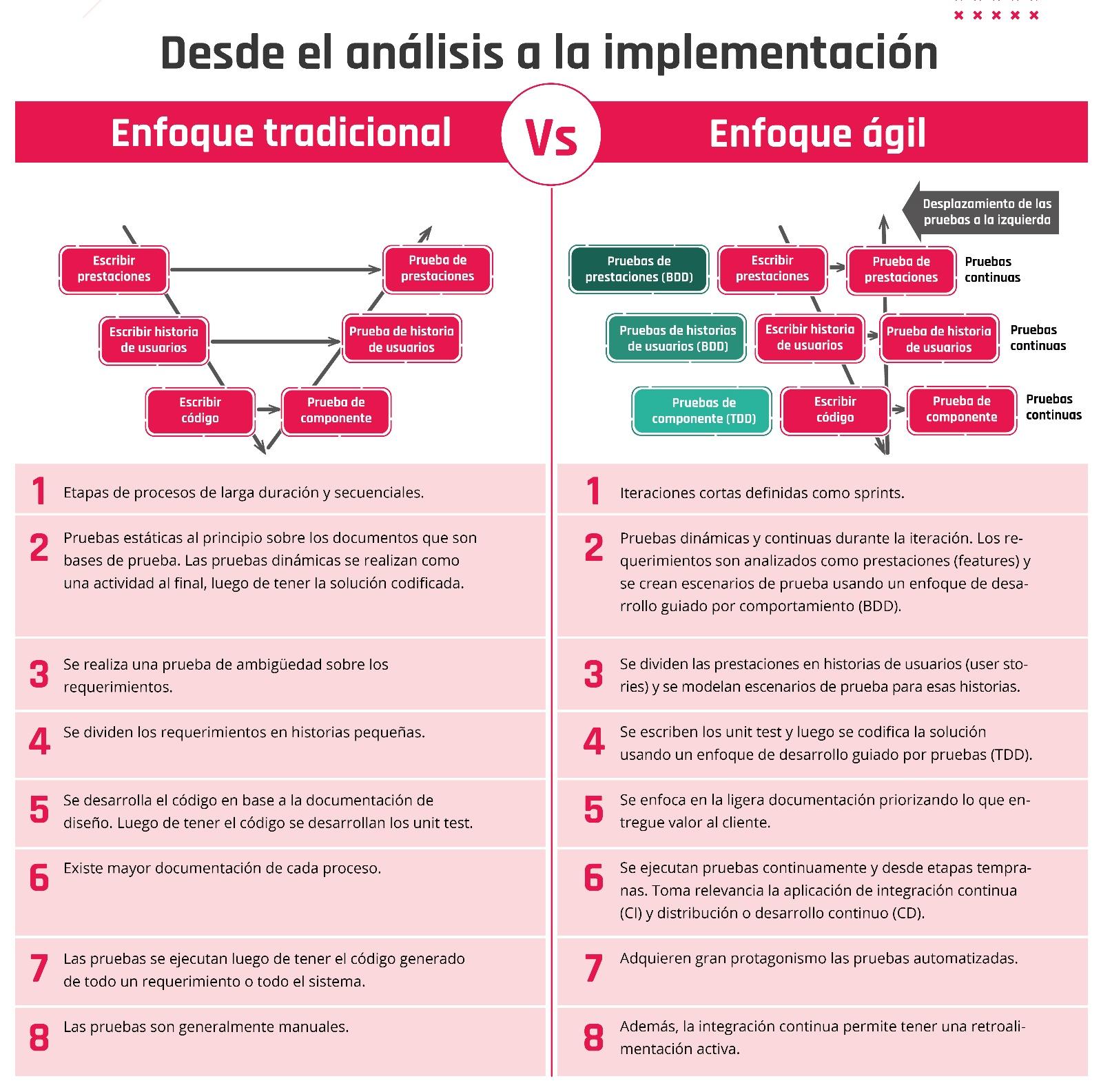






# Fundamentos del Testing

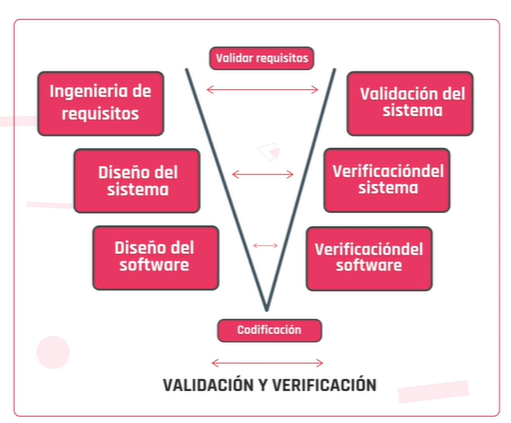
## Desde el análisis a la implementación



## Niveles de pruebas

### Modelo V

Cuando se desarrolla un software, hay que llevar a cabo pruebas en todos los niveles para entregar al cliente un producto que cumpla sus expectativas y necesidades asegurando la calidad del mismo. Para entregar un producto de calidad y confiable, hay que testearlo; para esto, existen distintos niveles de pruebas o actividades que se organizan y gestionan conjuntamente.

Para comprender qué son niveles de pruebas, se utiliza el **modelo en V**, que es un modelo empleado en diversos procesos de desarrollo (por ejemplo, en el desarrollo de software). En los años 90, apareció su primera versión, pero con el tiempo, se ha ido perfeccionando y adaptando a los métodos modernos de desarrollo. Además de las **fases de desarrollo** de un proyecto, el modelo V también define los **procedimientos de gestión de la calidad** que lo acompañan y describe cómo pueden **interactuar** estas fases individuales entre sí. Este modelo compara las fases de desarrollo con las fases del control de la calidad correspondiente.

**Para cada fase del desarrollo debe existir un resultado verificable.**

Del lado izquierdo se observan las fases del desarrollo y diseño, y para cada una de ellas, del lado derecho, se encuentran las medidas del control de la calidad.

En la unión de ambos lados se sitúa la implementación del producto.

Comenzando del lado izquierdo, cuando empieza el proyecto, el modelo prevé un análisis de requisitos y especificaciones; luego el proyecto se centra en las características funcionales y no funcionales, posteriormente se definen los componentes y las interfases, y finalmente se realiza la implementación.



En paralelo, de acuerdo a estas etapas, se produce el desarrollo en sí del software a través de diferentes niveles del control de la calidad o también llamadas de verificación o validación, que siempre están relacionadas con cada una de las fases de desarrollo.

**Verificación** es el conjunto de actividades que aseguran que el software implemente correctamente una función específica y **validación** es un conjunto diferente de actividades que aseguran que el software construido respete los requisitos del cliente.



Otra forma de definirlo sería:

Verificación: ¿Estamos construyendo el producto correctamente?

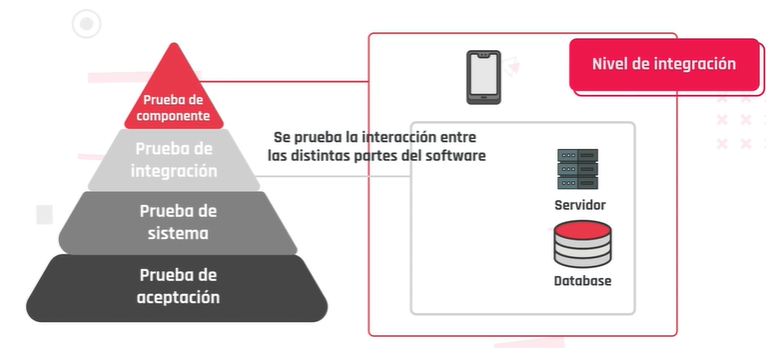
Validación: ¿Estamos construyendo el producto correcto?

El método V abarca las pruebas:

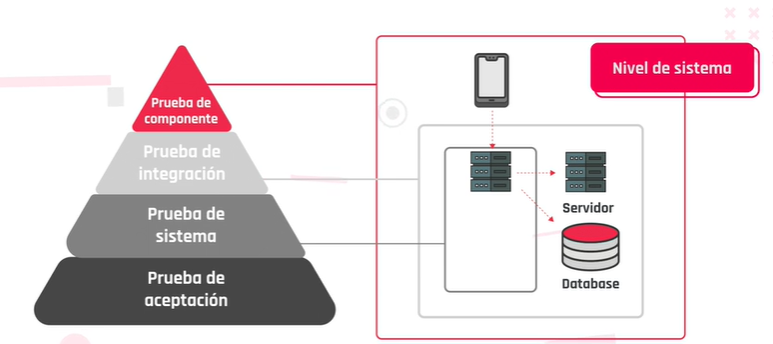
* De componente
* De integración
* De sistema
* De aceptación

En las pruebas de componente se aísla cada parte del programa (clases, módulos, objetos, paquetes, subsistemas, etc) para comprobar que cada una de esas partes funciona correctamente por separado.

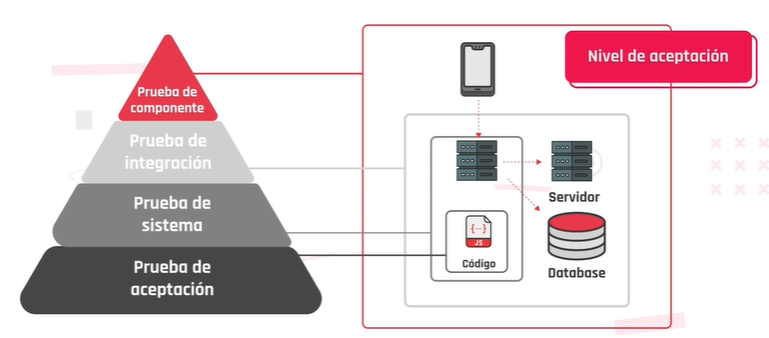
Por otro lado, se hacen las pruebas de integración que justamente se realizan sobre un conjunto de módulos de la aplicación para comprobar que funcionan correctamente.



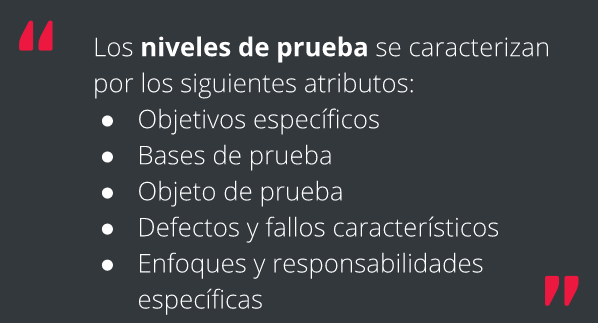
Es decir, se prueba la interacción entre las distintas partes del software. Luego las prueba del sistema son llevadas a cabo por el proveedor en un entorno de laboratorio para demostrar que el sistema cumple con los requisitos funcionales y no funcionales, y con el diseño técnico. Están diseñadas para probar el sistema en su totalidad.



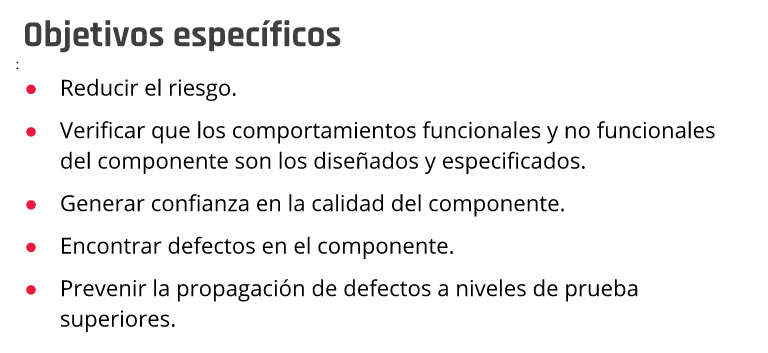
Por último, las pruebas de aceptación son realizadas por el usuario en un entorno muy similar al de producción para demostrar que el sistema cumple las especificaciones funcionales y requisitos del cliente.

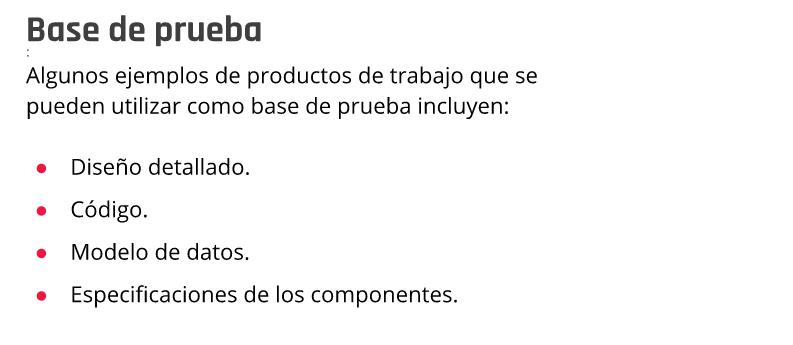


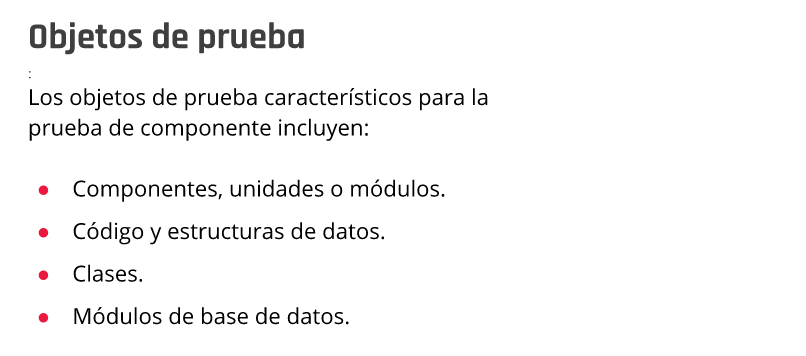
### Atributos de niveles de pruebas

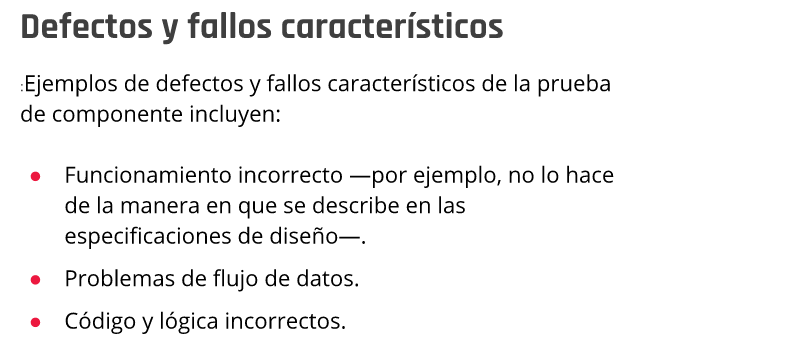


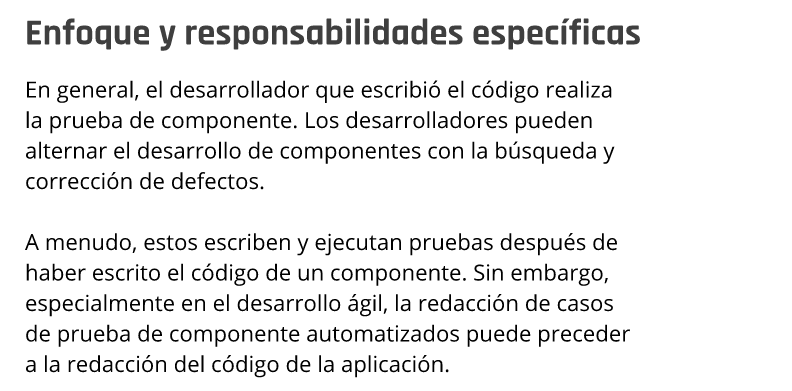
### Prueba de componente



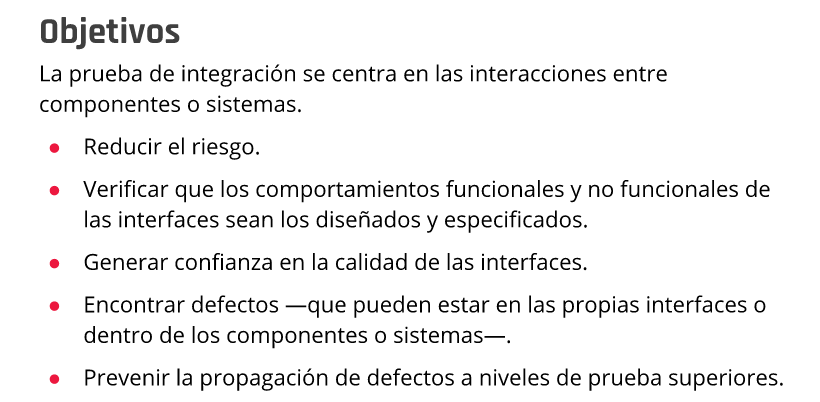




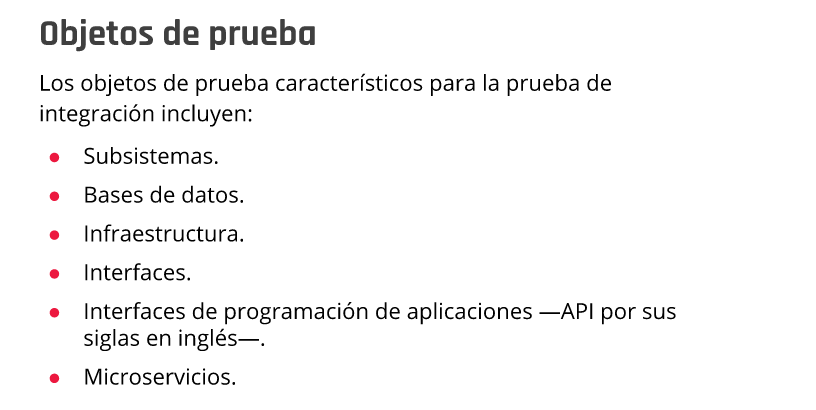


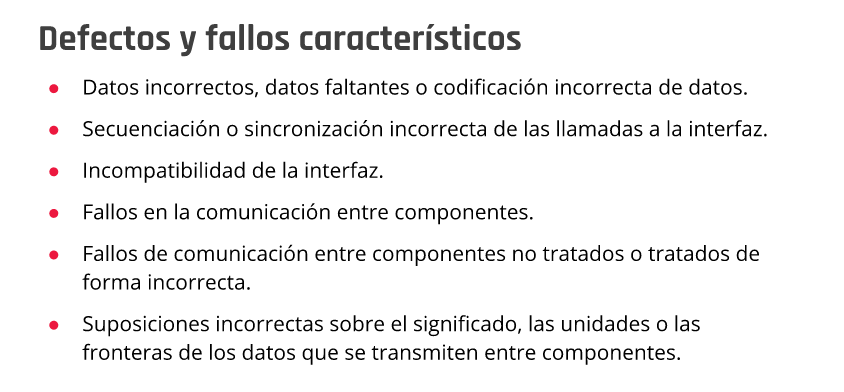


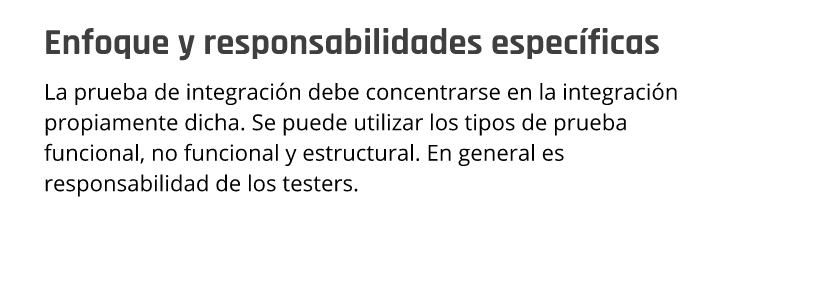
### Prueba de integración



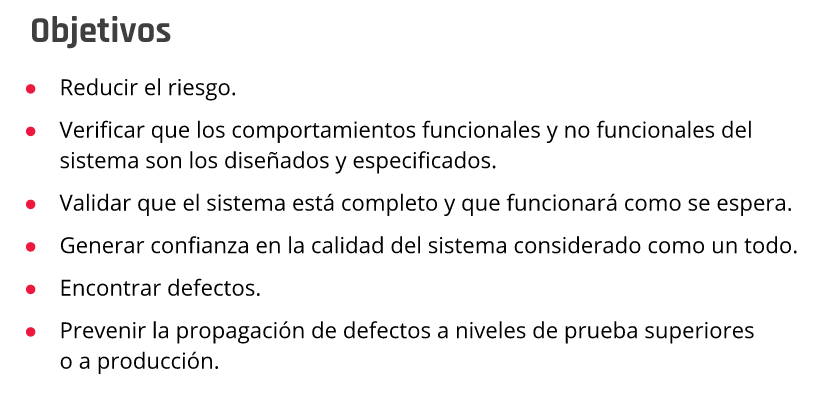


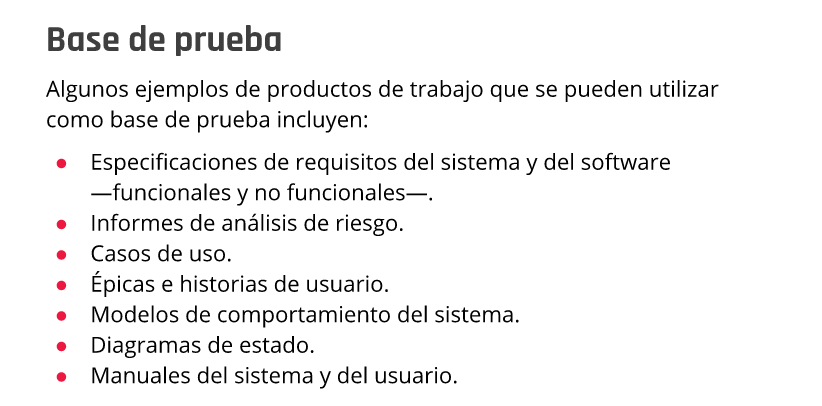


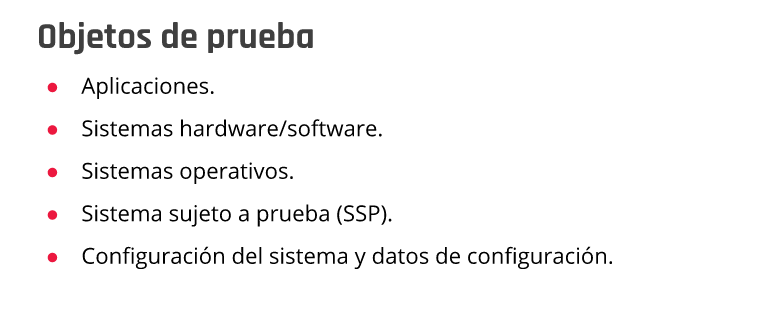


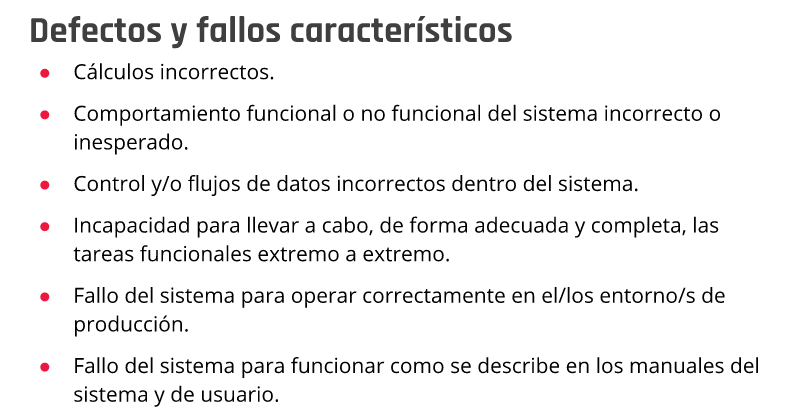


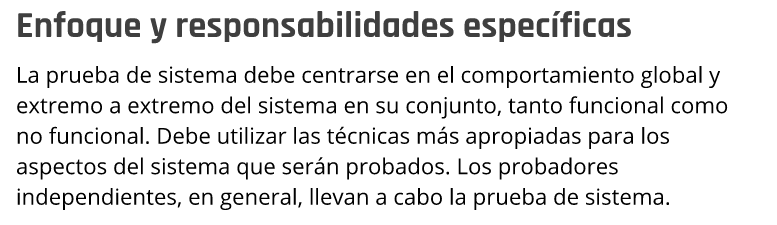
### Prueba de Sistema



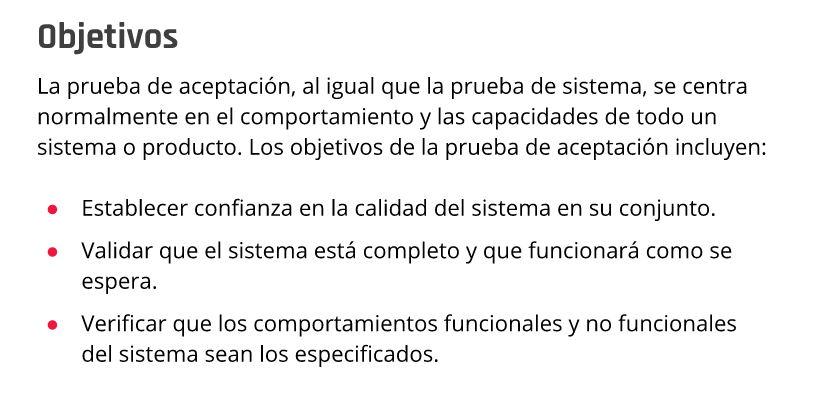


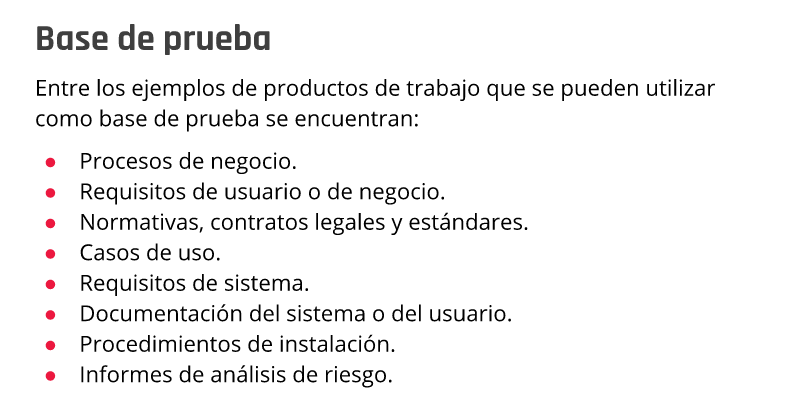


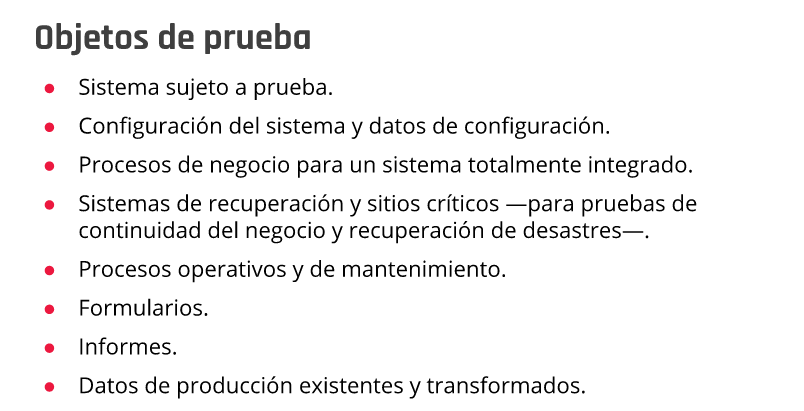


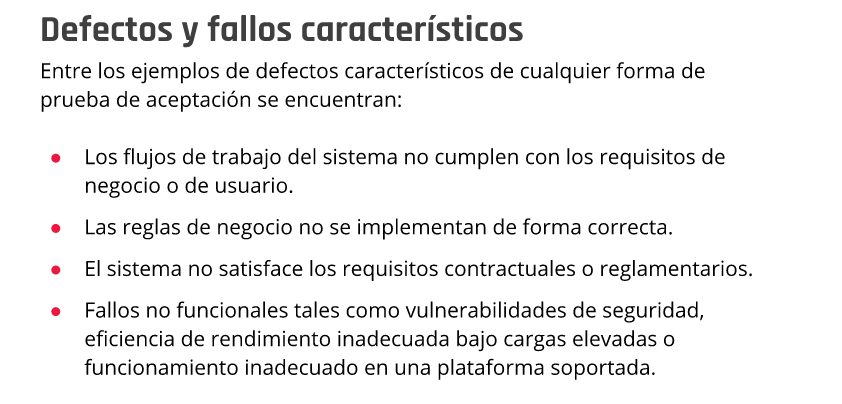


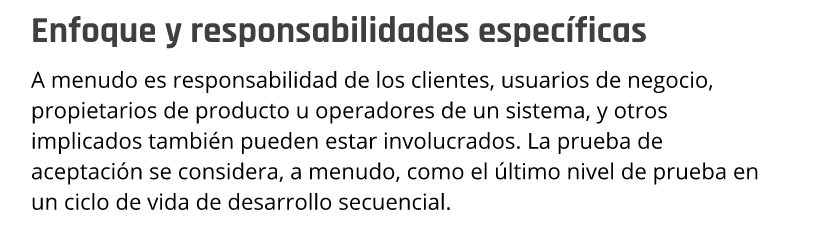
### Prueba de aceptación











## Tipos de prueba

Un tipo de prueba es un grupo de actividades **de pruebas destinadas a probar las características específicas de un sistema de software**, o de una parte de un sistema, basados en objetivos de pruebas específicas.

Dichos objetivos pueden incluir:

● Evaluar las características de calidad funcional tales como la completitud, corrección y pertinencia.

● Evaluar características no funcionales de calidad, tales como la fiabilidad, eficiencia de desempeño, seguridad, confiabilidad y usabilidad.

