DESARROLLO DE SOFTWARE

PRUEBAS DE SOFTWARE

QA / QC

El término de **QA** es muy utilizado entre las compañías del sector servicios con una gran presencia en el mundo tecnológico.

Aunque el término "**QA**" es relativamente nuevo, ya que el mundo de "testing", "automatización de pruebas", etc., encontró salida en el mercado IT a mediados del año 2008, muchísima gente confunde control de calidad (**QC**) con el término **QA** (Quality Assurance), que en sus siglas en Inglés, viene a significar GARANTÍA DE CALIDAD.

Los nombres son muy parecidos pero se refieren a procesos totalmente distintos, si bien es verdad que el QA se basa mucho en el feedback que se recibe desde **QC**, aumentando aún más la confusión.

QA / QC

QA (Quality Assurance)	vs	QC (Quality Control)
Se diseñan y definen todos los parámetros de aceptación de un paquete de Software		Se controla el comportamiento del producto final
Es un sistema de **PREVENCIÓN** de fallos que predice casi todo sobre la seguridad, funcionamiento, normas de calidad y legalidad de un paquete de SOFTWARE y genera medidas correctivas para controlar y evitar que los productos o servicios defectuosos lleguen a la fase de producción		Es un sistema de **CORRECCIÓN** de fallos e introducción de mejoras
El departamento QA trabaja junto a desarrollo, ingenieros, managers y el cliente		El departamento de QC, trabaja junto a QA
El departamento QA está presente desde la fase del diseño del producto		El departamento de QC entra en acción cuando el producto está finalizado.
El QA está orientado al proceso		QC está orientado al producto

QA / QC

QA asegura que todos los desarrolladores siguen el mismos estándar de calidad, dentro de una gran corporación	QC asegura que el funcionamiento del producto, es el esperado
QA se diseña y ejecuta antes de tener un producto finalizado	QC se ejecuta durante la puesta en pre-producción

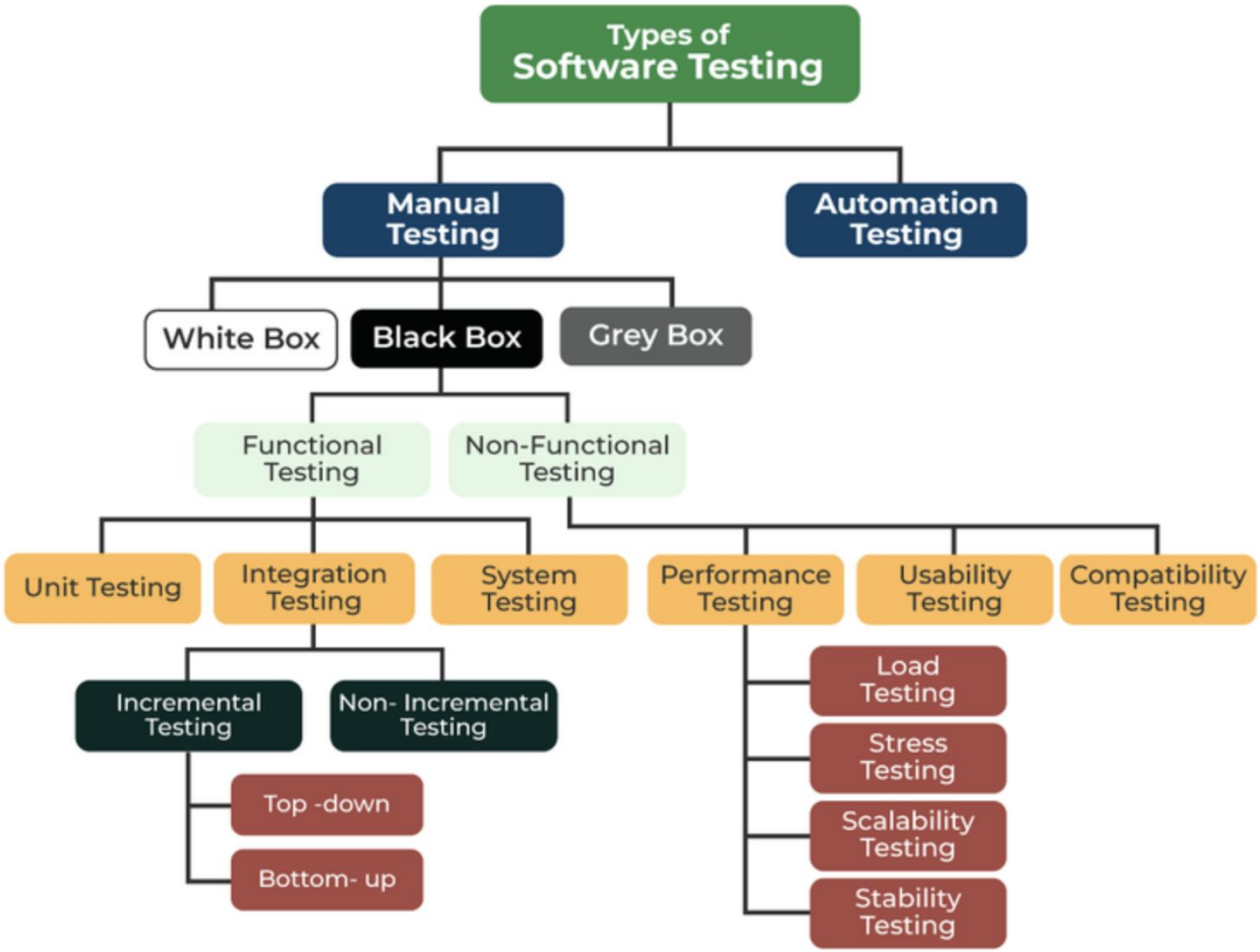
PRUEBAS DE SOFTWARE

Las pruebas de software son un conjunto de actividades y procesos que se realizan para evaluar la calidad de un software.

Su objetivo es identificar y corregir errores, verificar que el software cumple con los requisitos especificados y garantizar que funcione de manera correcta y confiable.

Las pruebas de software son una parte integral del desarrollo de software y se realizan en diferentes etapas del ciclo de vida del software.

TIPOS DE PRUEBAS DE SOFTWARE



TIPOS DE PRUEBAS DE SOFTWARE

Pruebas unitarias: Se enfocan en probar unidades individuales de código, como funciones o clases.

Pruebas de integración: Verifican que las unidades de código se integren y funcionen correctamente juntas.

Pruebas de sistema: Evalúan el sistema completo en su conjunto, incluyendo la interacción entre todas sus partes.

Pruebas de aceptación: Determinan si el software cumple con los requisitos definidos por los usuarios o clientes.

Pruebas de caja negra: Se realizan sin conocer la estructura interna del código, solo se evalúa el comportamiento del software a través de sus entradas y salidas.

Pruebas de caja blanca: Se basan en el conocimiento del código interno del software para diseñar casos de prueba que cubren diferentes rutas y escenarios de ejecución.

TESTING MANUAL

Un tester (usuario) prueba el software siguiendo unos casos de prueba

- Es acertado ya que un tester puede visualizar los errores
- Barato ya que no se requiere conocimientos de programación
- No se programa (no hay código)
- Eficiente cuando hay cambios inesperados (hot fix)

TESTING AUTOMATIZADO

El tester usa codigo de programacion para generar scripts que puedan ejecutarse de manera automática.

Simple Python Selenium webdriver example

simplesel.py — Raw

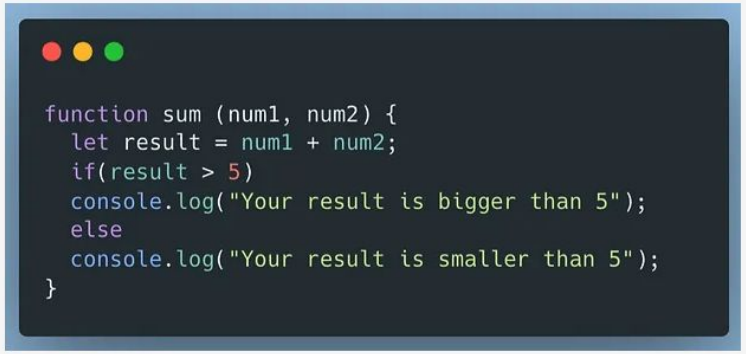
```python
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.keys import Keys


def selectOption(select,texttoselect):
    options = select.find_elements_by_tag_name("option")
    if options:
        for o in options:
            if o.text == texttoselect:
                o.click()

browser = webdriver.Firefox()
browser.get('http:///www.seek.co.nz')

keywords = browser.find_element_by_id('Keywords')
#require initial backspace to delete default text
keywords.send_keys(Keys.BACK_SPACE + "tech")
industries = browser.find_element_by_id('catindustry')
selectOption(industries,'Legal')
search = browser.find_element_by_id('DoSearch')
search.submit()
#this is an example to visually demonstrate controlling the browser
#however normally you would close the browser instance  with:
#browse
```

TEST DE CAJA BLANCA

- Se requiere conocer el código fuente y el diseño del software.
- Entiende de estructuras de datos.
- El tester revisa el código y realiza los casos de prueba en base al código.
- Técnicas:
  - Técnica de cobertura de sentencia
  - Técnica de cobertura de decisiones

TEST DE CAJA BLANCA

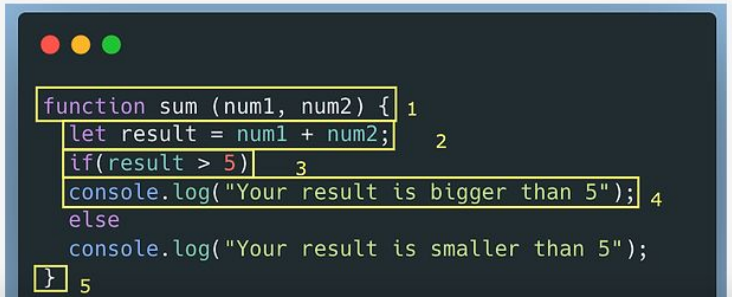
Dado el siguiente bloque de código, vamos a calcular cuantos casos de prueba se necesitan para alcanzar el 100% de cobertura.



```
function sum (num1, num2) {
  let result = num1 + num2;
  if(result > 5)
  console.log("Your result is bigger than 5");
  else
  console.log("Your result is smaller than 5");
}
```

Lo primero que debemos hacer es contar las sentencias, en este caso, vamos a decir que cada línea de código es una sentencia. Entonces para este ejemplo tenemos *7 sentencias*.

Si *num1 = 4* y *num2 = 6* entonces la variable *result* será igual a *10* y por tanto entraremos en la primera condición.

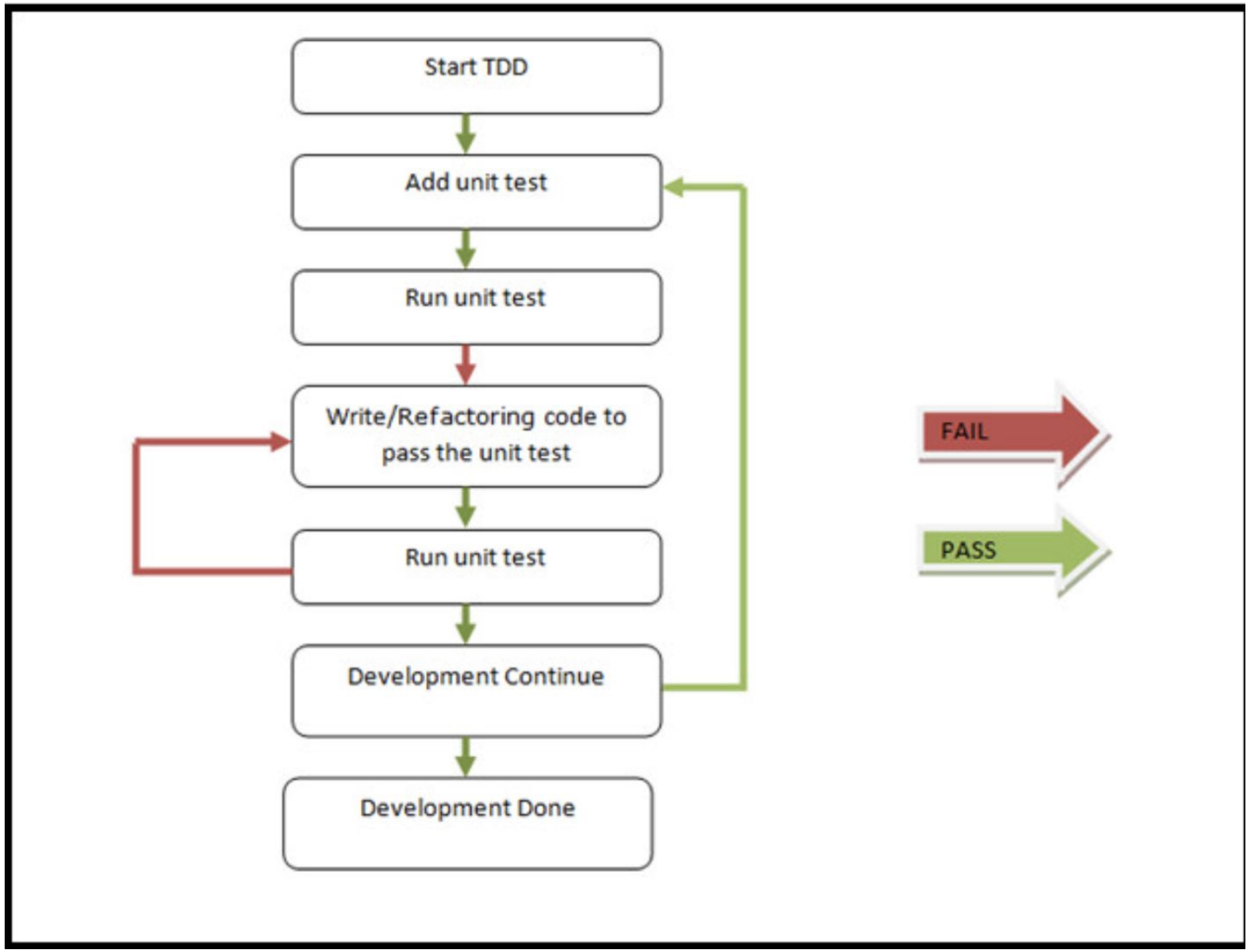


```
function sum (num1, num2) { 1
  let result = num1 + num2;   2
  if(result > 5)   3
  console.log("Your result is bigger than 5"); 4
  else
  console.log("Your result is smaller than 5");
} 5
```

Porcentaje de cobertura (como por ejemplo SonarQube)

Plantear unos casos de prueba

TDD - TEST DRIVEN DEVELOPMENT



- Hacer el test garantiza la cobertura y el correcto diseño.
- Detectar casos de borde
- El developer está obligado a hacer el test.
- Complicado para legacy
- Se tiene que entender el requerimiento.
- Verificar tema costos y tiempos

HERRAMIENTAS DE TESTING

- **JUnit**: Un marco de pruebas unitarias para Java.
- **PyUnit**: Un marco de pruebas unitarias para Python.
- **Selenium**: Una herramienta para pruebas de automatización web.
- **Appium**: Una herramienta para pruebas de automatización móvil.
- **Postman**: Una herramienta para pruebas de API.

CONCLUSIÓN

- Las pruebas de software son esenciales para desarrollar software de alta calidad y confiable.
- Al realizar pruebas de manera regular, se pueden detectar y corregir errores tempranamente, prevenir problemas y mejorar la experiencia del usuario.
- Existen diferentes tipos de pruebas de software, cada una con su propio enfoque y objetivo.
- TDD (Test Driven Development) es una metodología de desarrollo de software que promueve la escritura de pruebas unitarias antes de escribir el código.
- Existen diversas herramientas de testing que facilitan la realización de pruebas de software.

EVOLUCIÓN DE DEFECTOS

- ¿Cómo obtenerlo?
  - Registrar el defecto encontrado (QA)
  - Corregir el defecto (Desarrollo)
  - Revisar la corrección (QA)
  - Cerrar / Reabrir el defecto (QA)
- Esta estadística nos da una señal de la evolución de la estabilización.
  - Ej: se abren más defectos de los que se cierran por día.

EVOLUCIÓN DE DEFECTOS

Total

Promedio de Detectados por día	10
Promedio de Cerrados por día	6,95
Relación Cerrados / Total	69%

Últimos diez días

Promedio de Detectados por día	8,9
Promedio de Cerrados por día	5,1
Relación Cerrados / Total	57%

Últimos cinco días

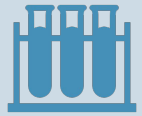
Promedio de Detectados por día	11,6
Promedio de Cerrados por día	5
Relación Cerrados / Total	43%

Estado/Severidad	A	B	C	D	Total
Abierto	43	53	70	35	201
Analizado	0	1	4	0	5
Revisado	1	2	3	0	6
Corregido	3	11	22	5	41
Cerrado	109	160	247	89	605
No Reproducible	0	3	0	0	3
Suspendido	4	6	2	0	12
Total	160	236	348	129	873

COBERTURA DE LA PRUEBA

- ¿Cuál es la calidad de lo construido hasta ahora?
- ¿Cómo mido el trabajo del equipo de QA?

- La cantidad de defectos es una medida muy indirecta de la calidad.



Mide la ejecución de la prueba a través de los casos de prueba.

Muestra:

Planificados: Cantidad de casos a ejecutar
Disponibles: Cantidad completada por desarrollo
Ejecutados: Cantidad de casos que QA pudo probar.
Ejecutados OK: Cantidad de casos sin defectos críticos.

El objetivo es obtener un indicador objetivo.

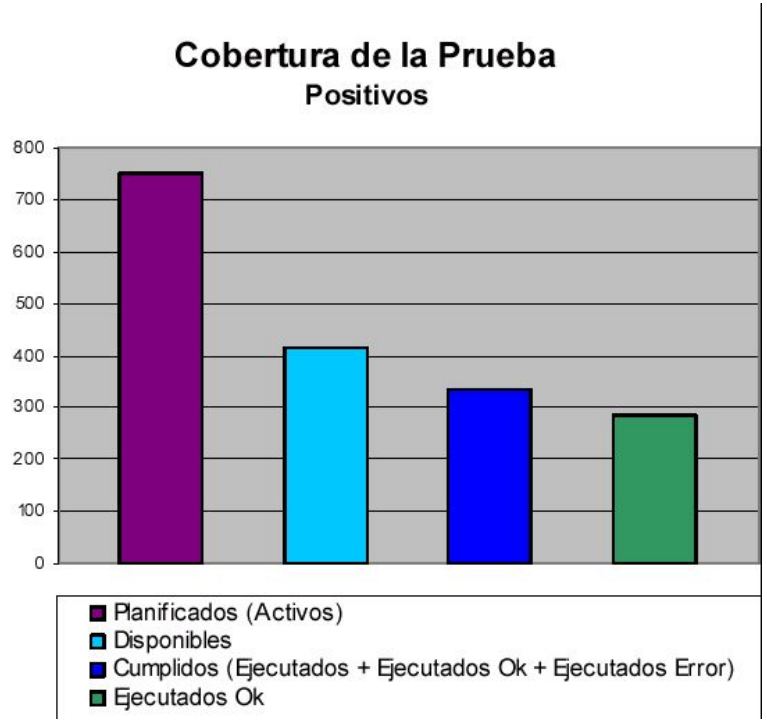
Avance no es lo que Desarrollo completó. Avance es lo que QA probó y no tenía defectos críticos.

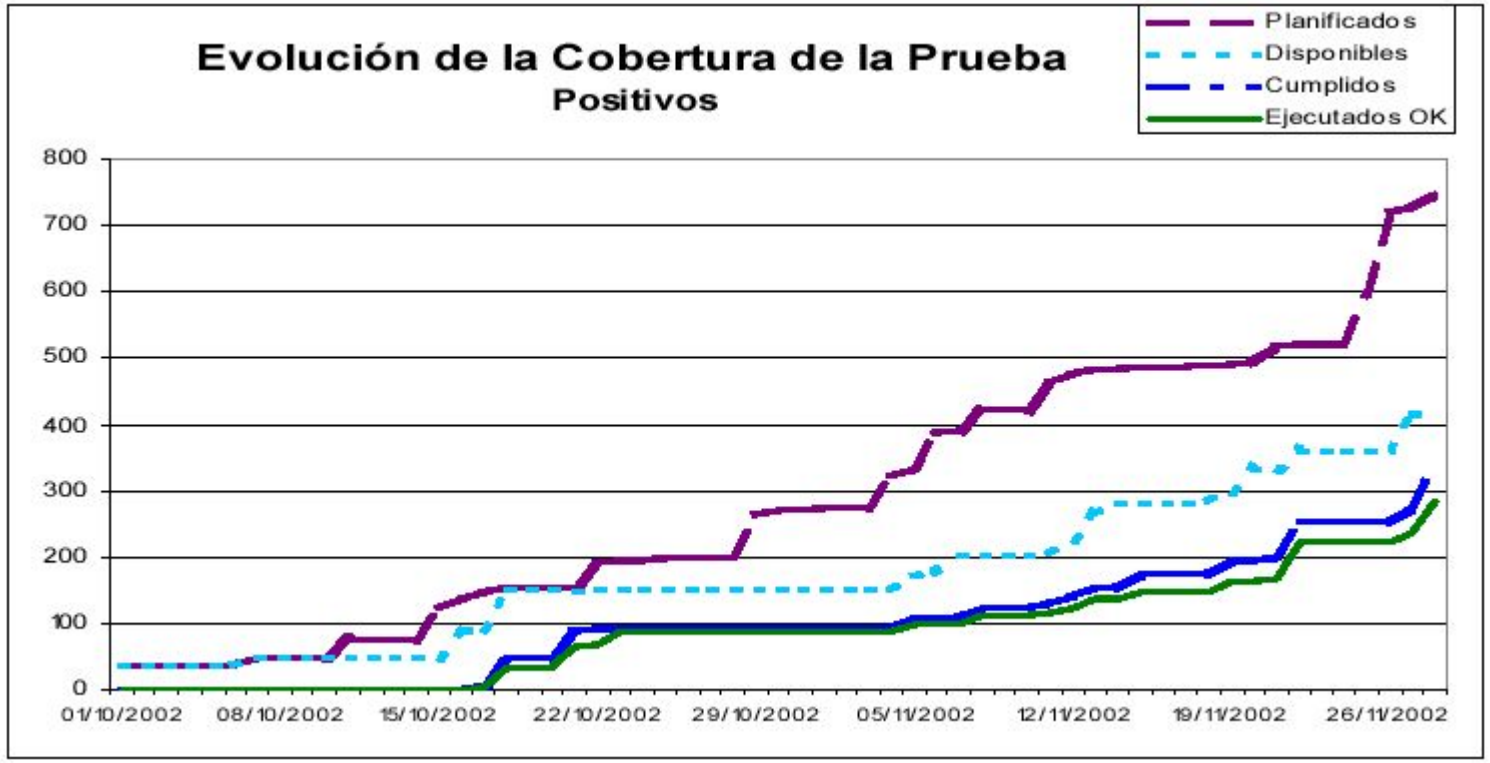
COBERTURA DE LA PRUEBA

COBERTURA DE LA PRUEBA

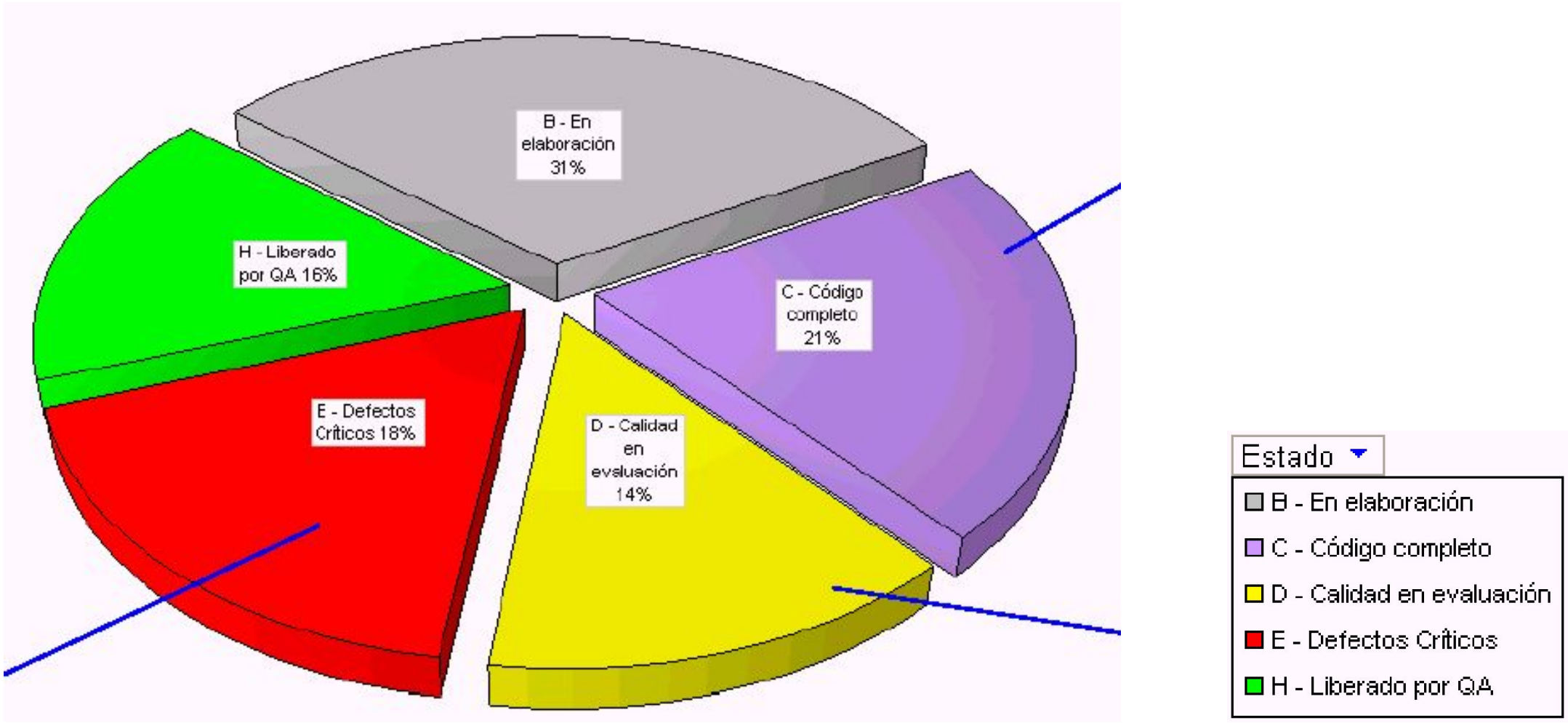
- ¿Cómo obtenerlo?
  - Planificar los casos de prueba.
  - Registrar los casos de prueba.
  - Registrar el resultado de la ejecución de las pruebas.

COBERTURA DE LA PRUEBA





COBERTURA DE LA PRUEBA



COBERTURA DE LA PRUEBA

Muestra estados detallados de cada funcionalidad.

Muestra el nivel de calidad del producto

Brinda información para tomar medidas correctivas:

¿Faltan recursos de QA?
¿La calidad del producto es mala?
¿Cuál es el estado del producto X ?

GRACIAS!