Especialización en sistemas embebidos (CESE)

Testing de software embebido

Clase 1

Laboratorio de Sistemas Embebidos

Colaboradores y Docentes

A continuación se lista los nombres de las personas que han colaborado tanto en el armado y/o en el dictado de esta materia (por orden alfabético de apellidos):

- Ing. Alejandro Permingeat <apermingeat@gmail.com>
- Ing. Esteban Volentini <evolentini@gmail.com>

https://sites.google.com/site/testingdesoftwareembebido/programa-materia

Clase 1

- Introducción al testing de software
- Master Test Plan

ing. Alejandro Permingeat

https://sites.google.com/site/testingdesoftwareembebido/programa-materia

Introducción al testing de SW

Testing es el proceso centrado en el objetivo de encontrar defectos en un sistema.

Por razones de depuración

evitar que el software haga lo que deba hacer incorrectamente

Por razones de aceptación

evitar que el software haga correctamente algo que no debe hacer

Introducción al testing de SW

Mejor prevenir que corregir bugs

Aseguramiento de Calidad (QA)

Asegurando que se cumple el proceso

Control de calidad (QC): detectar y corregir bugs

Ejecutando pruebas y haciendo seguimiento de corrección de bugs

Objetivo del testing

Proporcionar una visión clara de los riesgos asociados para la organización

las debilidades observadas en el software

los riesgos asociados para la organización

No mejora la calidad del software en forma directa, pero si de manera indirecta

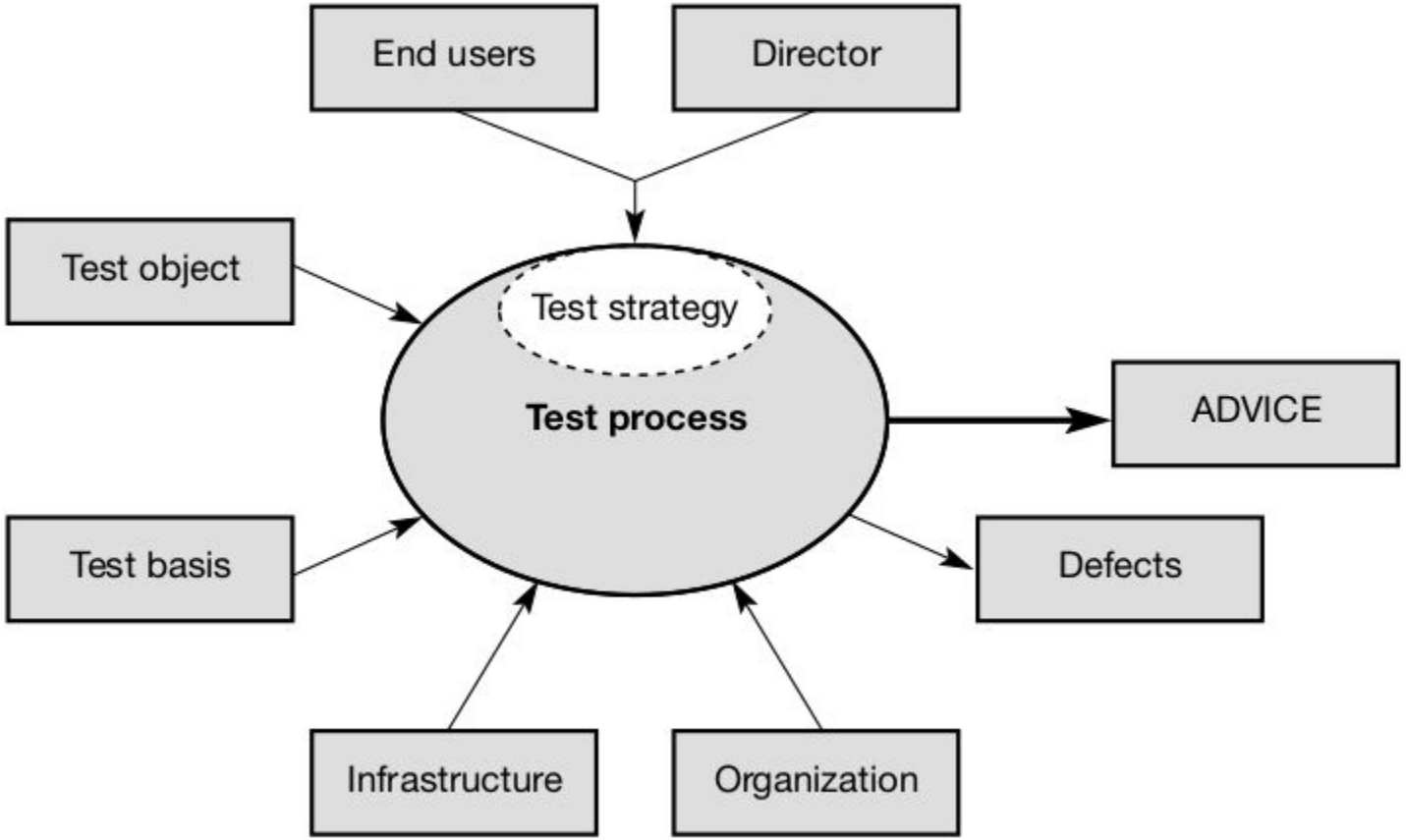
Ejemplo: ECU automovil

- ¿Los frenos se activan en menos de 100 ms?
- ¿El motor enciende luego de 200.000 encendidos?
- ¿Las puertas se destraban luego de un choque frontal?
- ¿El motor regula a mayor RPMs cuando hace frío?
- ¿Se puede accionar el levanta vidrios en los 4 cristales al mismo tiempo?
- ¿El sistema de climatización del habitáculo continúa funcionando aún con el vehículo con nivel de batería baja?

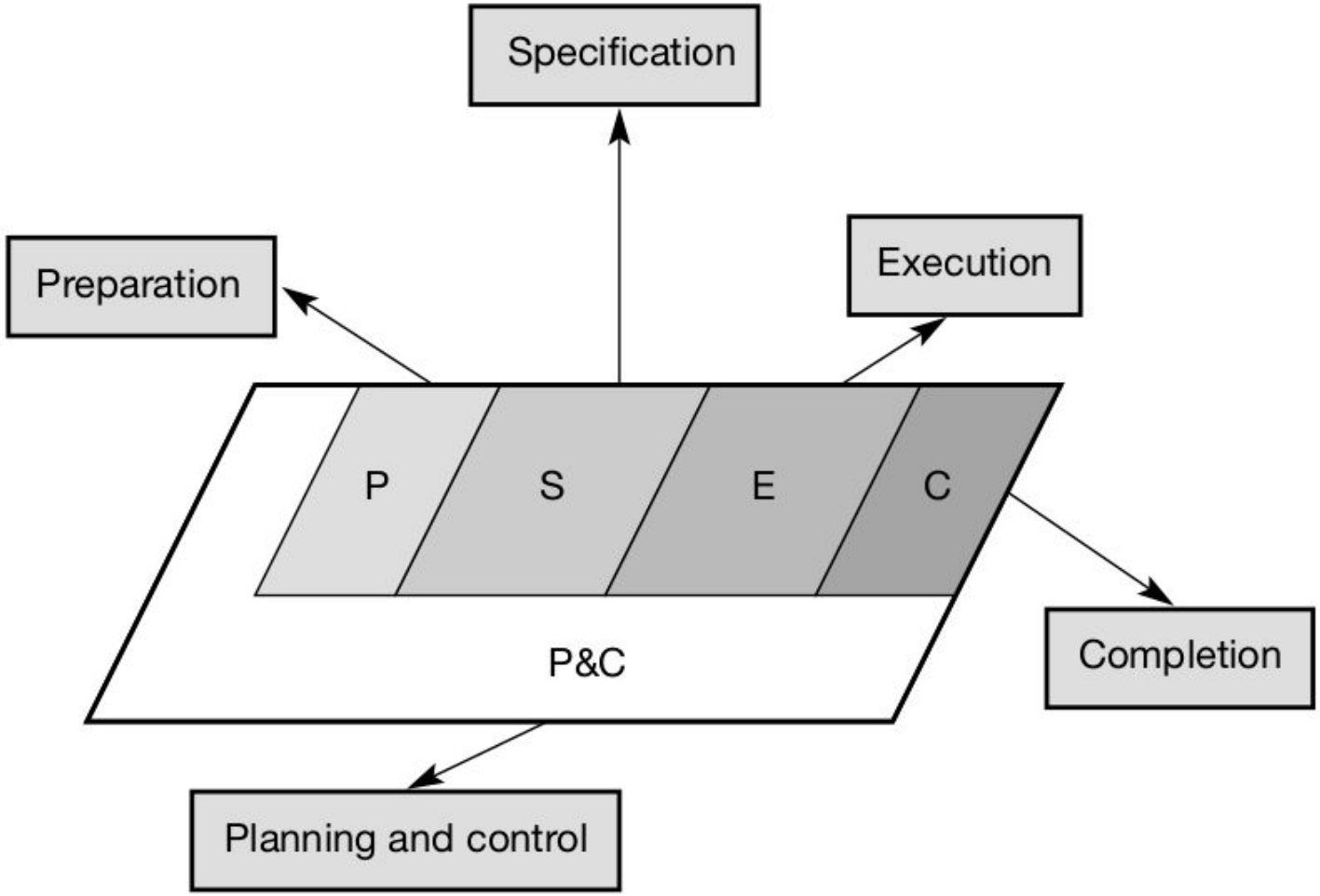
Ejemplo: ECU automovil

- Se debe tener información: ***Test basis***
- Qué es lo más importante a probar y en qué profundidad: ***Estrategia de prueba***
- Dependiendo la severidad de defectos encontrados se debe confeccionar una **Evaluación de la calidad** y dar **consejos** acerca de **cómo proceder**
- Se debe contar con una **Infraestructura adecuada**
- Muchas veces se debe contar con soporte de especialistas en la **organización**

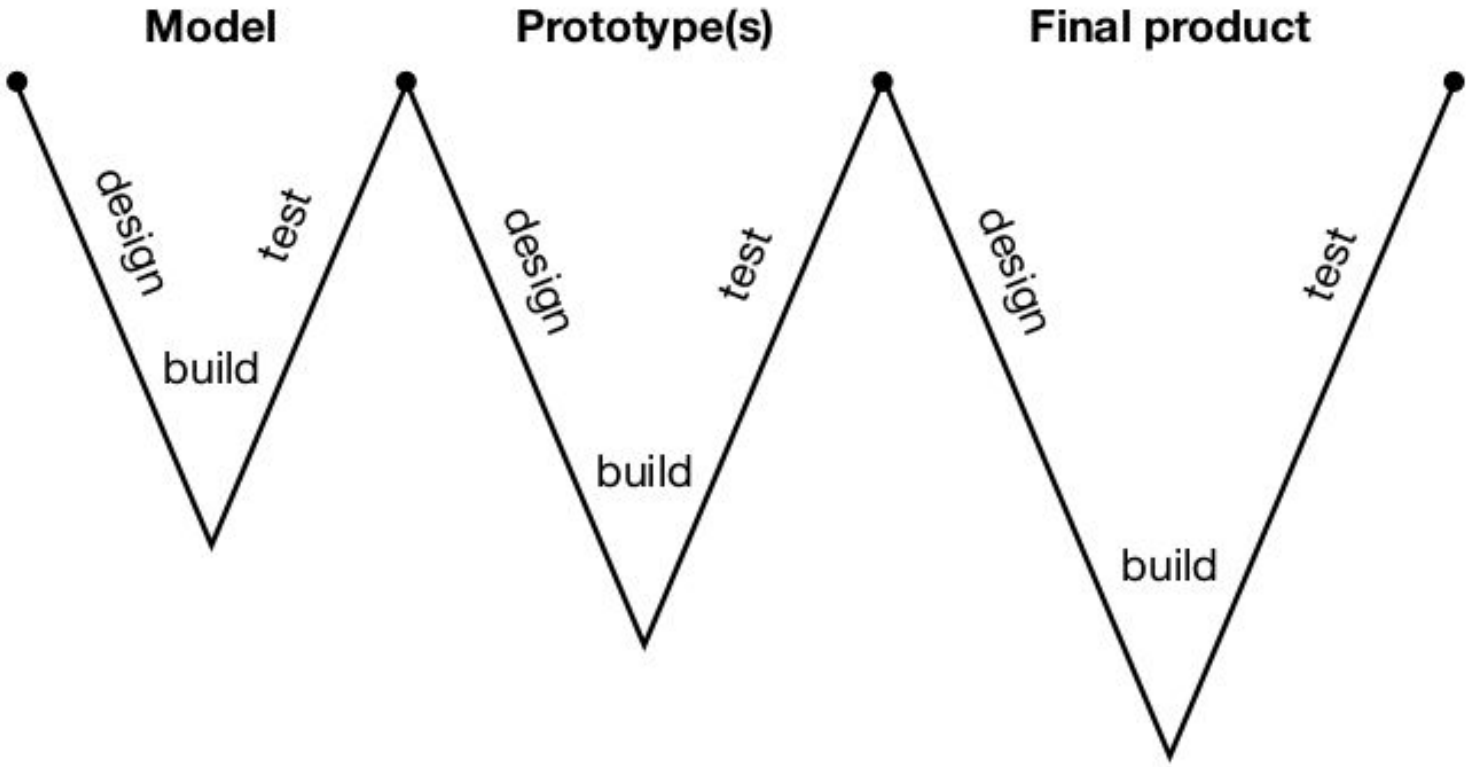
Ejemplo: ECU automovil



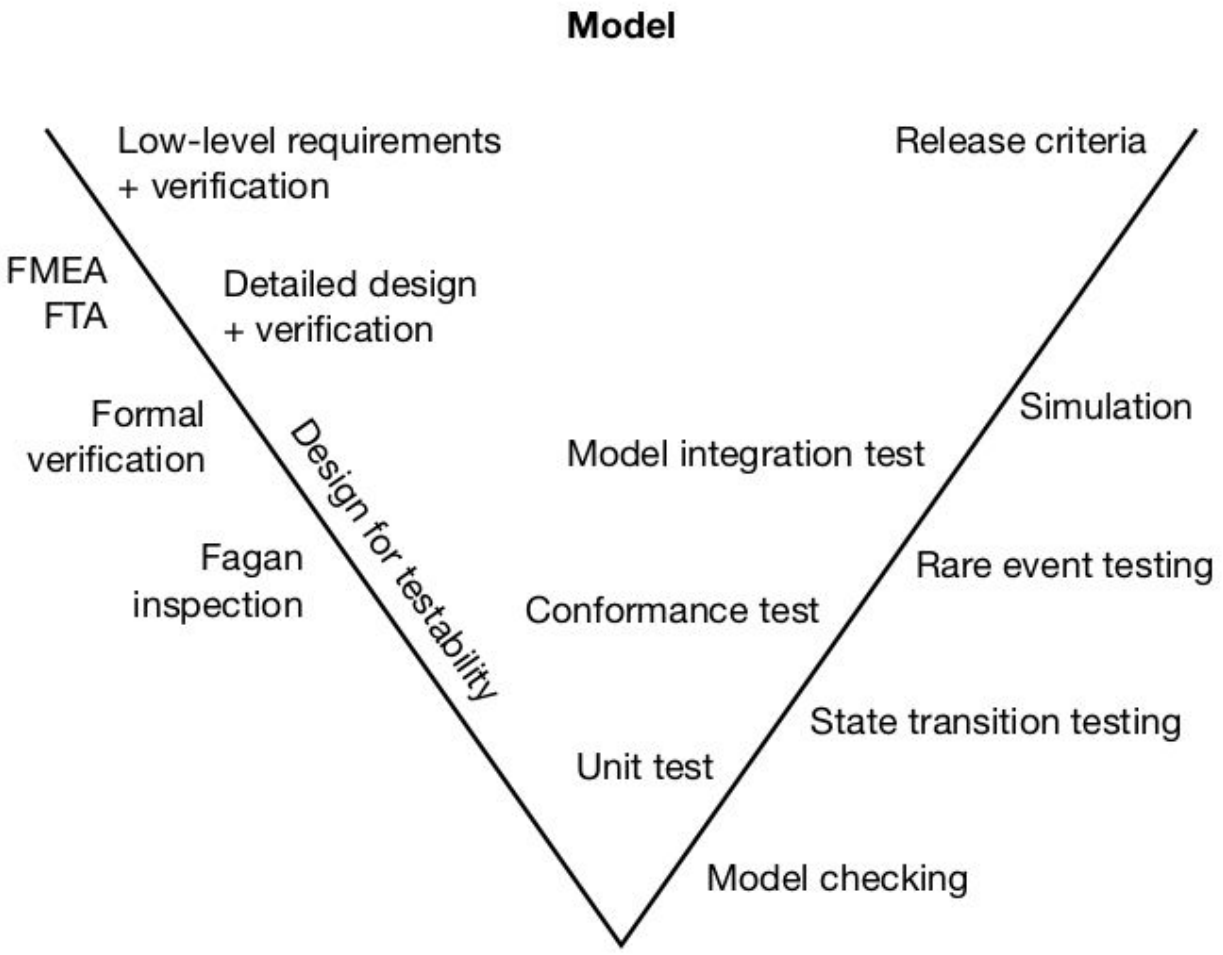
Ciclo de vida del test



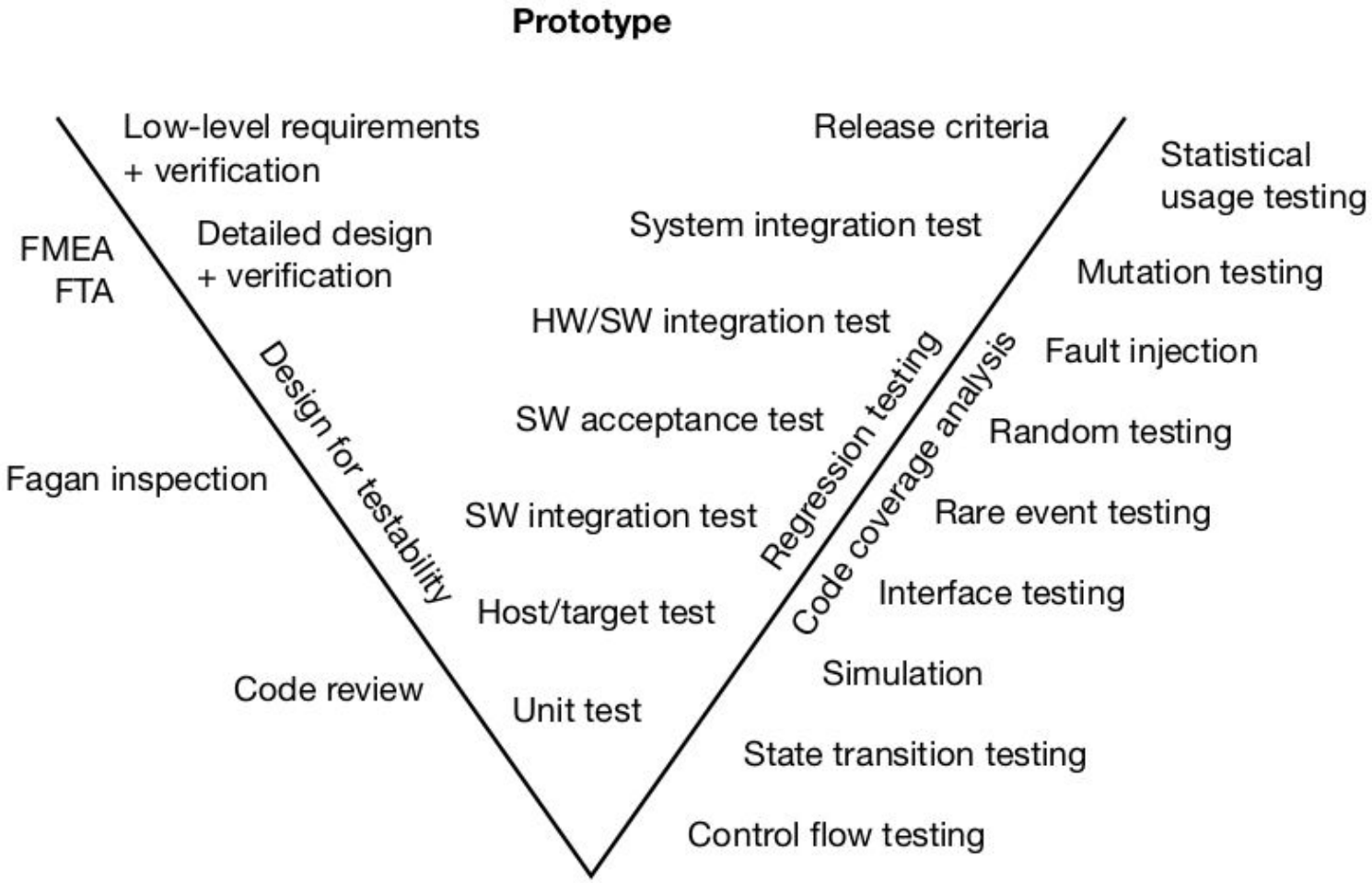
Ciclo de vida - V múltiple



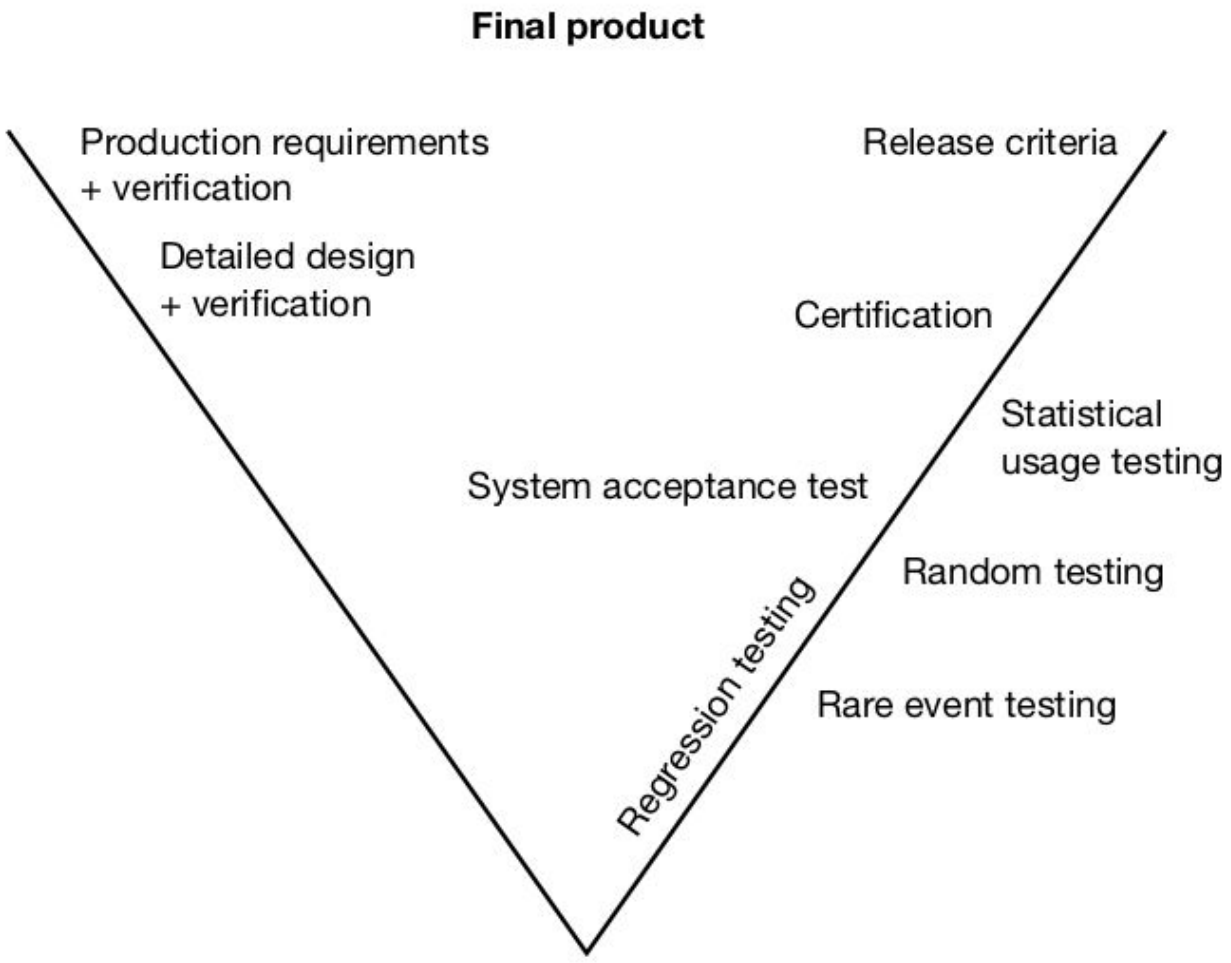
Ciclo de vida - V múltiple



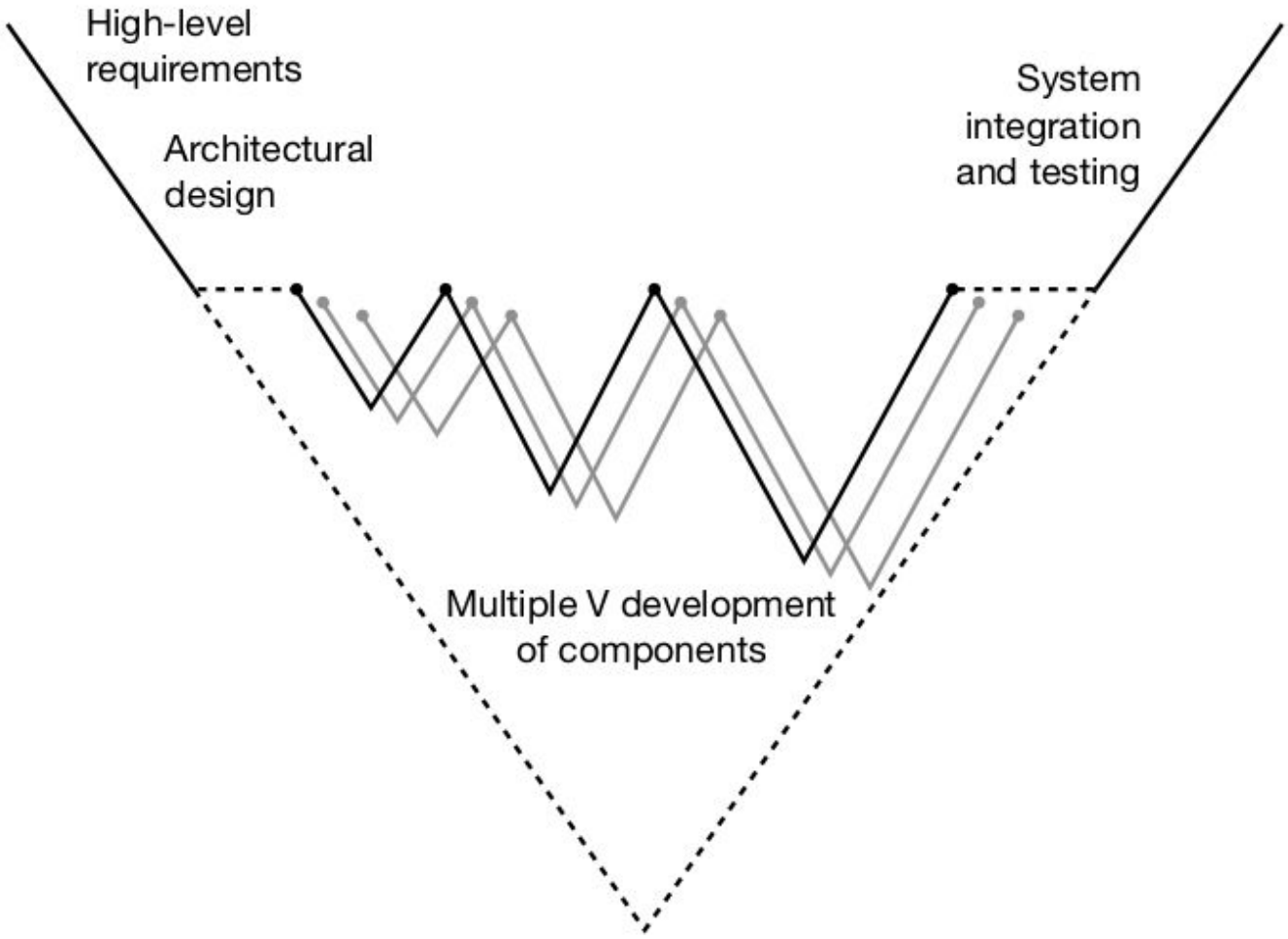
Ciclo de vida - V múltiple



Ciclo de vida - V múltiple



Ciclo de vida - V múltiple iterativo



Caso de prueba (test case)

ISO/IEC/ IEEE 24765 item 3.3064

conjunto de

- test inputs
- condiciones de ejecución (procedimiento)
- resultados esperado

Desarrollado con un objetivo particular como:

- ejercitar una ruta (path) de programa
- verificar el cumplimiento de un requisito específico

Técnicas diseño casos de prueba

- State transition testing (STT)
- Control flow test (CFT)
- Elementary comparison test (ECT)
- Classification-tree method (CTM)
- Evolutionary algorithms (EA)
- Statistical usage testing (SUT)
- Rare event testing (RET)
- Mutation analysis (MA)

Se abordarán en detalle en próximas clases

¿Cómo abordar el testing?

"Es imposible encontrar todos los defectos."

"Nunca hay suficiente tiempo (o personal o dinero) para probar todo."

Se deben tomar decisiones sobre cómo distribuir sabiamente los recursos disponibles

Planificación

Tipos de prueba

Un tipo de prueba es un grupo de actividades que se realizan con el objetivo de evaluar un sistema teniendo en cuenta un conjunto de atributos de calidad relacionados.

Los tipos de prueba indican qué se va a probar (y qué no).

Test type	Description	Quality characteristics included
Functionality	Testing functional behavior (includes dealing with input errors)	Functionality
Interfaces	Testing interaction with other systems	Connectivity
Load and stress	Allowing large quantities and numbers to be processed	Continuity, performance
Support (manual)	Providing the expected support in the system's intended environment (such as matching with the user manual procedures)	Suitability
Production	Test production procedures	Operability, continuity
Recovery	Testing recovery and restart facilities	Recoverability
Regression	Testing whether all components function correctly after the system has been changed	All
Security	Testing security	Security
Standards	Testing compliance to standards	Security, user-friendliness
Resources	Measuring the required amount of resources (memory, data communication, power, ...)	Efficiency

Nivel de prueba

Un nivel de prueba es un grupo de actividades que se organiza y gestiona como una entidad

Los niveles de prueba indican quién va a realizar las pruebas y cuándo.

Test level	Level	Environment	Purpose
Hardware unit test	Low	Laboratory	Testing the behavior of hardware component in isolation
Hardware integration test	Low	Laboratory	Testing hardware connections and protocols
Model in the loop	High/low	Simulation models	Proof of concept; testing control laws; design optimization
Software unit test, host/target test	Low	Laboratory, host + target processor	Testing the behavior of software components in isolation
Software integration test	Low	Laboratory, host + target processor	Testing interaction between software components
Hardware/software integration test	High	Laboratory, target processor	Testing interaction between hardware and software components
System test	High	Simulated real life	Testing that the system works as specified
Acceptance test	High	Simulated real life	Testing that the system fulfills its purpose for the user/customer
Field test	High	Real life	Testing that the system keeps working under real-life conditions.

Planificación de testing

El master test plan consiste en la combinación de

Tipos de prueba

***qué** se va a probar*

Niveles de prueba

***quién** y **cuándo** va a probar*

Master Test Plan

Las áreas de mayor interés en el master test plan son:

- elección de estrategia de test (que probar y como)
- asignación de recursos escasos
- comunicación entre las disciplinas involucradas

Estrategias en master test plan

Concientizar a toda la organización de los riesgos que deben cubrirse y acordar cuantas pruebas deben realizarse, y cuando dentro del proceso de desarrollo de software

1. Seleccionar características de calidad
2. Determinar la importancia relativa de dichas características
3. Asignar las características de calidad a los niveles de prueba

1. Seleccionar características de calidad

ISO/IEC 9126

Characteristic	Description
Functionality	The capability of the software to provide functions which meet the stated and implied needs of users under specified conditions of usage (what the software does to meet needs)
Reliability	'The capability of the software product to maintain its level of performance under stated conditions for a stated period of time.'
Usability	The capability of the software product to be understood, learned, used and provide visual appeal, under specified conditions of usage (the effort needed for use)
Efficiency	The capability of the software product to provide desired performance, relative to the amount of resources used, under stated conditions.
Maintainability	The capability of the software product to be modified which may include corrections, improvements or adaptations of the software to changes in the environment and in the requirements and functional specifications (the effort needed for modification)
Portability	The capability of the software product to be 'transferred from one environment to another. The environment may include organizational, hardware or software.'

Characteristic	Sub Characteristics	Explanation
Functionality	Suitability	'Can software perform the tasks required?'
	Accurateness	'Is the result as expected?'
	Interoperability	'Can the system interact with another system?'
	Compliance	'Is the system compliant with standards?'
	Security	'Does the system prevent unauthorized access?'
Reliability	Maturity	'Have most of the faults in the software been eliminated over time?'
	Fault tolerance	'Is the software capable of handling errors?'
	Recoverability	'Can the software resume working & restore lost data after failure?'
Usability	Understandability	'Does the user comprehend how to use the system easily?'
	Learnability	'Can the user learn to use the system easily?'
	Operability	'Can the user use the system without much effort?'
	Attractiveness	'Does the interface look good?'
Efficiency	Time Behaviour	'How quickly does the system respond?'
	Resource utilization	'Does the system utilize resources efficiently?'
Maintainability	Analyzability	'Can faults be easily diagnosed?'
	Changeability	'Can the software be easily modified?'
	Stability	'Can the software continue functioning if changes are made?'
	Testability	'Can the software be tested easily?'
Portability	Adaptability	'Can the software be moved to other environments?'
	Installability	'Can the software be installed easily?'
	Conformance	'Does the software comply with portability standards?'
	Replaceability	'Can the software easily replace other software?'

2. Determinar la importancia relativa de dichas características

Quality characteristic	Relative importance (%)
Connectivity	10
Efficiency (memory)	–
Functionality	40
Maintainability	–
Performance	15
Recoverability	5
Reliability	10
Security	–
Suitability	20
Usability	–
Total	100

3. Asignar las características de calidad a los niveles de prueba

	Functionality	Connectivity	Reliability	Recoverability	Performance	Suitability
Relative importance (%)	40	10	10	5	15	20
Unit test	++			+		
Software integration test	+	++				
Hardware/software integration test	+	++		++		
System test	++		+		+	
Acceptance test	+			++		++
Field test			++		++	

++	the quality characteristic will be covered thoroughly – it is a major goal in this test level
+	this test level will cover this quality characteristic
(empty)	this quality characteristic is not an issue in this test level

Estrategia para un nivel de prueba

1. Seleccionar características de calidad
2. Determinar la importancia relativa de dichas características
3. Dividir el sistema en subsistemas
4. Determinar la importancia relativa de los subsistema
5. Determinar la importancia de test por combinaciones de subsistema / característica de calidad
6. Establecer las técnicas de test a ser usadas

Estrategia para un nivel de prueba

1. Seleccionar características de calidad

Igual al paso 1 de la estrategia del master test plan

2. Determinar la importancia relativa de dichas características

Igual al paso 2 de la estrategia del master test plan

Estrategia para un nivel de prueba

3. Dividir el sistema en subsistemas

El sistema se divide en subsistemas que se pueden probar por separado

También se puede pensar en "componentes" o "unidades funcionales" u otros términos similares.

En general se sigue la división en componentes que se realiza en el diseño arquitectónico del software

Estrategia para un nivel de prueba

4. Determinar la importancia relativa de los subsistema

Subsystem	Relative importance (%)
Part A	30
Part B	10
Part C	30
Part D	5
Total system	25
Total	100

Estrategia para un nivel de prueba

5. Determinar la importancia de test por combinaciones de subsistema / característica de calidad

Relative importance (%)		Part A	Part B	Part C	Part D	Total system
	100	30	10	30	5	25
Functionality	40	++	+	+	+	+
Performance	25	+	++	+		+
Reliability	10		+			++
Suitability	25	+		+	+	++

++	the quality characteristic is predominant for this subsystem
+	the quality characteristic is relevant for this subsystem
(empty)	this quality characteristic is insignificant for this subsystem

Estrategia para un nivel de prueba

6. Establecer las técnicas de test a ser usadas

Applied test technique	Part A	Part B	Part C	Part D	Total system
W	+	+	+		
X		+	+		
Y	+			+	+
Z					+

TP 1: Master test plan

Confeccionar el master test plan para el software que formará parte del TP final de la carrera de especialización.

Considerar como referencia el ejemplo de Master Test Plan del Appendix E del libro: “Testing Embedded Software” de Bart Broekman y Edwin Notenboom

Bibliografía

- “Testing Embedded Software”. Autores: Bart Broekman y Edwin Notenboom. Año 2003.
- ISO 9126 external systems quality characteristics, sub-characteristics and domain specific criteria for evaluating e-Learning systems. Autores: I. Padayachee, P. Kotze, A. van Der Merwe