

# **Especialización en sistemas embebidos (CESE)**

## **Testing de software embebido Clase 2**



# 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](mailto:apermingeat@gmail.com)>
- Ing. Esteban Volentini <[evolentini@gmail.com](mailto:evolentini@gmail.com)>


# Clase 2

- Pruebas a nivel sistemas
- Pruebas de aceptación
- Ejemplos de técnicas de diseño de test:
  - Classification-tree method (CTM)
  - Elementary comparison test (ECT)

ing. Alejandro Permingeat



# REPASO



¿tipo de prueba  
vs  
nivel de prueba?

# Tipos de prueba vs nivel prueba

## Tipos de prueba



qué se va a probar



características de calidad  
(funcionalidad, interfaces,  
rendimiento . . .)

## Nivel de prueba



quién lo va a probar  
y cuándo



nivel de detalle.  
(pruebas unitarias, de  
sistema, de aceptación)

<b>Test type</b>	<b>Description</b>	<b>Quality characteristics included</b>
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

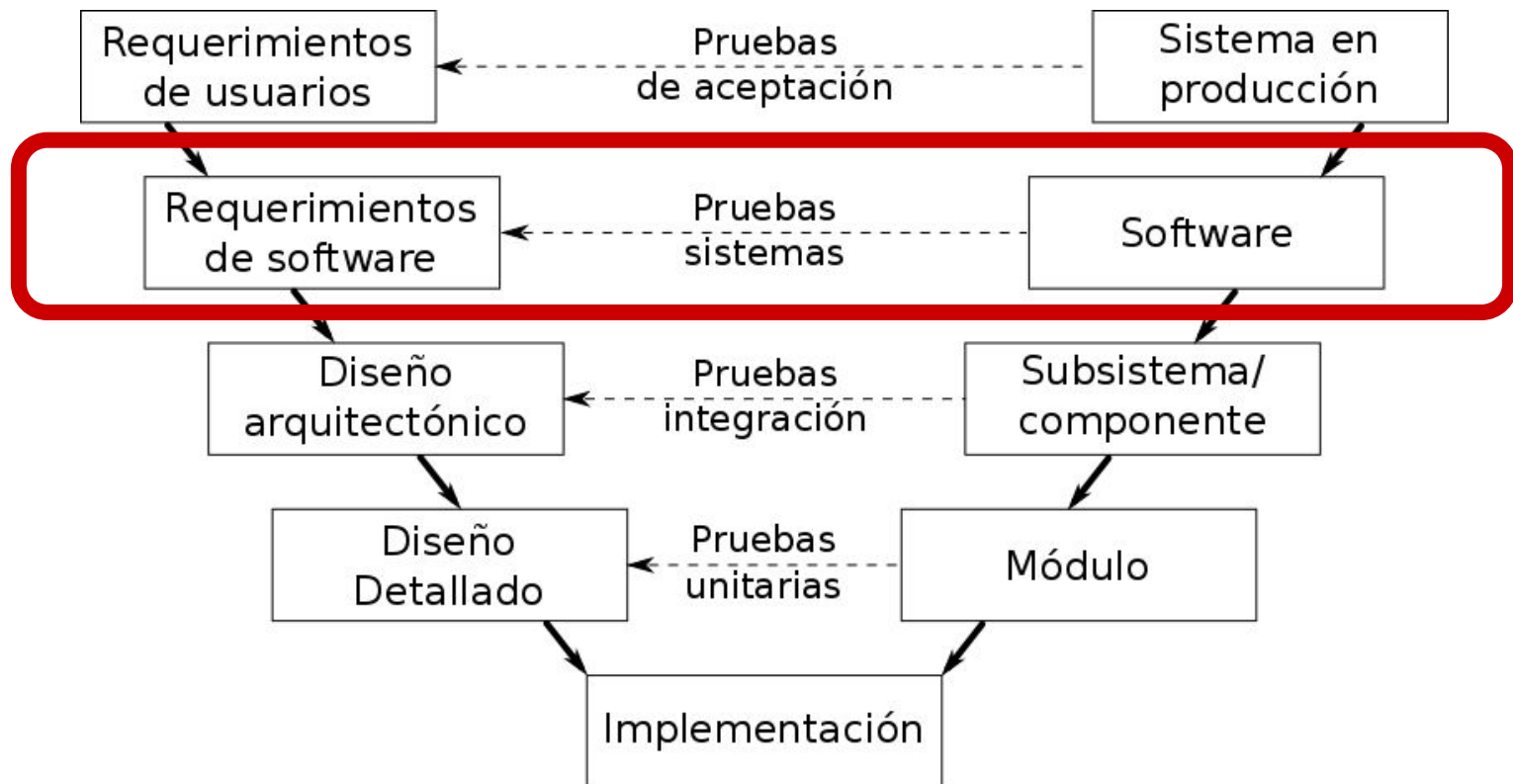
<b>Test level</b>	<b>Level</b>	<b>Environment</b>	<b>Purpose</b>
Hardware unit test	Low	Laboratory	Testing the behavior of hardware component in isolation
Hardware integration test	Low	Laboratory	Testing hardware connections and protocols
Model in the loop	High/low	Simulation models	Proof of concept; 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.



<b>Test level</b>	<b>Level</b>	<b>Environment</b>	<b>Purpose</b>
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.

# Pruebas a nivel sistemas

Conjunto de pruebas realizadas con el fin de probar que se cumplen todos los requerimientos



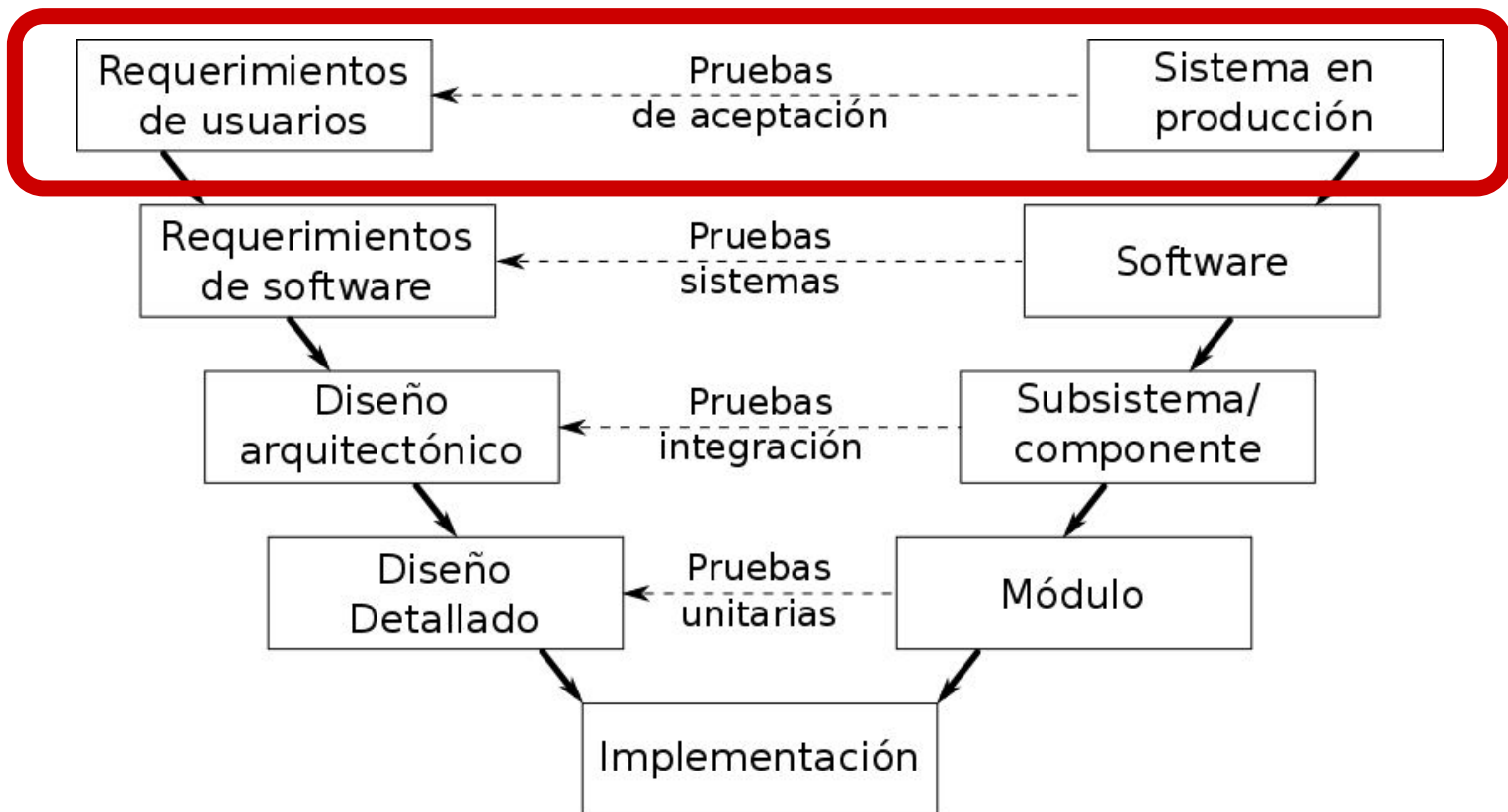
# Pruebas a nivel sistemas

## *Cobertura*

- Cada requerimiento de software debe tener al menos una prueba de sistema
- Muchas veces con una sola prueba no se cubre todo el requerimiento

# Pruebas de aceptación

Pruebas formales realizadas para permitir que un usuario, cliente o entidad autorizada decida si acepta un sistema o componente.



# Pruebas de aceptación

Muchas veces son un subconjunto de las  
*pruebas de sistema*



¿Diseño de testing?

# 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)

# 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)



# Técnicas diseño casos de prueba

## **Classification-tree method (CTM)**

- El método de árbol de clasificación (CTM) es utilizado en el diseño sistemático de casos de prueba de caja negra.
- El dominio de entrada del objeto de prueba se divide en clases disjuntas y se representa gráficamente en forma de árbol
- CTM utiliza los requerimientos funcionales como base de la prueba.

# Técnicas diseño casos de prueba

## **Classification-tree method (CTM)**

Pasos:

1. Identificar aspectos del objeto de prueba;
2. Dividir el dominio de entrada de acuerdo con los aspectos;
3. Especificando casos de prueba lógicos;
4. Ensamblando el script de prueba.

# Técnicas diseño casos de prueba

## **Classification-tree method (CTM)**

Ejemplo. Sistema control de crucero (simplificado).

Se consideran:

- la velocidad del auto y la diferencia entre la velocidad actual y la deseada
- la diferencia entre la velocidad del auto de adelante y la velocidad del auto con el control de crucero
- la distancia entre el auto de adelante y el auto con control crucero

# Técnicas diseño casos de prueba

## **Classification-tree method (CTM)**

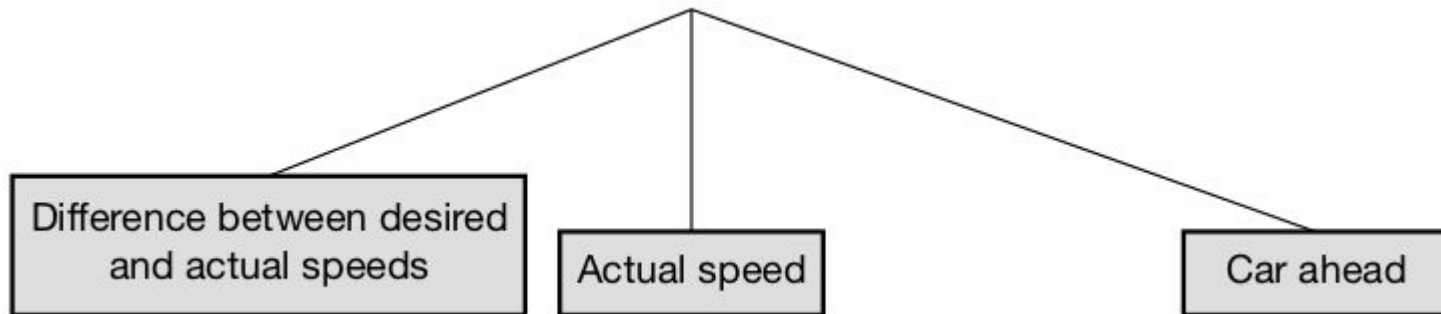
Requerimientos:.

- se puede activar una velocidad crucero entre un valor mayor a 40 km/h y hasta 90 km/h
- la diferencia entre la velocidad actual y la deseada no debe superar los 30 km/h
- la diferencia entre la velocidad del auto de adelante y la velocidad del auto con el control de crucero genera una alarma si está entre 0 y 30 km/h y desactiva el control crucero si el auto de adelante va más despacio que el auto con control crucero
- la distancia entre el auto de adelante y el auto con control crucero nunca debe ser menor a 25 m

# Técnicas diseño casos de prueba

## Classification-tree method (CTM)

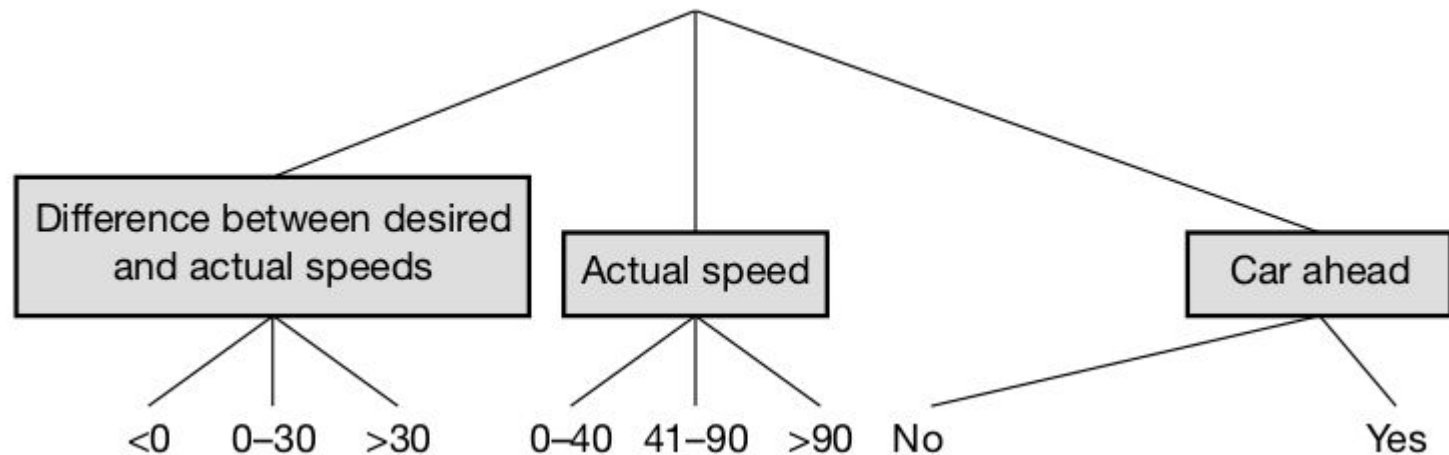
1. Identificar aspectos del objeto de prueba;



# Técnicas diseño casos de prueba

## Classification-tree method (CTM)

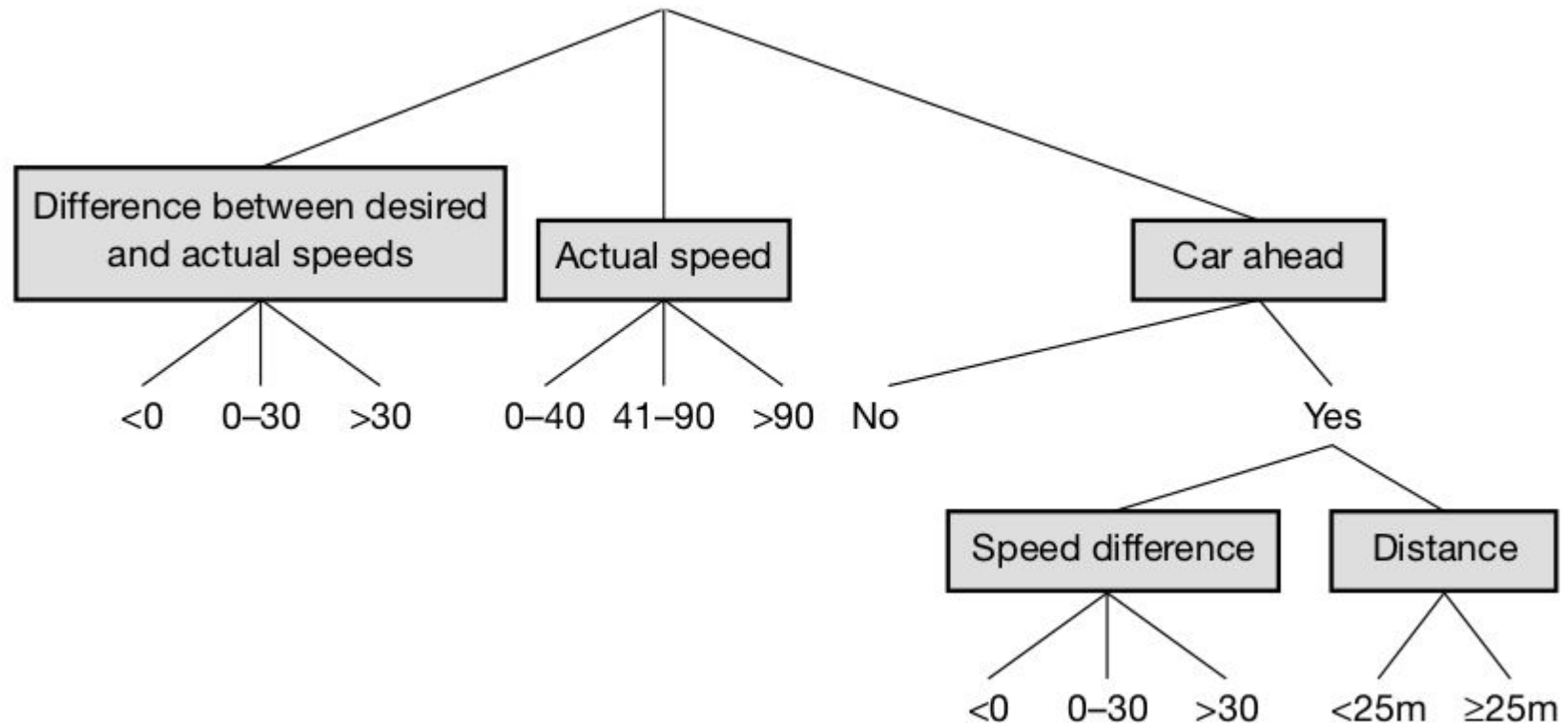
2. Dividir el dominio de entrada de acuerdo con los aspectos;



# Técnicas diseño casos de prueba

## Classification-tree method (CTM)

2. Dividir el dominio de entrada de acuerdo con los aspectos (**recursivo**)



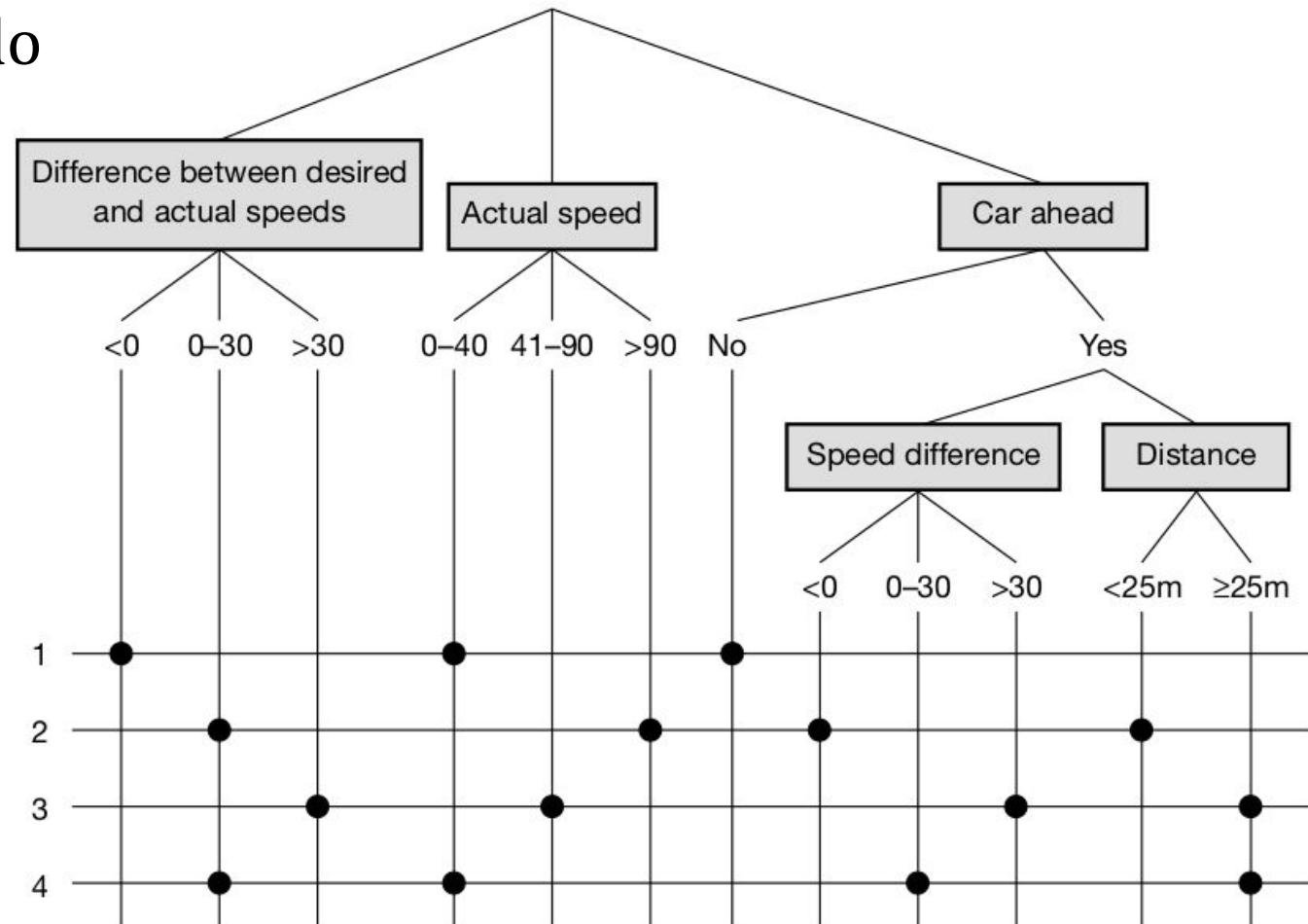
# Técnicas diseño casos de prueba

## Classification-tree method (CTM)

3. Especificando  
casos de  
prueba  
lógicos

mínimo:  
4 casos

máximo:  
63 casos!!!  
( $3 \times 3 \times (1 + 2 \times 3)$ )





Test case	Aspect	Value	Comment
1	Speed of car	39 km/h	
	Desired speed	30 km/h	Difference between desired/actual speed: -9 km/h
	Car ahead	No	
2	Speed of car	100 km/h	
	Desired speed	120 km/h	Difference between desired/actual speed: 20 km/h
	Speed of car ahead	90 km/h	Difference between car ahead/car: -10 km/h
	Distance	10 m	
3	Speed of car	65 km/h	
	Desired speed	135 km/h	Difference between desired/actual speed: 70 km/h
	Speed of car ahead	105 km/h	Difference between car ahead/car: 40 km/h
	Distance	100 m	
4	Speed of car	25 km/h	
	Desired speed	50 km/h	Difference between desired/actual speed: 25 km/h
	Speed of car ahead	55 km/h	Difference between car ahead/car: 30 km/h
	Distance	30 m	

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

- En la prueba de comparación elemental, el procesamiento es probado en detalle.
- Todas las condiciones funcionales tienen que ser identificadas y traducidas en pseudocódigo.
- Los casos de prueba se derivan del pseudocódigo y cubren los caminos funcionales identificados.
- Debido a que esta técnica formal es de mano de obra intensiva, se utiliza principalmente en funciones muy importantes y / o cálculos complejos.

# Técnicas diseño casos de prueba

## **Elementary comparison test (ECT)**

- El diseño detallado se utiliza como base de la prueba.
- Debe contener una descripción de la estructura del algoritmo bajo prueba - pseudocódigo, un diagrama de flujo, una tabla de decisiones, diagramas de actividad, etc.
- Si la base de prueba no proporciona esta información entonces puede ser necesario preparar documentación sobre la estructura del programa basada en la información disponible.

# Técnicas diseño casos de prueba

## **Elementary comparison test (ECT)**

Pasos:

1. Analizar la descripción de la función;
2. Establecer situaciones de prueba;
3. Establecer casos de prueba lógica;
4. Establecer casos de pruebas físicas;
5. Establecer acciones de prueba;
6. Establecer controles;
7. Establecer la situación de inicio;
8. Ensamblando el script de prueba.

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

```
IF TempSensor1 - TempSensor2 ≤ 40
THEN Actorheater_1 = OFF
ELSE
    IF TempSensor2 ≥ 70 AND TempSensor1 < 30 AND Actorvalve_1 = ON
        Actorvalve_2 = OFF
    ENDIF
    IF VolumeSensor < 1000 AND Actorvalve_2 = ON AND Actorheater_1
        = ON AND (Actorheater_2 = OFF OR TempSensor2 ≥ 50)
        Actorvalve_1 = OFF
    ENDIF
ENDIF
```

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

### 1. Analizar la descripción de la función;

```
C1 IF TempSensor1 - TempSensor2 ≤ 40
```

```
    THEN Actorheater1 = OFF
```

```
    ELSE
```

```
C2      IF TempSensor1 ≥ 70 AND TempSensor2 < 30 AND Actorvalve1 = ON
```

```
          Actorvalve2 = OFF
```

```
      ENDIF
```

```
C3      IF VolumeSensor < 1000 AND Actorvalve2 = ON AND Actorheater1 = ON  
        AND (Actorheater2 = OFF OR TempSensor1 ≥ 50)
```

```
          Actorvalve1 = OFF
```

```
      ENDIF
```

```
ENDIF
```

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

2. Establecer situaciones de prueba;

### Comparación simple

Test situation	1	2
C1	true	false
$\text{TempSensor}_1 - \text{TempSensor}_2$	$\leq 40$	$> 40$

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

Comparación compuesta no combinada

Test situation	C2.1	C2.2	C2.3	C2.4
C2(=C2a & C2b & C2c)	1(111)	0(011)	0(101)	0(110)
C2a TempSensor <sub>1</sub> ≥ 70 AND	≥ 70	< 70	≥ 70	≥ 70
C2b TempSensor <sub>2</sub> < 30 AND	< 30	< 30	≥ 30	< 30
C2c Actor <sub>valve1</sub> = ON	ON	ON	ON	OFF

- No se consideran todas las posibilidades
- Las situaciones de prueba se seleccionan de manera que un cambio en el valor de cualquiera de las comparaciones elementales cambie el valor de la condición compleja



# Técnicas diseño casos de prueba

## Elementary comparison test (ECT) Comparación compuesta combinada

<b>C3 (=C3a &amp; C3b &amp; C3c &amp; (C3d  C3e)</b>	<b>1 (Actor<sub>valve2</sub> = OFF)</b>	<b>0</b>
VolumeSensor < 1000 (C3a)	<u>1</u> .1.1.01	0. <u>1</u> .1.01
Actor <sub>valve2</sub> = ON (C3b)	1. <u>1</u> .1.01	1. <u>0</u> .1.01
Actor <sub>heater1</sub> = ON (C3c)	1.1. <u>1</u> .01	1.1. <u>0</u> .01
Actor <sub>heater2</sub> = OFF (C3d)	1.1.1. <u>1</u> 0	1.1.1. <u>0</u> 0
TempSensor ≥ 50 (C3e)	1.1.1.0 <u>1</u>	1.1.1.0 <u>0</u>

Primera columna se enumeran las comparaciones elementales.  
La segunda columna muestra como se puede llegar el valor de verdad “verdadero” para la condición compleja de tal manera que un cambio en el valor de la comparación elemental (su valor está subrayado) cambiaría el valor de la condición compleja a falso.

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

### Comparación compuesta combinada

<b>C3 (=C3a &amp; C3b &amp; C3c &amp; (C3d  C3e)</b>	<b>1 (Actor<sub>valve2</sub> = OFF)</b>	<b>0</b>
VolumeSensor < 1000 (C3a)	<u>1</u> .1.1.01	<u>0</u> .1.1.01
Actor <sub>valve2</sub> = ON (C3b)	<del>1</del> . <del>1</del> .1.01	1. <u>0</u> .1.01
Actor <sub>heater1</sub> = ON (C3c)	<del>1</del> . <del>1</del> . <u>1</u> .01	1.1. <u>0</u> .01
Actor <sub>heater2</sub> = OFF (C3d)	1.1.1. <u>1</u> 0	1.1.1. <u>0</u> 0
TempSensor <sub>1</sub> ≥ 50 (C3e)	<del>1</del> . <del>1</del> .1. <u>0</u> 1	<del>1</del> . <del>1</del> .1. <u>0</u> 0

Se eliminan los casos repetidos

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

### Comparación compuesta combinada

Test situation	C3.1	C3.2	C3.3	C3.4	C3.5	C3.6
C3	1(11101)	1(11110)	0(01101)	0(10101)	0(11001)	0(11100)
VolumeSensor	< 1000	< 1000	≥ 1000	< 1000	< 1000	< 1000
Actor <sub>valve2</sub>	ON	ON	ON	OFF	ON	ON
Actor <sub>heater1</sub>	ON	ON	ON	ON	OFF	ON
Actor <sub>heater2</sub>	ON	OFF	ON	ON	ON	ON
TempSensor <sub>1</sub>	≥ 50	< 50	≥ 50	≥ 50	≥ 50	< 50

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

### 3. Establecer casos de prueba lógica;

- Encontrar los caminos funcionales en los que cada situación de cada condición debe ser considerada al menos una vez.
- Tener en cuenta que las condiciones subsiguientes pueden influir entre sí. En este ejemplo, si la condición 2 es verdadera, esto cambiará la VALVE actor2 a OFF y esto automáticamente implicará que la condición 3 será falsa.

# Técnicas diseño casos de prueba

Test situation	Value	To	1	2	3	4	5	6	7	Control
C1.1	1	End	x							1
C1.2	0	C2		x	x	x	x	x	x	6
C2.1	1	C3		x						1
C2.2	0	C3			x			x		2
C2.3	0	C3				x			x	2
C2.4	0	C3					x			1
C3.1	1	End						x		1
C3.2	1	End			x					1
C3.3	0	End					x			1
C3.4	0	End		x						1
C3.5	0	End				x				1
C3.6	0	End							x	1

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

### 4. Establecer casos de pruebas físicas;

Test case	1	2	3	4	5	6	7
Path	C1.1	C1.2, C2.1, C3.4	C1.2, C2.2, C3.2	C1.2, C2.3, C3.5	C1.2, C2.4, C3.3	C1.2, C2.2, C3.1	C1.2, C2.3, C3.6
TempSensor <sub>1</sub>	50	90	45	90	90	60	90
TempSensor <sub>2</sub>	40	20	5	40	20	20	40
Actor <sub>valve1</sub>	OFF	ON	ON	ON	OFF	ON	ON
Actor <sub>valve2</sub>	OFF	ON	ON	OFF	ON	ON	ON
Actor <sub>heater1</sub>	OFF	ON	OFF	OFF	ON	ON	ON
Actor <sub>heater2</sub>	ON	ON	ON	ON	ON	ON	ON
VolumeSensor	500	500	500	500	1200	500	500

# Técnicas diseño casos de prueba

## Elementary comparison test (ECT)

### 6. Establecer controles;

Check	Test case	Expected situation
C01	1	Actor <sub>heater1</sub> = OFF and other actors unchanged
C02	2	Actor <sub>valve2</sub> = OFF and other actors unchanged
C03	3	Actor <sub>valve1</sub> = OFF and other actors unchanged
C04	4	All actors unchanged
C05	5	All actors unchanged
C06	6	Actor <sub>valve1</sub> = OFF and other actors unchanged
C07	7	All actors unchanged

# TP 2: Escribir ensayos de sistemas y de aceptación

Seleccionar un componente o subsistema del software que formará parte del TP final de la carrera de especialización y escribir ensayos de sistema (al menos 5).

Luego desarrollar para el mismo componente o subsistema al menos 2 ensayos de aceptación.

Los ensayos deberán ser escritos en un documento y entregado a la cátedra.



# Bibliografía

- “Testing Embedded Software”. Autores: Bart Broekman y Edwin Notenboom. Año 2003.