Winston W. Royce

JUICIO AL "CREADOR" DEL MODELO WATERFALL

Abogados Defensores

- Cecilia Sanchez
- Marta Ponzoni
- Matías Pérez
- Santiago Avendaño

El acusado

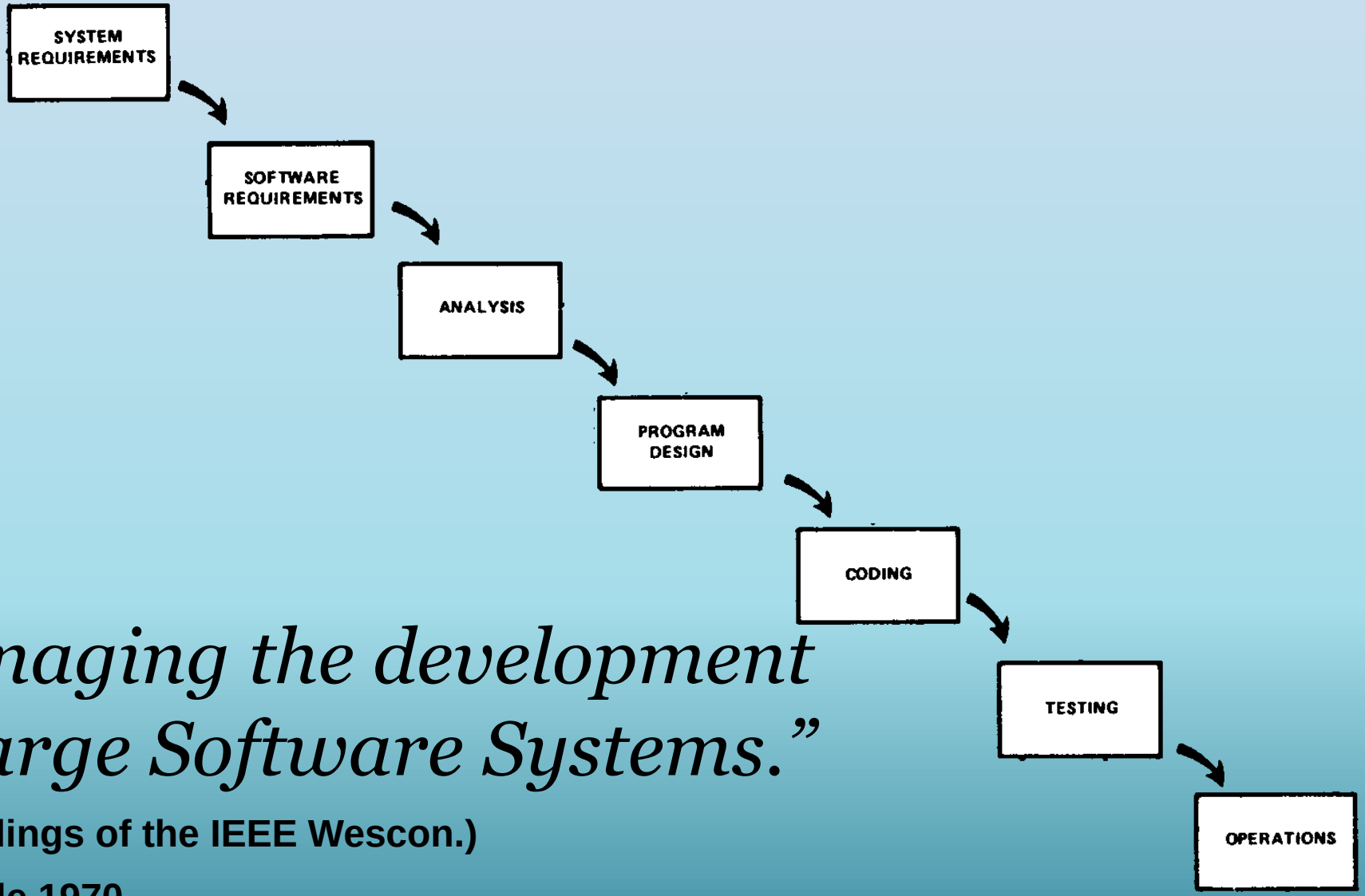
Winston W. Royce (1929-1995)

- Doctor en ingeniería aeronáutica
- Trabajó para la NASA (hasta 1970)
- Director del Lockheed Software Technology Center in Austin, Texas (desde 1970)
- Information Systems Award (AIAA, 1975)

El Contexto

- **Primera Fase. Los albores (1945-1955)**
  - **Programar no es una tarea diferenciada del diseño de una máquina**
  - **Uso de lenguaje máquina y ensamblador**
- **Segunda Fase. El florecimiento (1955-1965)**
  - **Aparecen multitud de lenguajes**
  - **"Todo es posible"**
- **Tercera Fase. La crisis (1965-1970)**
  - **Desarrollo inacabable de grandes programas Ineficiencia, errores, coste impredecible**
  - **"Nada es posible"**
- **Cuarta Fase. Innovación conceptual (1970-1980)**
  - **Fundamentos de programación**
  - **Verificación de programas**
  - **Metodologías de desarrollo**
- **Quinta Fase. El diseño es el problema (1980-199?)**
  - **Especificación formal**
  - **Programación automática**

El Delito



"Managing the development of Large Software Systems."

(Proceedings of the IEEE Wescon.)

Agosto de 1970

Imputados secundarios

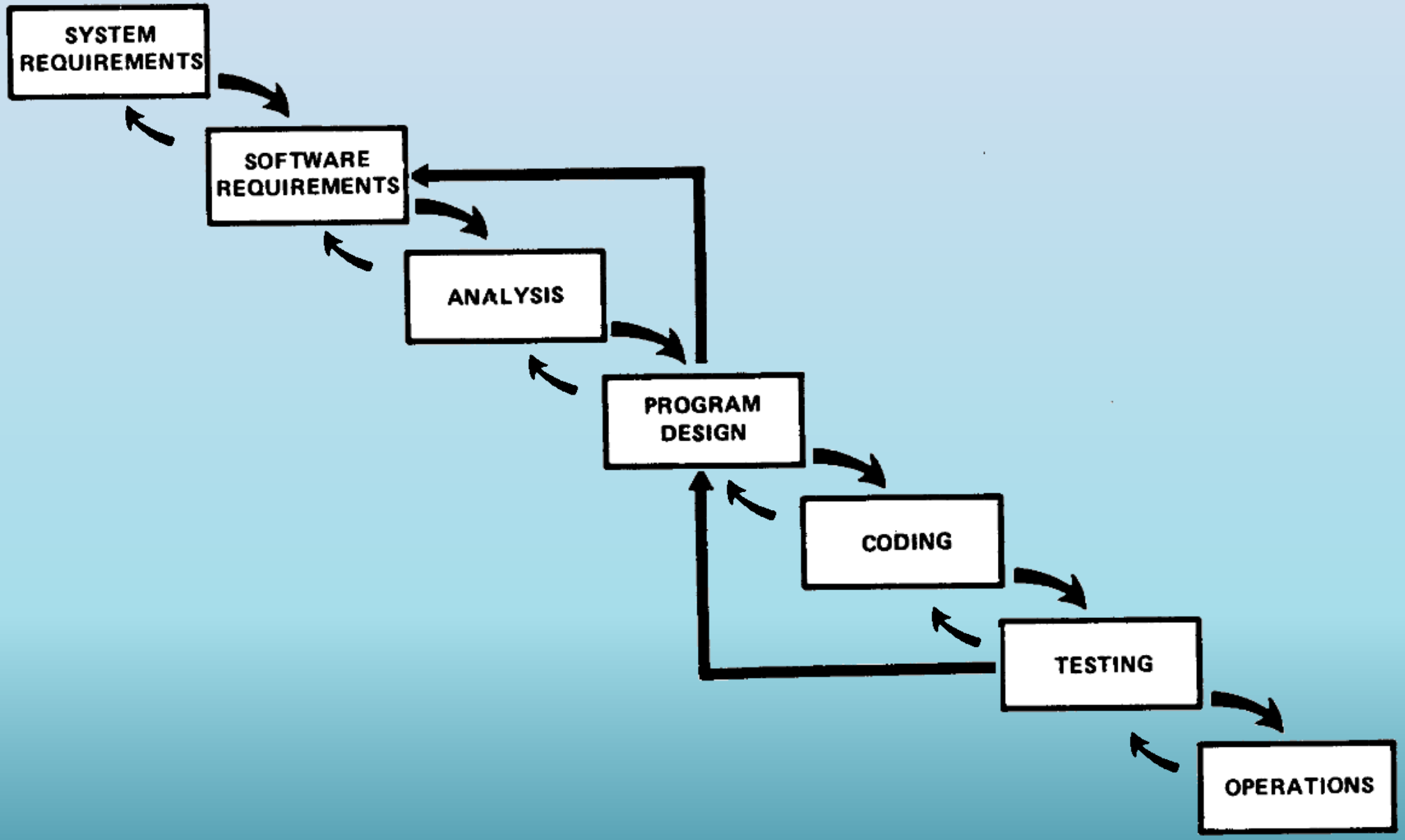
- Departamento de Defensa de los Estados Unidos (DoD)
  - MIL-STD-1521B (1976)
  - DOD-STD-2167 (1988)
- NASA
- Otros
  - JSP-188 (Británico),
  - German V-Model
  - GAM-T-17 (Francés)

Preguntas

- Qué es lo que dice el paper?
- Es Royce el creador del modelo en cascada?
- Royce describe el modelo en cascada en su paper?
- Royce defiende el modelo en cascada como la forma correcta de desarrollar grandes sistemas de software?

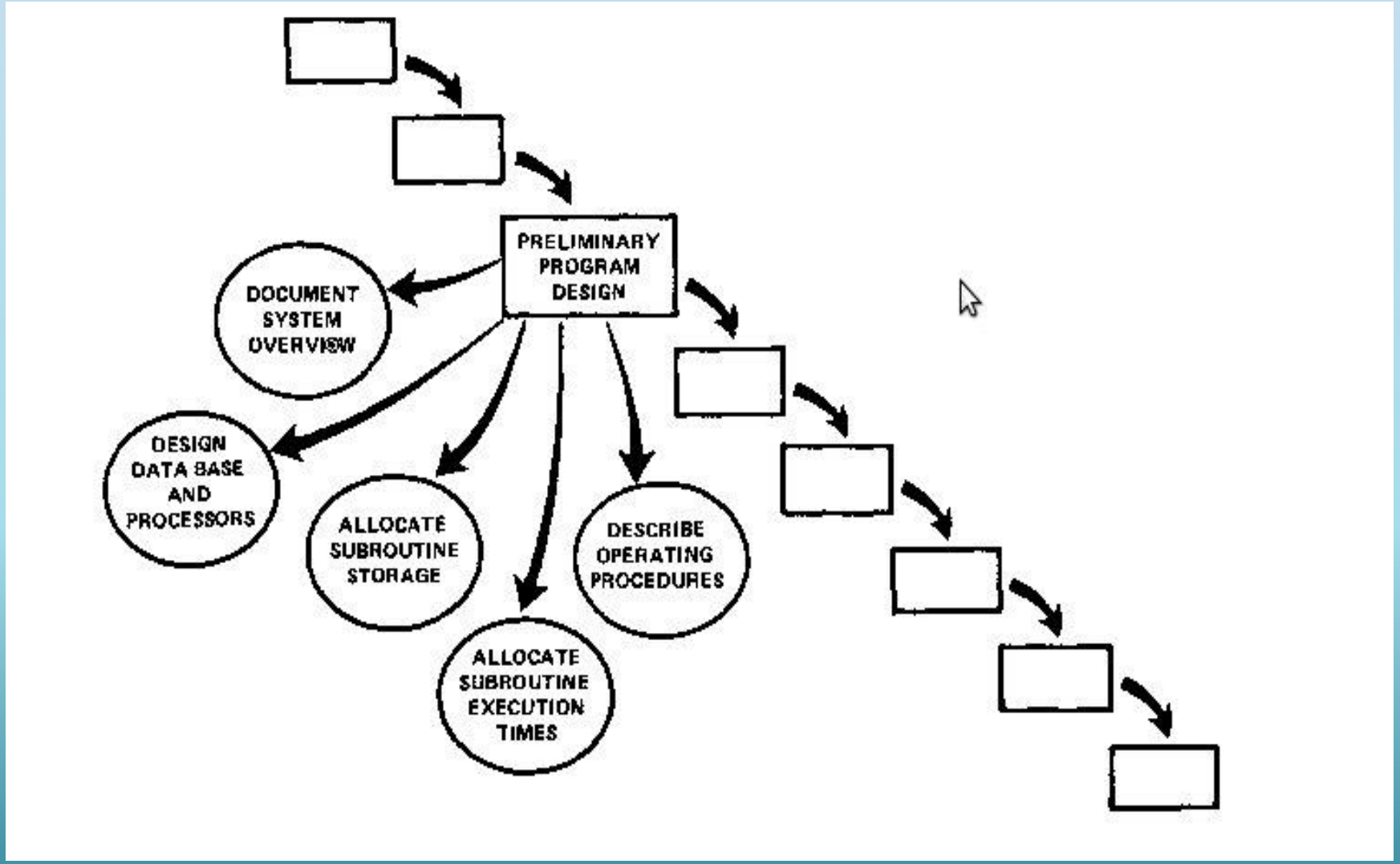
Análisis de la prueba
Introducción

- “I am going to describe my personal views about managing large software development”
- “I have become prejudiced by my experiences and I am going to relate some of these prejudices in this presentation.”


SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS

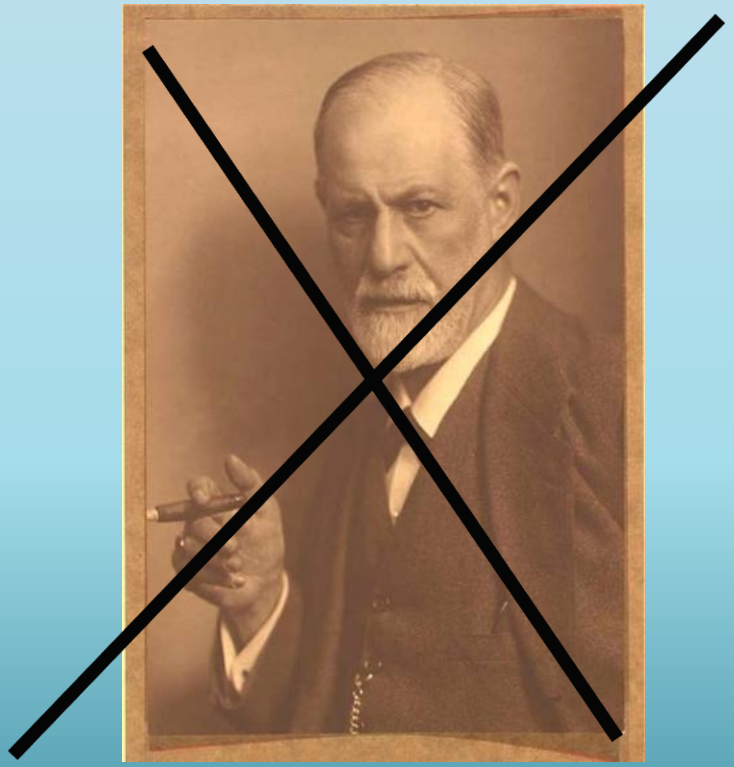
Step 1. El diseño viene primero

1. El diseño viene primero



1. El diseño viene primero

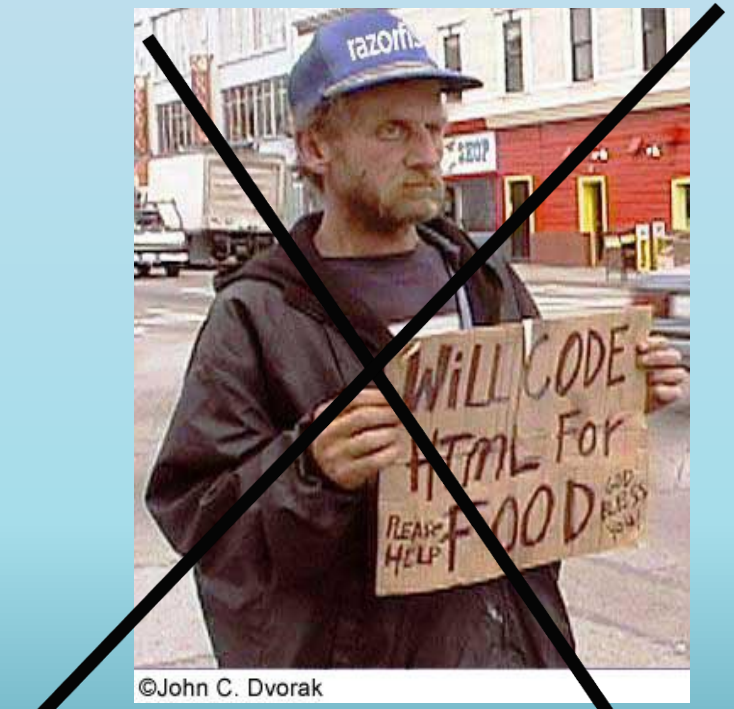
¿Quién debe hacer diseño?



ANALISTA

1. El diseño viene primero

¿Quién debe hacer diseño?



PROGRAMADOR

1. El diseño viene primero

Comenzar el proceso de diseño con diseñadores,
no analistas ni programadores.

DISEÑADOR

1. El diseño viene primero

- Comenzar el proceso de diseño con diseñadores, no analistas ni programadores.
- Diseñar, definir y alocar los módulos de procesamiento de datos aún a riesgo de equivocarnos.
- Escribir un documento entendible, informativo y actualizado

Step 2 . Documentar el Diseño

¿Cuánta documentación?

Mucha!!!!!!

Step 2 . Documentar el Diseño

¿Cuánta documentación?

Mucha!!!!!!

Para que???

- Evitar el síndrome “90% finalizado”

- Evitar el síndrome “90% finalizado”
- La documentación “es” la especificación y “es” el diseño.
- El verdadero valor de la documentación se verá a la hora del *testing*, en la *fase operacional* y en el *rediseño*.
- ***Testing:*** evitar que el que testea sea el mismo que cometió los errores.
- ***Operacional:*** poder hace el “testeo” mejor y más barato

Operacional: poder hace el “testeo” mejor y más barato

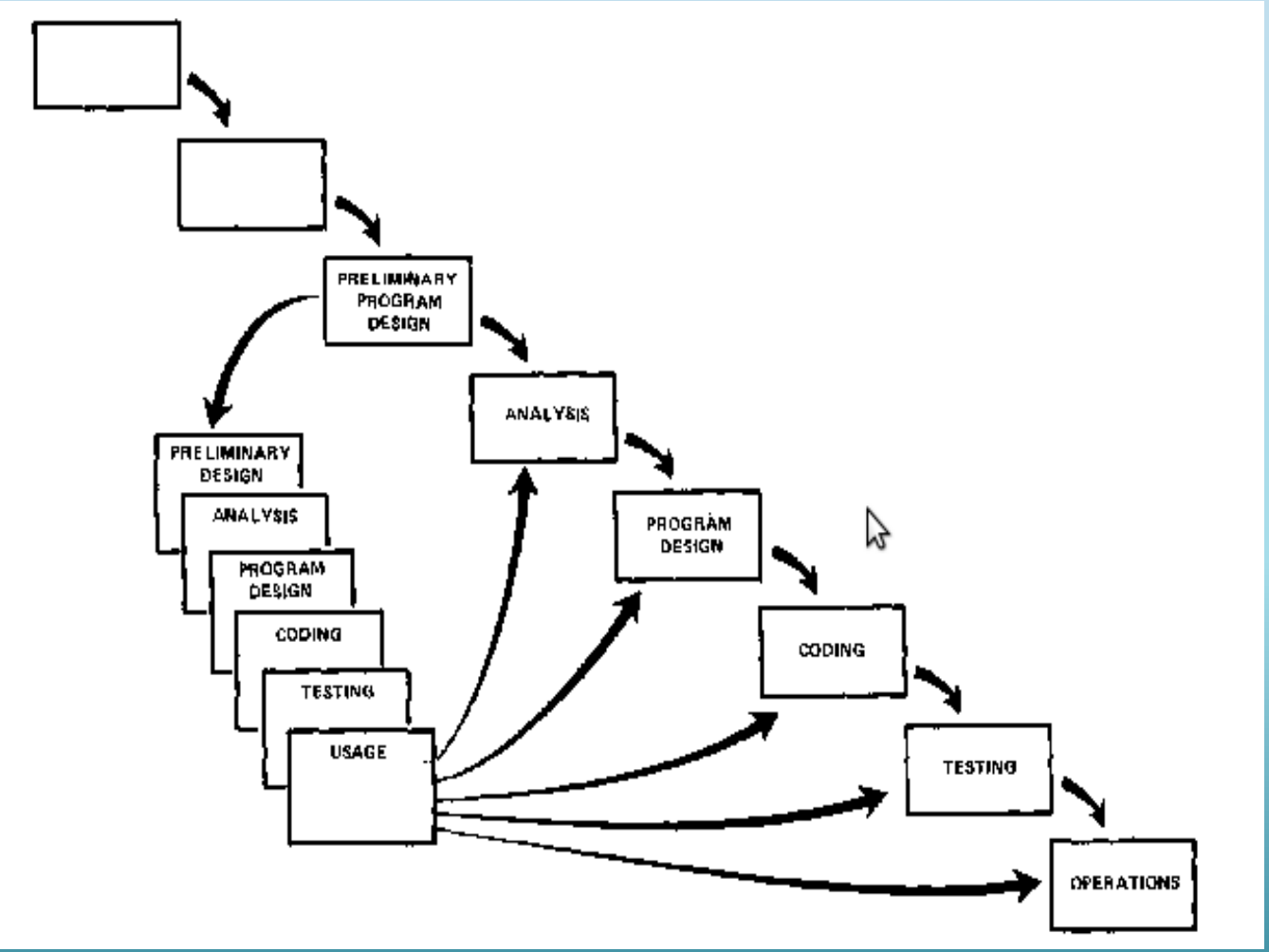
- Evitar el síndrome “90% finalizado”
- La documentación “es” la especificación y “es” el diseño.
- El verdadero valor de la documentación se verá a la hora del *testing*, en la *fase operacional* y en el *rediseño*.

Testing: evitar que el que testea sea el mismo que cometió los errores.

Operacional: poder hace el “testeo” mejor y más barato

Rediseño: facilitar los cambios

Step 3. Hacerlo dos veces



3. Hacerlo dos veces

- Para testear hipótesis
- Para probar distintas alternativas
- Para estimar

Step 4. Plan control and monitor Testing

Test:

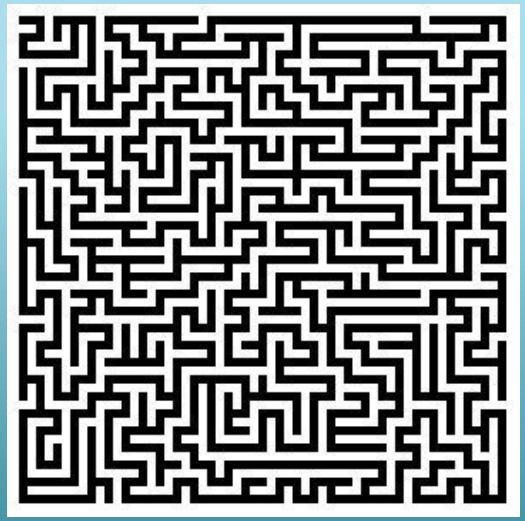
- Es la fase de mayor riesgo económico.
- Ocurre al final del proceso.

Consideraciones:

- Usar *“testers”.*
- Hacer un *scan* visual.
- Testear todo camino lógico al menos una vez.
- Detecto errores “simples” => pasar al área de testing.

¿Quien lo hace?

¿Cuando?

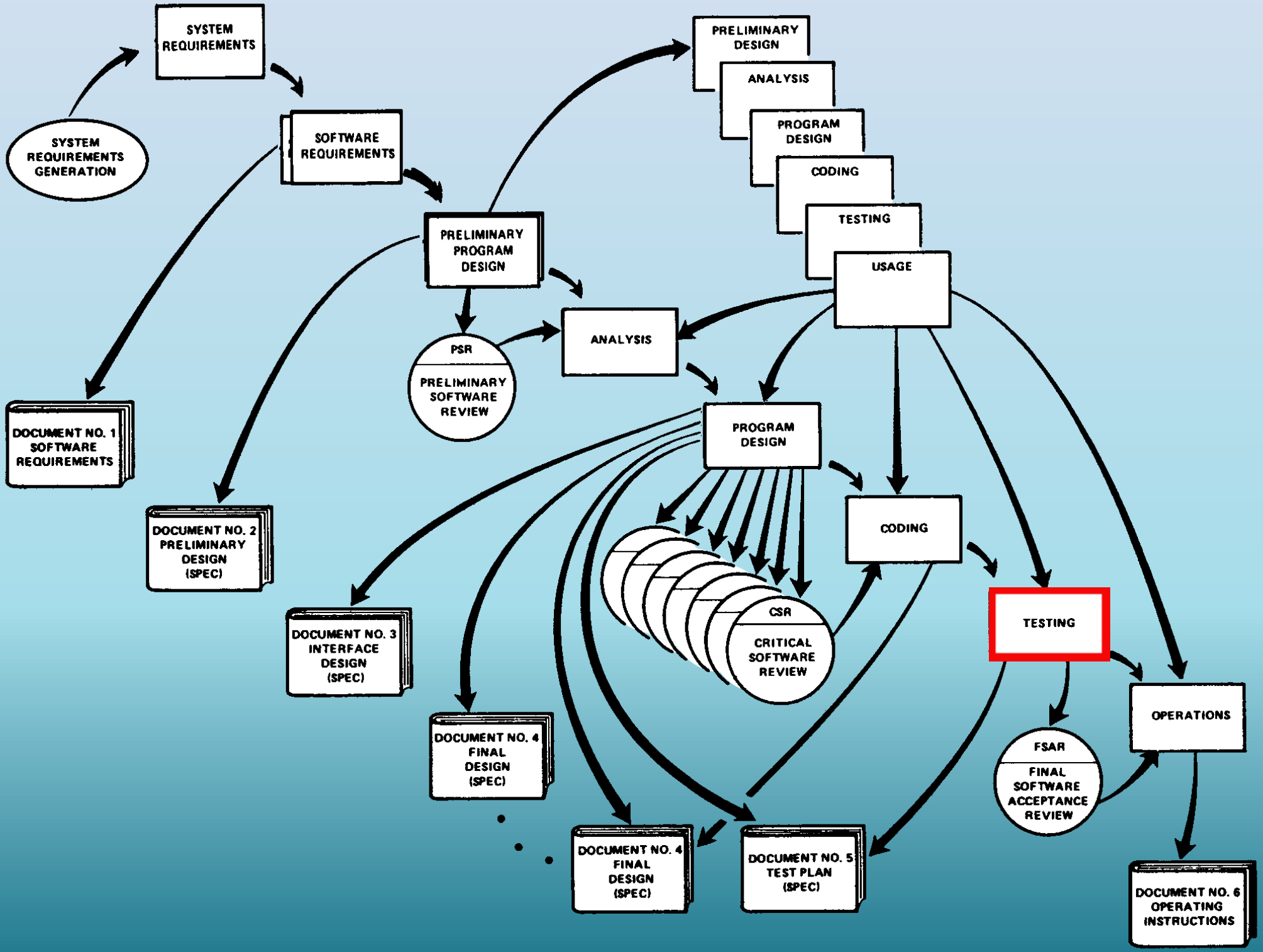


Step 5. involve the customer

Involve ≠> involucrar

Involve -> implicar

Resumiendo...


SYSTEM REQUIREMENTS
SYSTEM REQUIREMENTS GENERATION
SOFTWARE REQUIREMENTS
PRELIMINARY DESIGN
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
USAGE
PRELIMINARY PROGRAM DESIGN
PSR
PRELIMINARY SOFTWARE REVIEW
ANALYSIS
DOCUMENT NO. 1 SOFTWARE REQUIREMENTS
DOCUMENT NO. 2 PRELIMINARY DESIGN (SPEC)
PROGRAM DESIGN
CODING
DOCUMENT NO. 3 INTERFACE DESIGN (SPEC)
CSR
CRITICAL SOFTWARE REVIEW
TESTING
OPERATIONS
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
FSAR
FINAL SOFTWARE ACCEPTANCE REVIEW
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
DOCUMENT NO. 5 TEST PLAN (SPEC)
DOCUMENT NO. 6 OPERATING INSTRUCTIONS

Conclusiones

Si bien Royce no definió el modelo en cascada, el modelo que propone tiene gran parte de los problemas como ser:

Muchos...

Pero...

Ya pasamos por esto...

Ahora sí... conclusiones

- El paper no trata de introducir un modelo teórico de desarrollo, sino que intenta ser una recopilación de la experiencia de Winston.
- Alguno de los problemas del modelo *Waterfall* Royce ya los había visto y tratado de combatir.
- Es criticable la visión de que hay que hacer que el cliente se comprometa con una serie de requerimientos para que luego no de marcha atrás...

Preguntas

- Qué es lo que dice el paper?
  - El paper presenta la situación en que se encontraba el desarrollo de software en ese momento (1970) y propone mejoras
- Es Royce el creador del modelo en cascada?
  - No, Royce solo lo utiliza para mostrar como se desarrollaba software en ese momento.
- Royce describe el modelo en cascada en su paper?
  - No, solo muestra un dibujito
- Royce defiende el modelo en cascada como la forma correcta de desarrollar grandes sistemas de software?
  - No, al contrario, lo critica y propone mejoras

Referencias

- ***“Managing the development of Large Software Systems"***
  http://portal.acm.org/citation.cfm?id=41801
- ***Agile and Iterative Development: A Manager's Guide***
  By Craig Larman August 11, 2003 ISBN : 0-13-111155-8
  on page 102ff (The Historical Accident of Waterfall Validity?)

¿ Preguntas ?

¿ No ?

Listo !