UNIVERSDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

CURSO:
Métodos Formales de Pruebas - Testing de Software

Tema:
Métodos formales en las Pruebas de Software
Semana 4

Docente: Mg. Wilder Inga

Los métodos formales

- Los métodos formales permiten al ingeniero de software especificar, desarrollar y verificar un sistema informático mediante la aplicación de una notación matemática rigurosa.
- Utilizando un lenguaje de especificación formal, un método formal proporciona los medios de especificar un sistema de forma que se aseguren, de forma sistemática, la consistencia, la completitud y la corrección.
- Se suelen basar en notaciones matemáticas similares a las del álgebra de conjuntos y la lógica

Propósitos de los métodos formales

- Se introducir rigor en todas las fases de desarrollo de software, con lo que es posible evitar que se pasen por alto cuestiones críticas;
- Proporcionar un método estándar de trabajo a lo largo del proyecto; y
- Constituir una base de coherencia entre las muchas actividades relacionadas y, al contar con mecanismos de descripción precisos y no ambiguos, proporcionar el conocimiento necesario para realizarlas con éxito.

Los Métodos Formales En La Ingeniería Del Software

Conceptos

El concepto de los métodos formales involucra una serie de técnicas lógicas y matemáticas con las que es posible especificar, diseñar, implementar y verificar los sistemas de información.

La importancia de los métodos formales en la Ingeniería del Software se incrementó en el siglo XXI: se desarrollaron nuevos lenguajes y herramientas para especificar y modelar formalmente, y se diseñaron metodologías maduras para verificar y validar

Desde hace varias décadas se utiliza técnicas de notación formal para modelar los requisitos, principalmente porque estas notaciones se pueden verificar fácilmente y porque, de cierta forma, son más comprensibles para el usuario final

Propósitos de los métodos formales

1. Sistematizar e introducir rigor en todas las fases de desarrollo de software, con lo que es posible evitar que se pasen por alto cuestiones críticas;
2. Proporcionar un método estándar de trabajo a lo largo del proyecto; y
3. Constituir una base de coherencia entre las muchas actividades relacionadas y, al contar con mecanismos de descripción precisos y no ambiguos, proporcionar el conocimiento necesario para realizarlas con éxito.

Uso de los métodos formales

1. Las políticas de los requisitos. En un sistema seguro se convierten en las principales propiedades de seguridad que éste debe conservar, es decir, el modelo de políticas de seguridad formal, como confidencialidad o integridad de datos.

2. La especificación. Es una descripción matemática basada en el comportamiento del sistema, que utiliza tablas de estado o lógica matemática. No describe normalmente al software de bajo nivel, pero sí su respuesta a eventos y entradas, de tal forma que es posible establecer sus propiedades críticas.

3. Las pruebas de correspondencia entre la especificación y los requisitos. Es necesario demostrar que el sistema, tal como se describe en la especificación, establece y conserva las propiedades de las políticas de los requisitos. Si están en notación formal se puede diseñar pruebas rigurosas manuales o automáticas.

4. Las pruebas de correspondencia entre el código fuente y la especificación. Aunque muchas técnicas formales se crearon inicialmente para probar la correctitud del código, pocas veces se logra debido al tiempo y costo implicados, pero pueden aplicarse a los componentes críticos del sistema.

5. Pruebas de correspondencia entre el código máquina y el código fuente. Este tipo de pruebas raramente se aplica debido al costo y a la alta confiabilidad de los compiladores modernos.

Principales Métodos Formales utilizados en el desarrollo de software

- Métodos formales basados en Lógica de Primer Orden: Z, B, VDM, Object-Z, Z++ y VDM++.
- Métodos formales basados en Formalismos Algebraicos: HOSA (Hidden Order Sorted Algebras), TROLL, OBLOG, Maude y AS-IS (Algebraic Specifications with Implicit States).
- Métodos formales basados en Redes de Petri: CO-OPN (Concurrent Object-Oriented Petri Nets)
- Métodos formales basados en Lógica Temporal: TRIO, OO-LTL y ATOM.
- Métodos Semiformales: Syntropy, Statemate, UML y OCL (Object Constraint Language)

Definition 1. *A finite automaton M is defined by a 5-tuple $(\Sigma, Q, q_0, F, \delta)$, where*

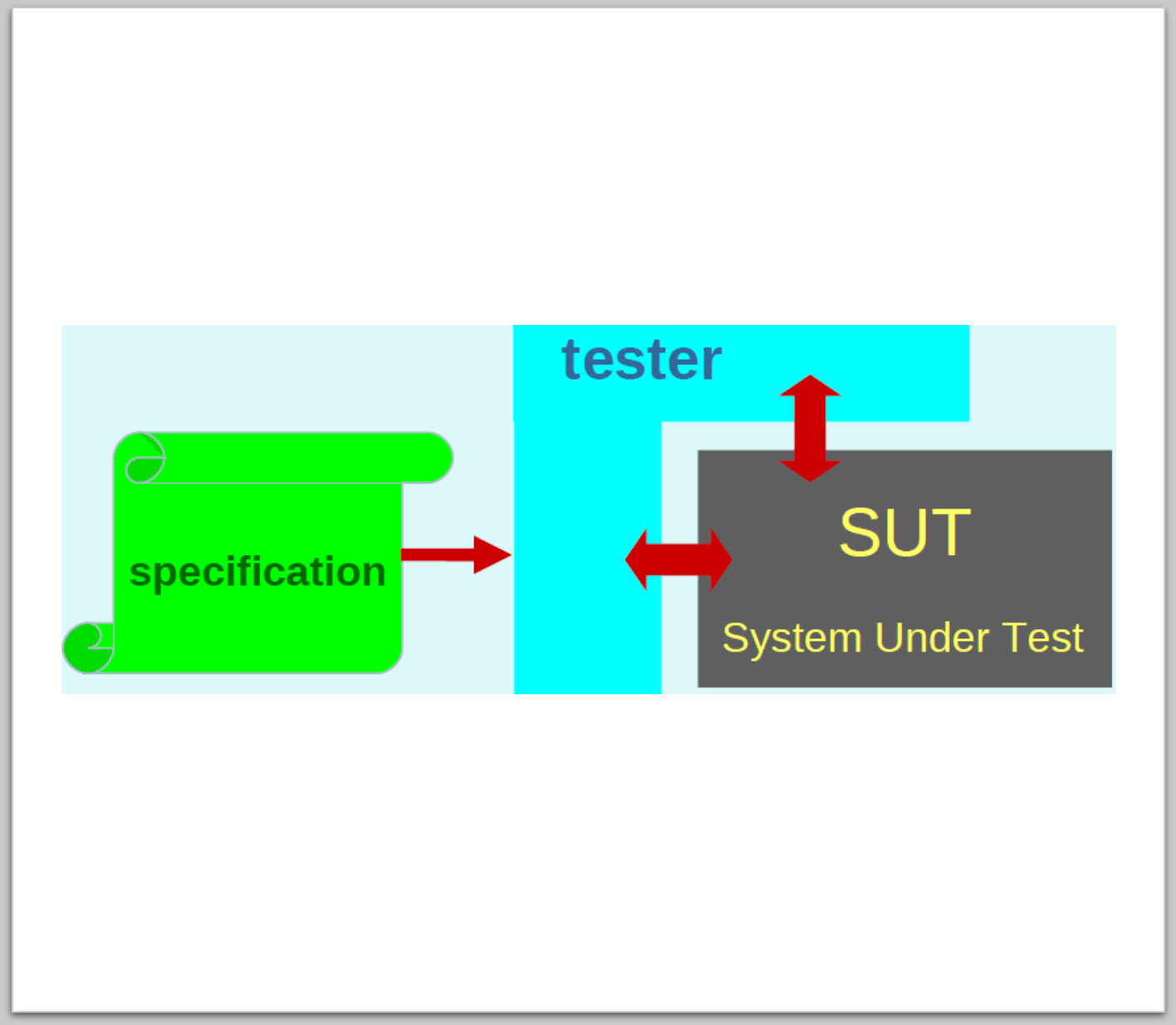
- Σ *is the set of symbols representing input to* M
- Q *is the set of states of* M
- $q_0 \in Q$ *is the start state of* M
- $F \subseteq Q$ *is the set of final states of* M
- $\delta : Q \times \Sigma \to Q$ *is the transition function*

Especificación en lógica formal de las funciones de insertar y borrar elementos en una estructura pila

▪ **L.push**(e:stelement): Inserta *e* después del último elemento insertado
▪ **PRE**: $\exists L \wedge L \neq \{NULL\} \wedge L = \{PRF\} \wedge$ tamaño(L) = n $\wedge$ ¬existe(L,clave(e))
▪ **POST**: L = {PRF,e} $\wedge$ tamaño(L) = n + 1
▪ **PRE**: $\exists L \wedge L = \{NULL\}$
▪ **POST**: L = {e} $\wedge$ n = 1
▪ **L.pop**(): borra el elemento recientemente insertado
▪ **PRE**: $\exists L \wedge L = \{PRF, e\} \wedge$ tamaño(L) = n
▪ **POST**: L = {PRF} $\wedge$ tamaño(L) = n – 1

Testing

Comprobar o medir algunas características de calidad de un objeto en ejecución mediante la realización de experimentos de forma controlada



¿Por qué Model-Based Testing?

Software bugs / errors cost US economy yearly:
$ 59.500.000.000 (www.nist.gov)
$ 22 billion could be eliminated…

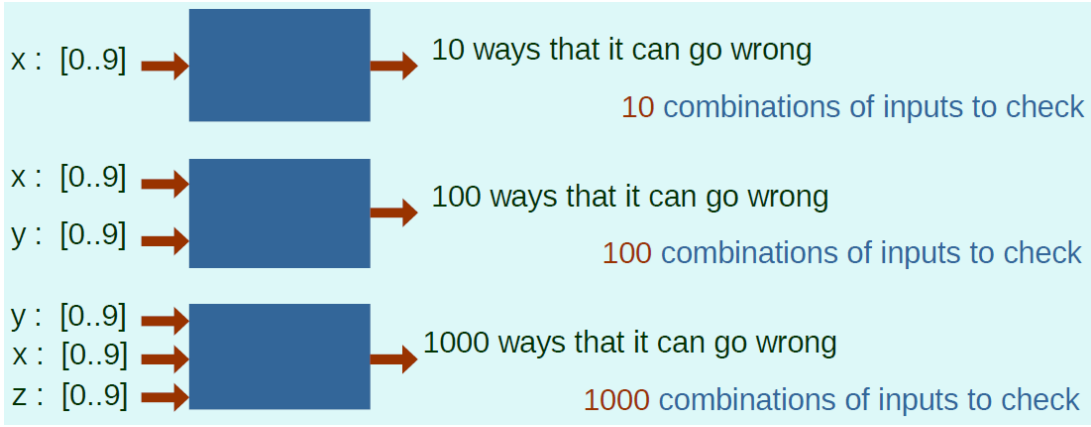
- Aumento de la complejidad y búsqu
  - El esfuerzo de prueba crece exponencialmente con el tamaño del sistema
  - Las pruebas no pueden seguir el ritmo del desarrollo

- Más abstracción
  - Menos detalle
  - Desarrollo basado en modelos; UML, MDA, Simulink/Matlab de OMG

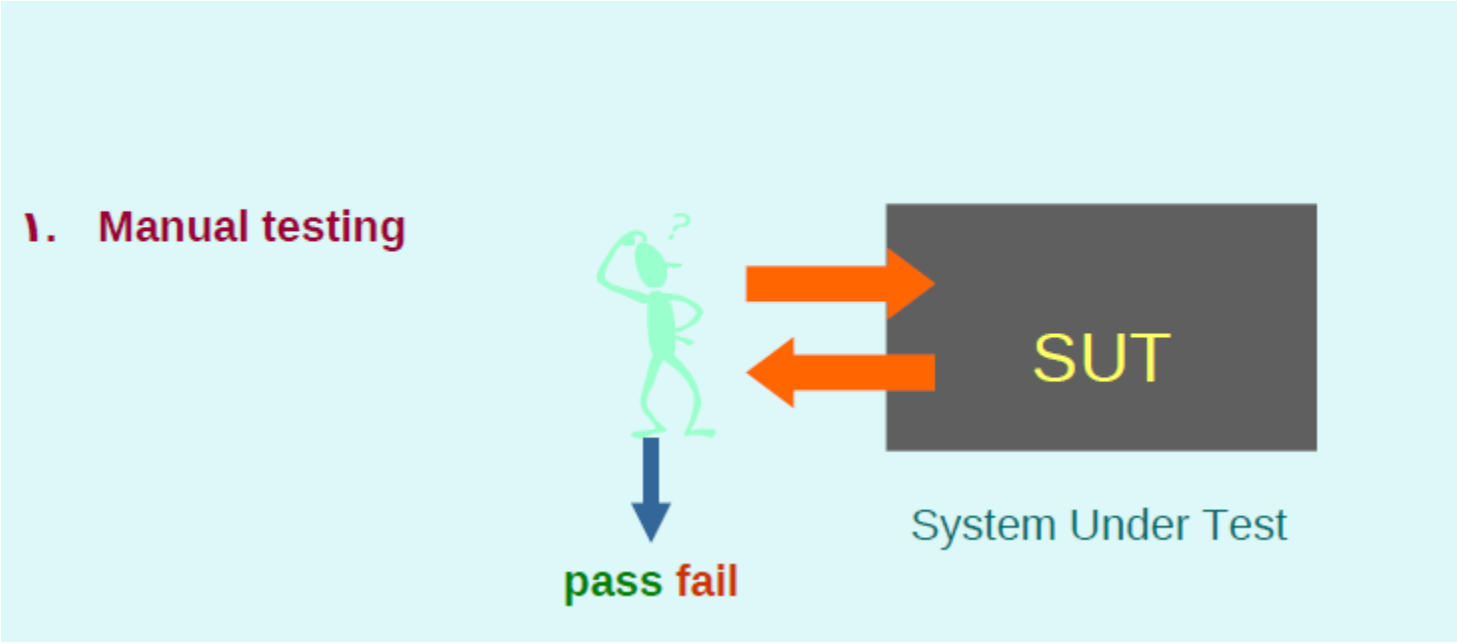
- Comprobando la calidad
  - En la práctica: pruebas - ad hoc, demasiado tarde, caras, mucho tiempo
  - Investigación: verificación formal - pruebas, verificación de modelos, . . .con un impacto práctico decepcionante

Testing Complexity

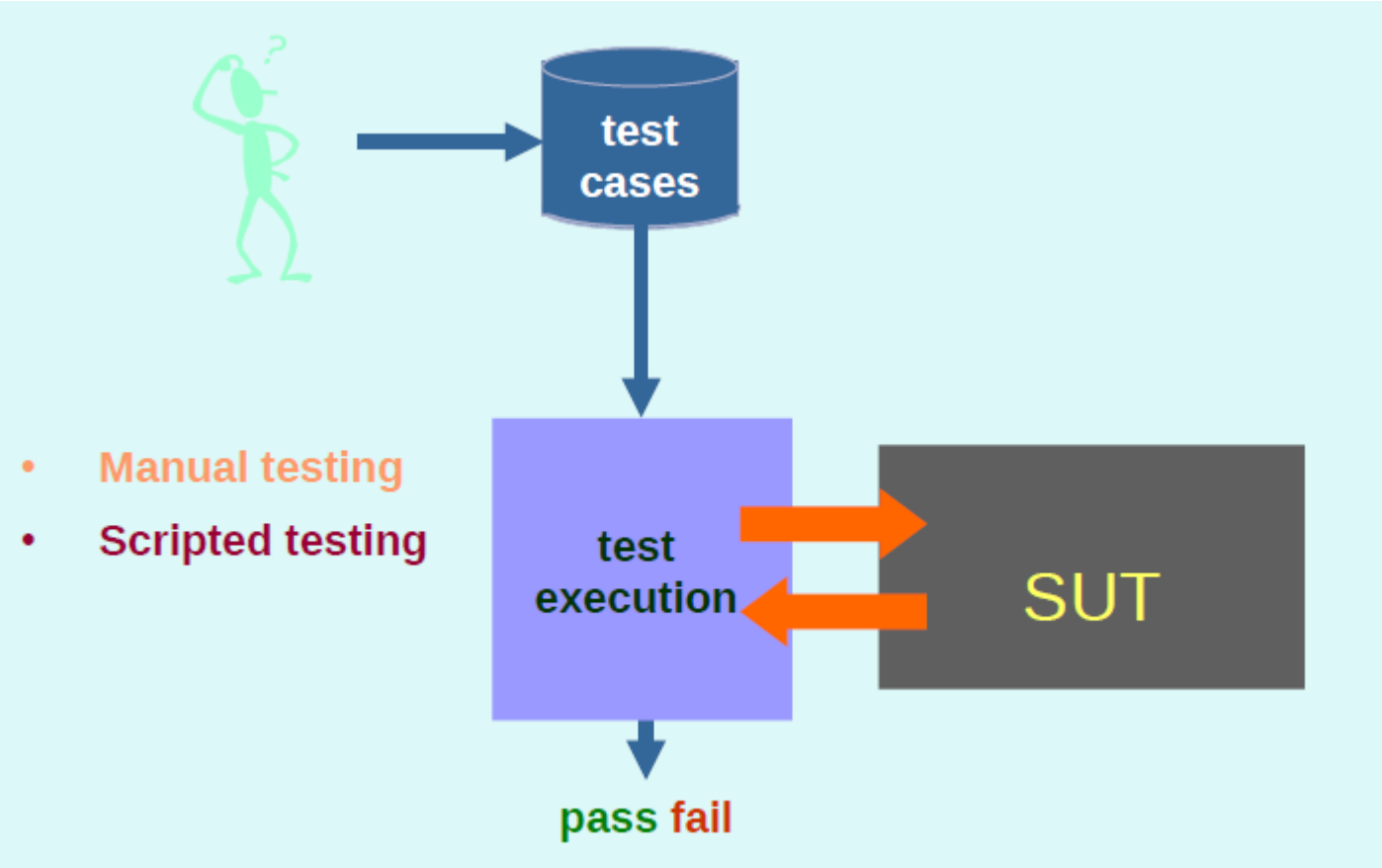
- El aumento de la complejidad y el tamaño del esfuerzo de prueba de sistemas crece exponencialmente con el tamaño del sistema. Las pruebas no pueden seguir el ritmo del desarrollo.
- Es necesaria la automatización de las pruebas.Las pruebas basadas en modelos son una de las técnicas



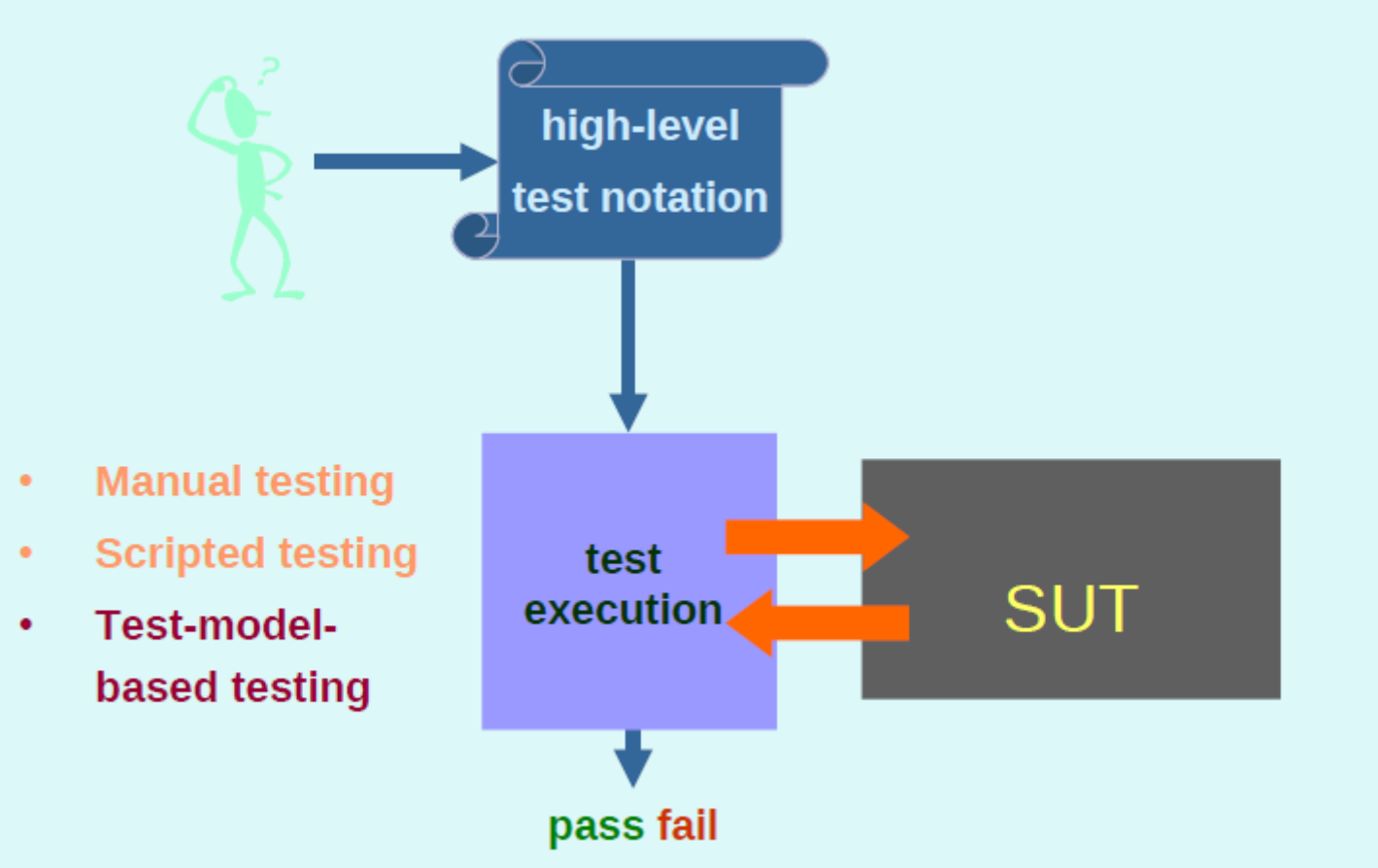
Developments in Testing 1



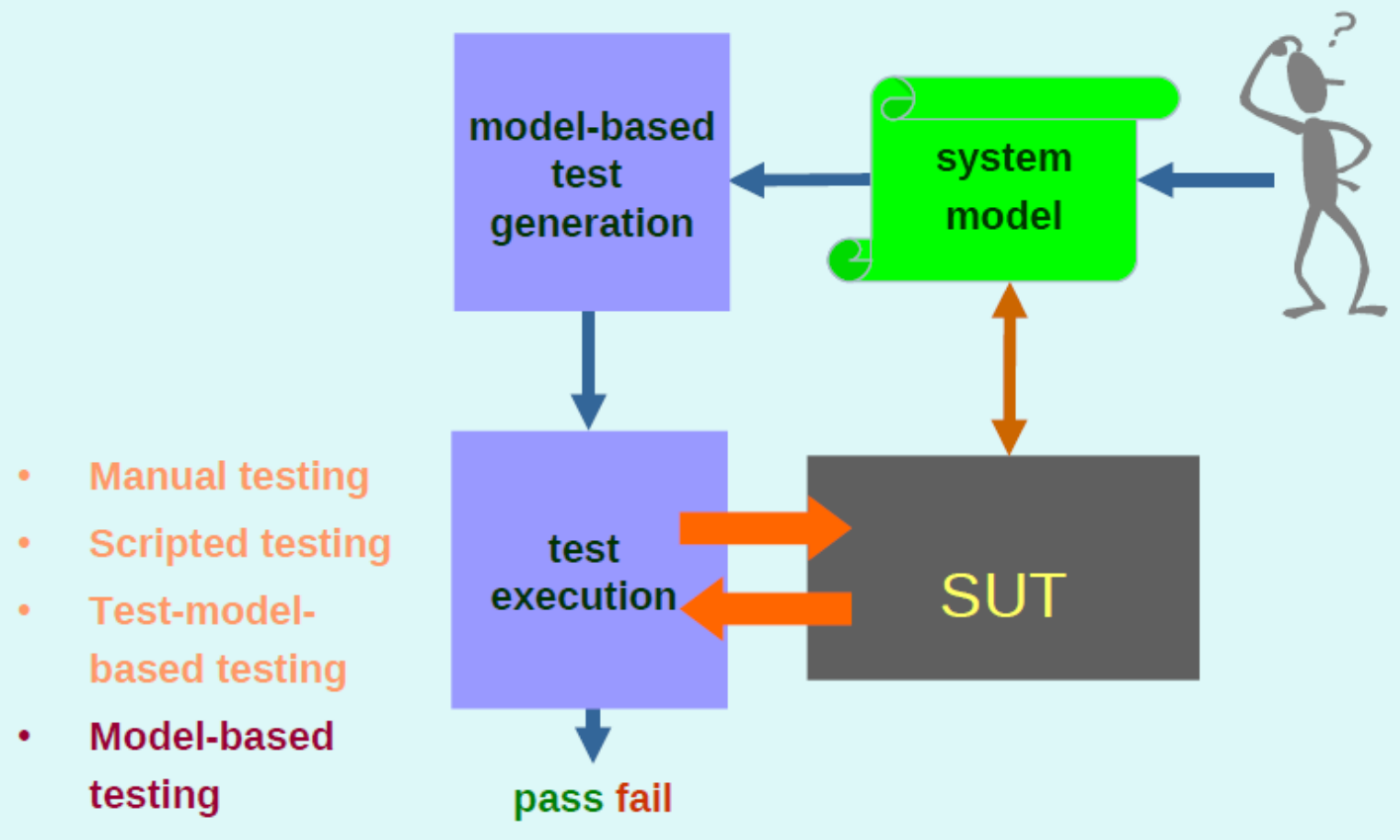
Developments in Testing 2



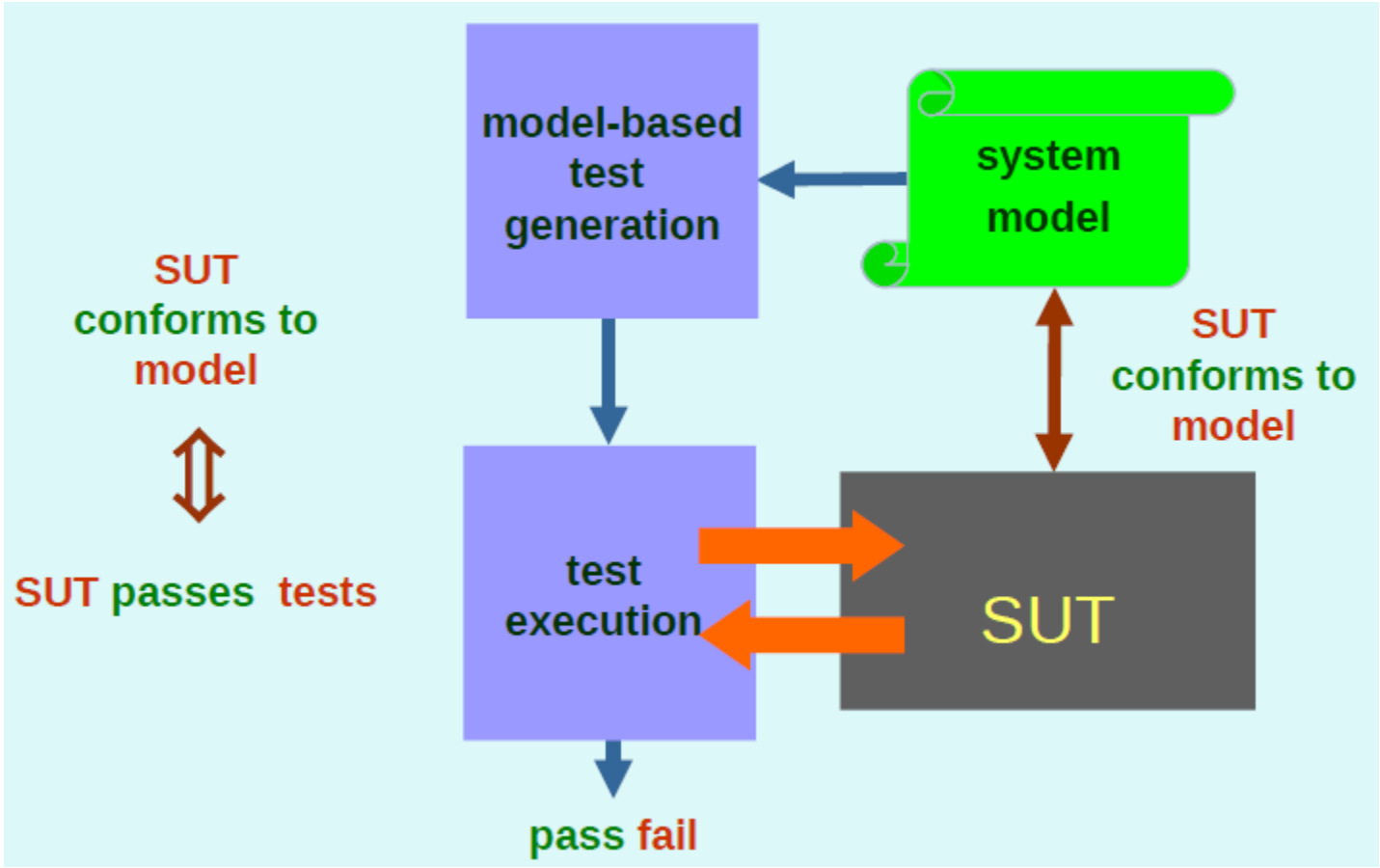
Developments in Testing 3



Developments in Testing 4



Model-Based Testing



Model-Based Testing

- Pruebas basadas en modelos -> Prueba de SUT con respecto a un modelo / especificación (formal)-> Modelo de estado/transición, condiciones previas/posteriores, CSP, Promela, UML, n.° de especificación, ...
- Promesa: mejores pruebas, más rápidas y más baratas
  - Generación algorítmica de pruebas y oráculos de prueba: herramientas
  - Base formal e inequívoca para las pruebas
  - Medición de la integridad de las pruebas
  - Mantenimiento de pruebas a través de la modificación del modelo
  - Potencial para combinar la práctica y la teoría

Utilidad de modelos formales

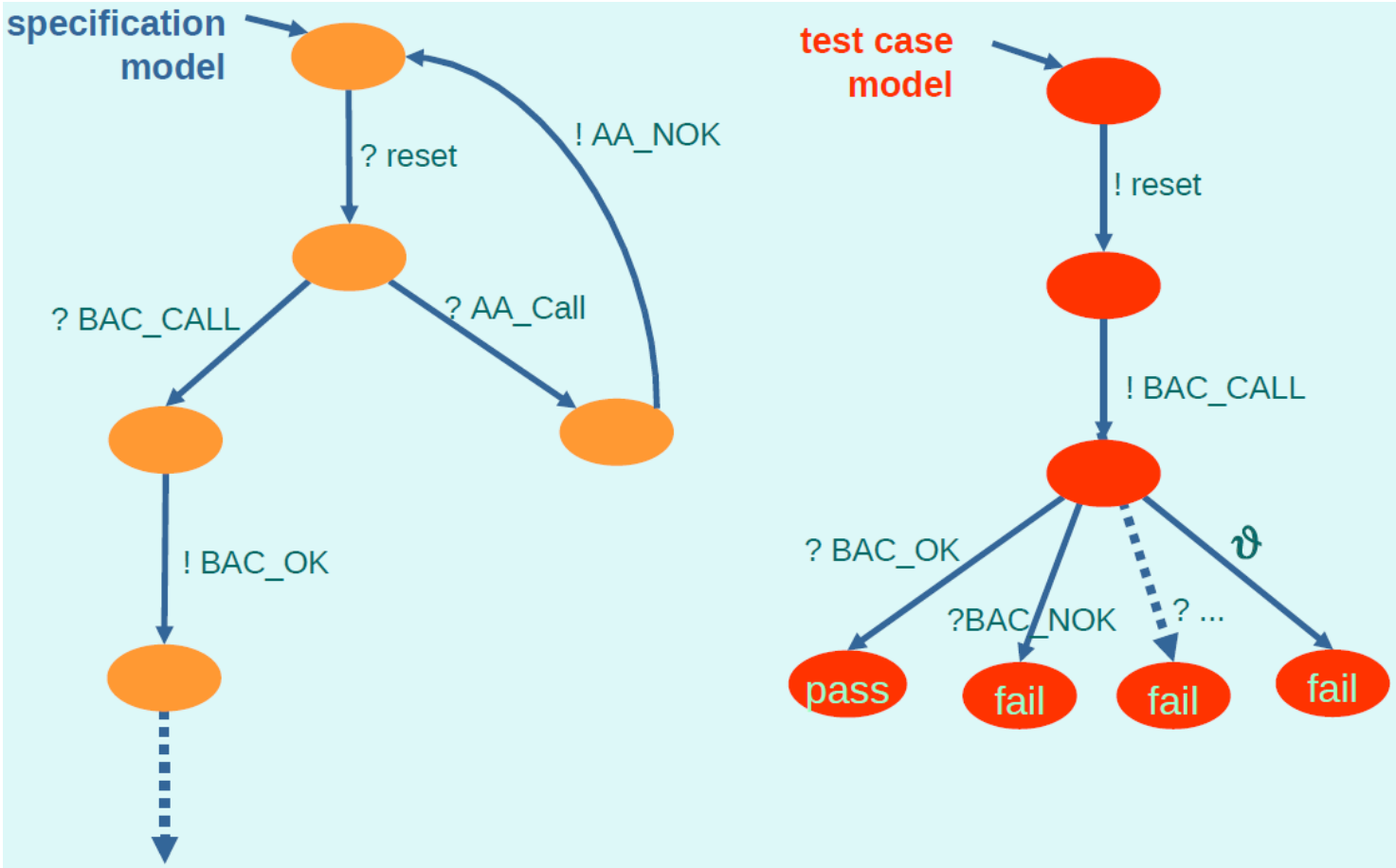
- Elaborar un modelo revela errores
- Simulación, paso a paso a través del modelo
- La comprobación del modelo pasa por todos los estados del modelo.
- Demostración de teoremas sobre el modelo
- Generación de código a partir del modelo.

Enfoques para Model-Based Testing

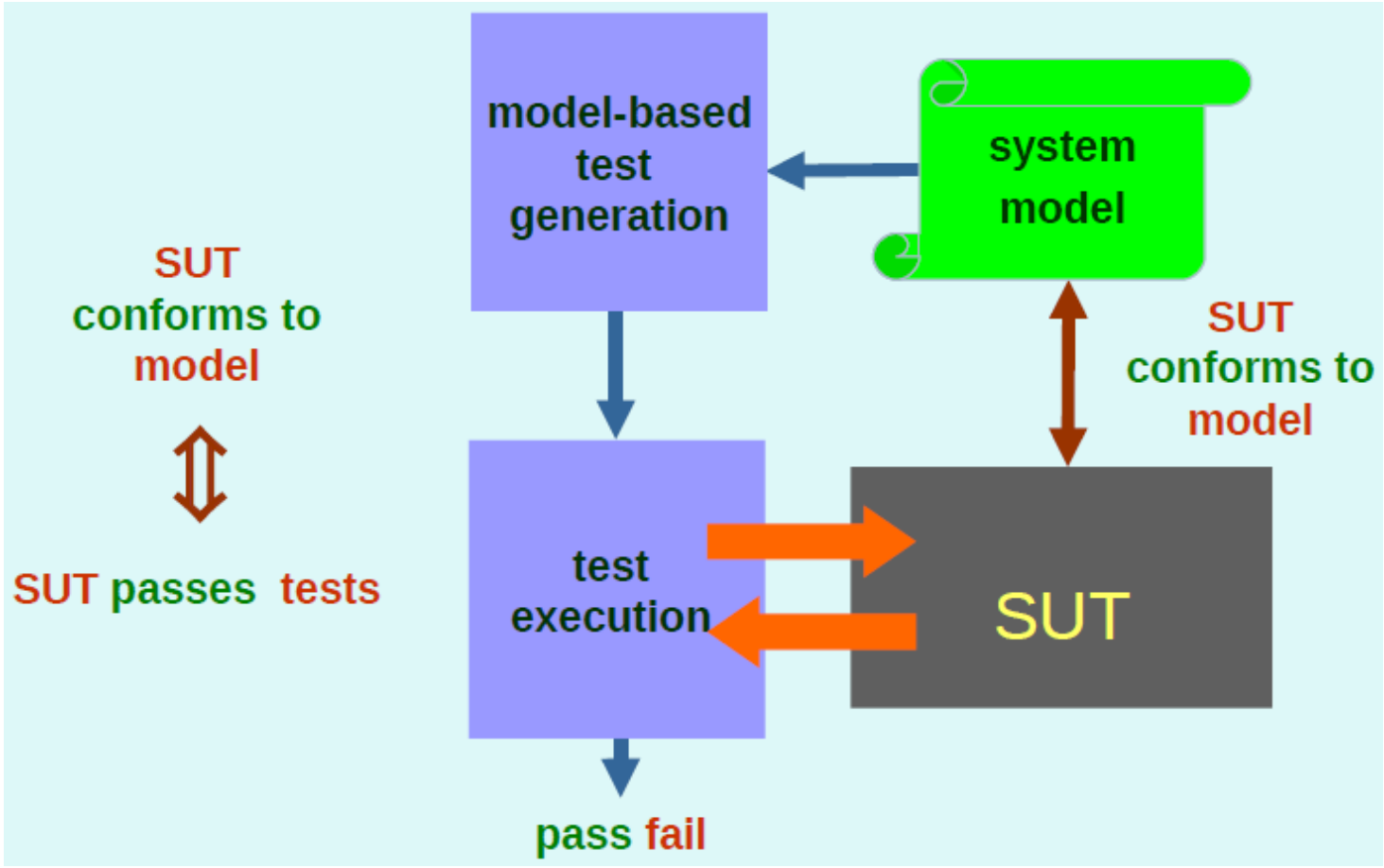
Varios paradigmas de modelado:

- Máquina de estados finitos
- Pre/Post Condiciones
- Sistemas de transición (LTS)
- Programas como Funciones
- Pruebas de tipos de datos abstractos
- Autómatas cronometrados
- . . . . . .

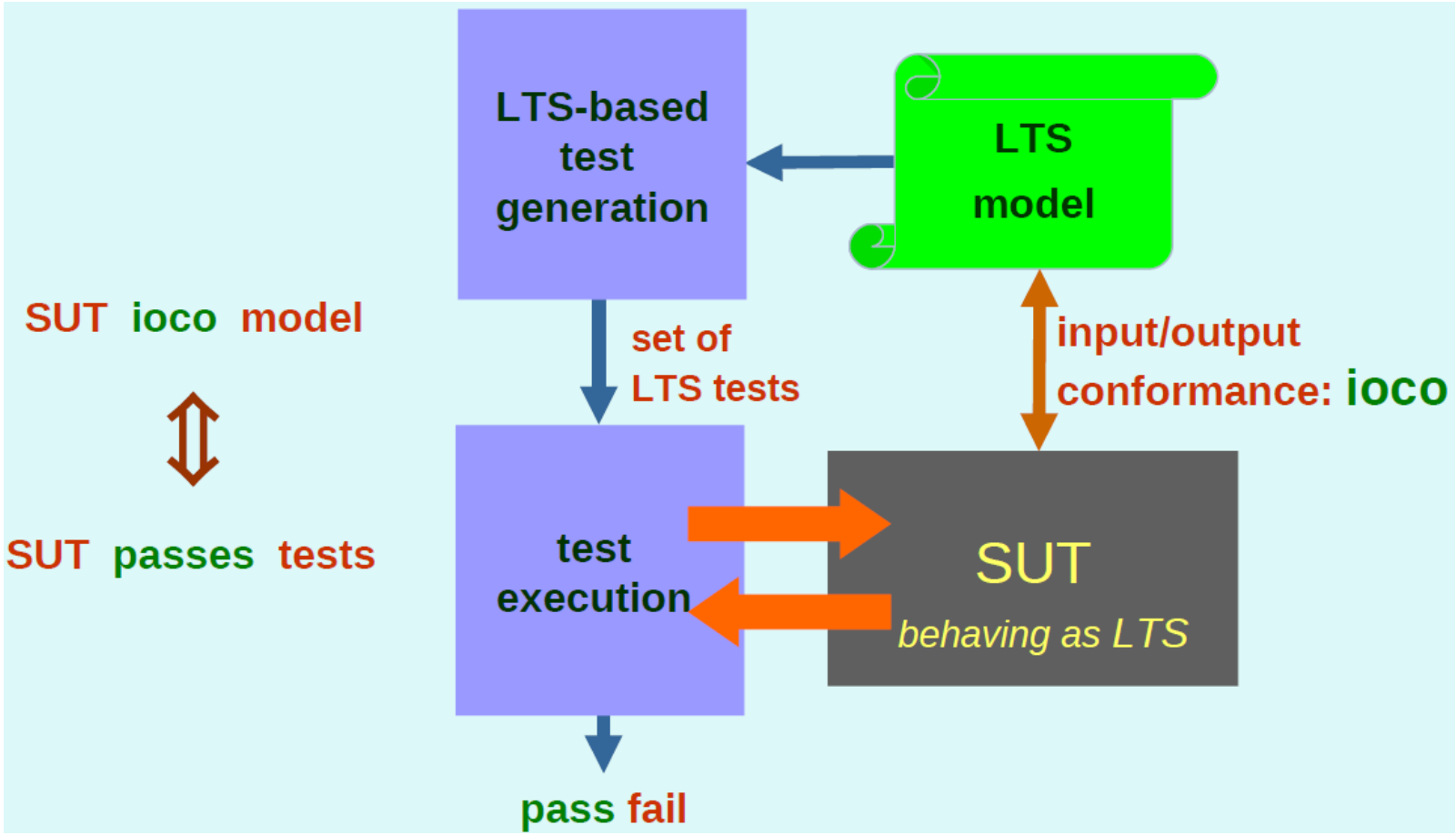
Model vs Model of Test Cases



Model-Based Testing



Model-Based Testing



Model-Based Testing Tools

Some Tools for Model-Based testing:

- AETG
- Agatha
- Agedis
- Autolink
- Conformiq
- Cooper
- Cover
- STG
- TestGen (Stirling)
- QuickCheck
- Reactis
- RT-Tester
- SaMsTaG
- Smartesting
- Spec Explorer
- Statemate
- TestGen (INT)
- TorXakis
- Test Composer
- TGV
- TorX
- Uppaal-Tron
- Tveda
- Gotcha
- Phact/The Kit

Actividad