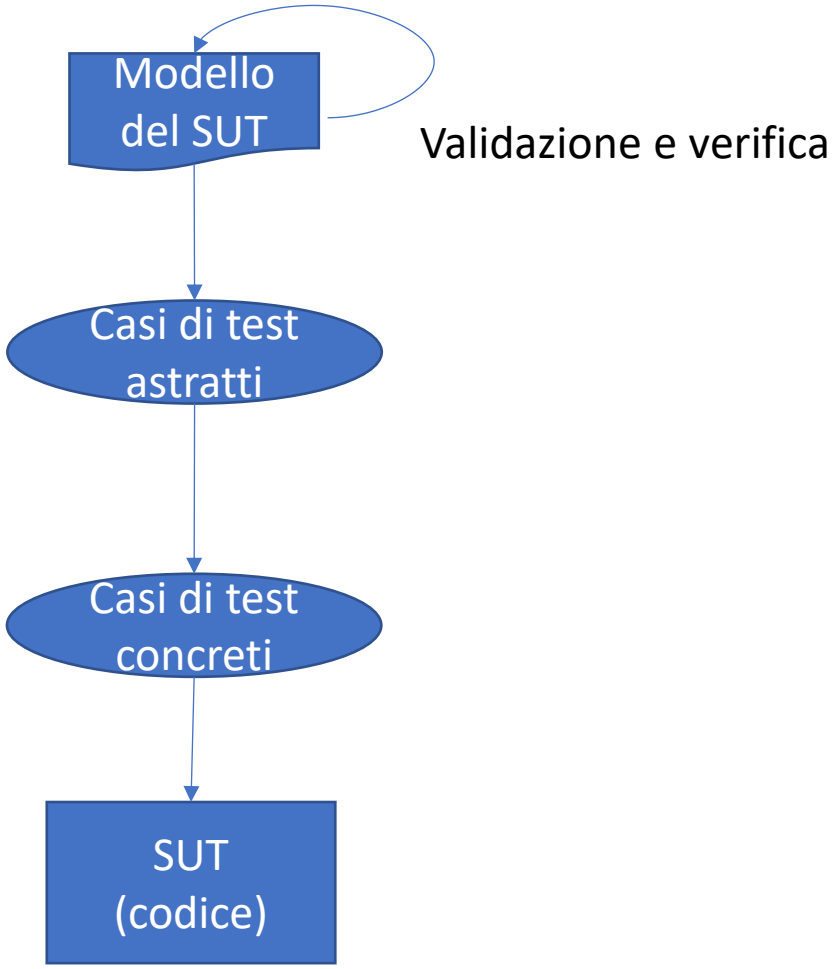
Model based testing approaches e tools

Tetsing e verifica del sw

AA 2026

Flusso del model-based testing



- Ogni metodo differisce per

- Tecniche per scrivere il modello
- Attività di verifica sul modello
- Generazione dei test astratti
- Esecuzione dei test concreti sulla SUT
- Come collegare SUT e modello

Caso di studio

Scriviamo la specifica Asmeta

- Ho una lampada che può essere in tre stati
- OFF – STANDYBY – ON
- Può essere anche spenta (poweroff) o accesa (poweron)
- ha solo bottone che può essere premuto dall'utente
  - Ha effetto solo se la lampada è accesa (STANDY o ON)
- Inizialmente la lampada è in OFF
- Se poweron ed è in OFF va in STANDBY
- Se premo il bottone ed è in STANDY passa a ON
- Se ripremo il bottone ed è ON passa in STANDBY
- Se stacco (poweroff) va sempre in OFF

Approci/tool che vedremo

1. Asmeta + ATGT con approcio OFFLINE
    1. Scriviamo la spec come ASM
    2. Scriviamo il codice in Java
    3. Generiamo i test dalla specifica e li traduciamo in Junit

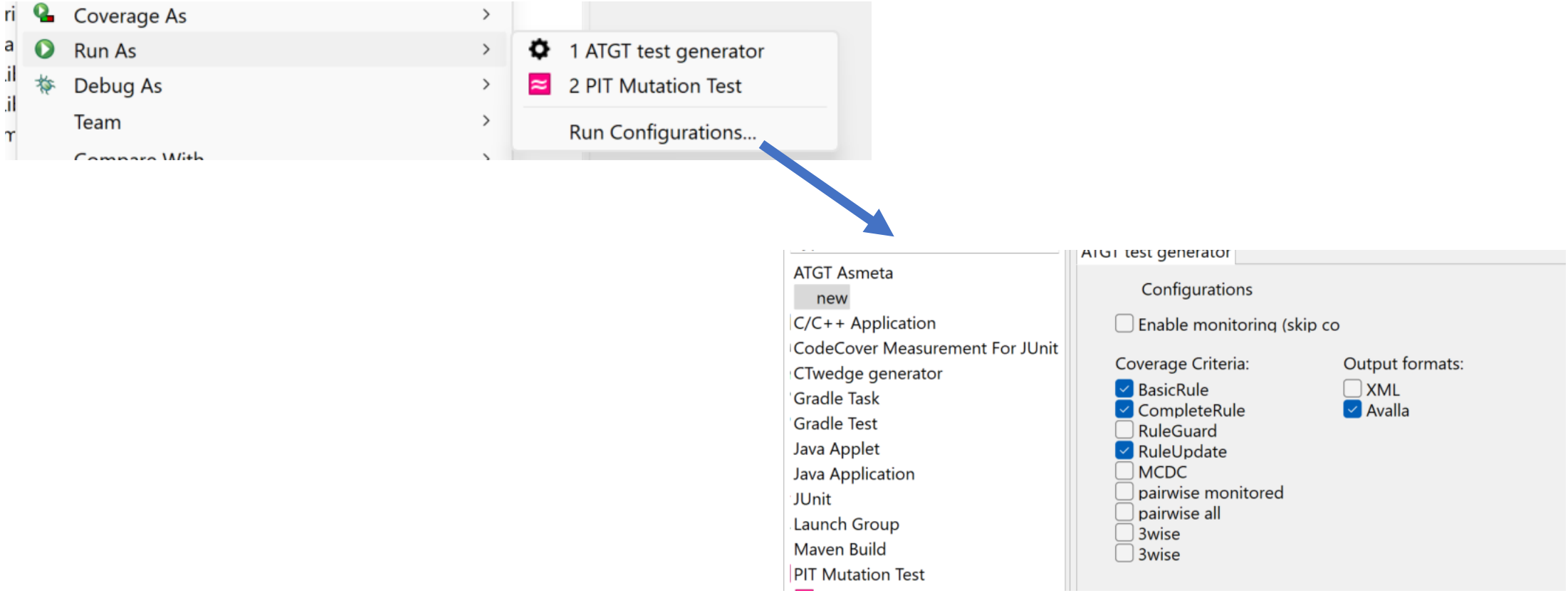
    Codice: codice_lezioni\6_atgt
2. GraphWalker

Asmeta e MBT ATGT

1. Possiamo generare i casi di test a partire da specifiche ASM
   - ATGT
2. Produrrà degli scenari avalla in modo automatico
3. Dovremo poi tradurre questi scenari in codice reale

Come eseguire ATGT

- Run ATGT direttamente sul file asmeta
- Produce i file avalla sotto abstract tests



Diversi criteri di copertura

- Basic Rule coverage
  - Ogni regola deve scattare almeno un volta
- Rule update coverage
  - Ogni update viene coperto e non è triviale (cioè cambia effettivamente il risultato)
- Nota: se un criteri non è copribile, il tool cerca di dimostrarlo come con il model advisor

Vantaggi/svantaggi

- Il modello in Asmeta l'abbiamo per la verifica/validazione
- Posso usare gli scenari per animare il modello

- Contro:
  - Il modello deve essere traducibile nel MC
  - Alcune volte ATGT non funziona correttamente

Random generator

- Si possono generare I casi di test anche random
- Si deve dire la lunghezza e in numero di test
- Random funziona con qualsiasi modello non solo quelli con cui funziona il model checker

Da avalla a Junit

- In genere
    - Ogni scenario si traduce in un test di Junit
    - Set di una monitorata corrisponde ad un set di un campo della classe
        - Oppure la chiamata da un metodo.
    - Check si traduce con un assert equals o true
    - Step si traduce in esecuzione di un metodo che esegue una operazione
        - Ad esempio in arduino c’è il loop
- Demo

2. Graph walker

- Modello: macchina a stati con label su transizioni

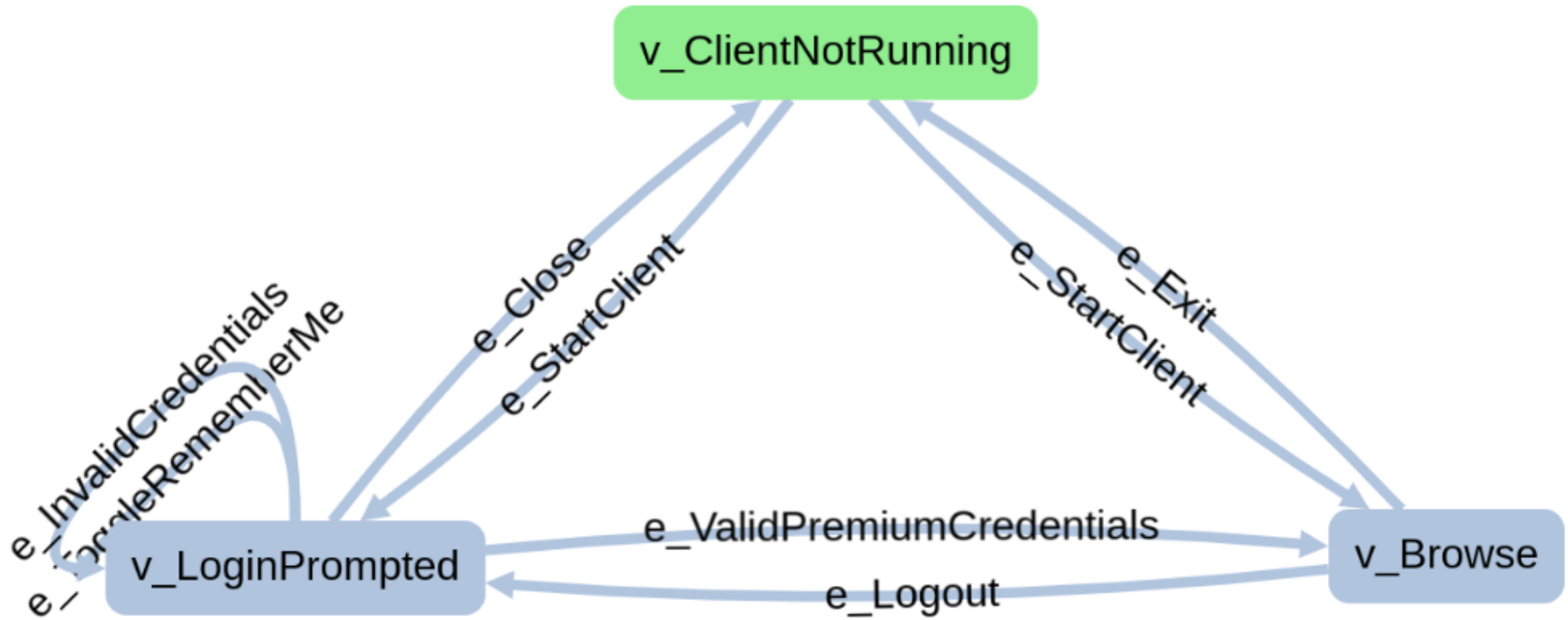
- https://graphwalker.github.io/
  - Wikis con la spiegazione di come funziona

Come creare il **modello**

- A GraphWalker model consists of 2 types of basic elements, the vertex and the edge.
- An **edge** represents an action, a transition.
- A **vertex** represents verification, an assertion.
- A model is a graph, which is a set of vertices and edges
  - From a model, GrapWalker will generate a path through it. A model has a start element, and a generator which rules how the path is generated, and associated stop condition which tells GraphWalker when to stop

Vertex and edges

- **An edge represents an action, a transition.**
- An action could be an API call, a button click, a timeout, etc. Anything that moves your System Under Test into a new state that you want to verify. But remember, there is no verification going on in the edge. That happens only in the vertex.
- **A vertex represents verification, an assertion.**
- A verification is where you would have assertions in your code. It is here that you verify that an API call returns the correct values, that a button click actually did close a dialog, or that when the timeout should have occurred, the System Under Test triggered the expected event.


v_ClientNotRunning
e_Close
e_StartClient
e_StartClient
e_Exit
e_InvalidCredentials
v_LoginPrompted
e_ValidPremiumCredentials
e_Logout
v_Browse

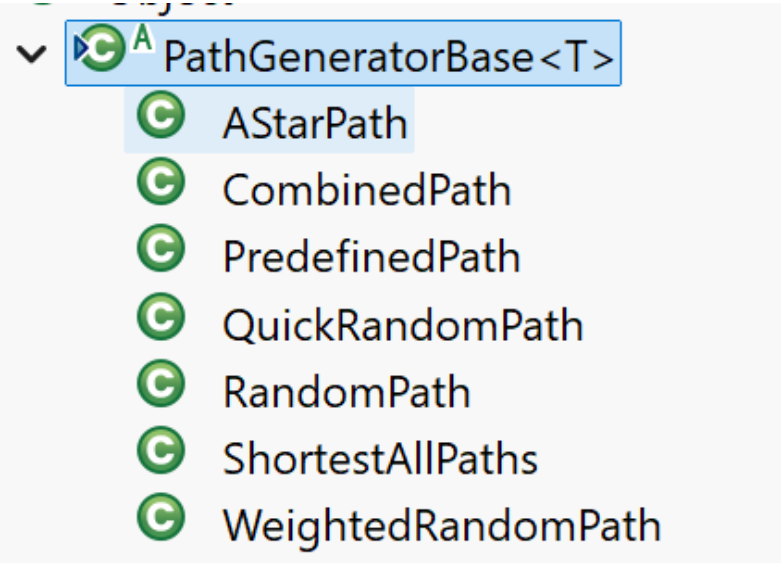
Come creare un modello (studio)

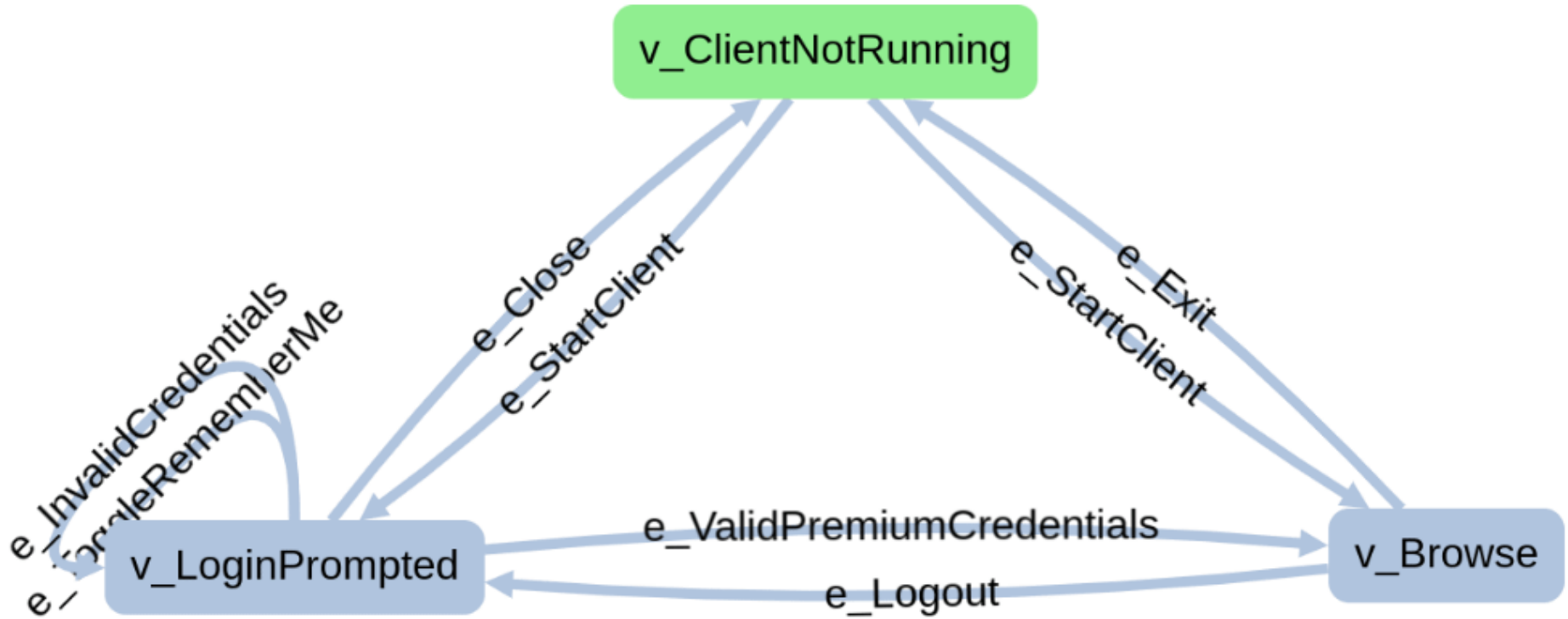
- Start the model editor:
  java -jar graphwalker-studio-4.3.3.jar

  (oppure doppio click)
- then open http://localhost:9090/studio.html in a web browser.
- + per creare un modello vuoto
- **V + mouse click sinistro** per creare un vertice
- Create an edge
  - While pressing the keyboard key **e** Click and hold the **left mouse button** on the first vertex.
  - Drag the mouse cursor to the second vertex and release the left mouse button over that vertex.

Posso simulare nello studio

- Usando i bottoni di play, etc
  - Con dei criteri:
  - **random(edge_coverage(100))**
  - **a_star(reached_vertex(v_ClientStarted))**




v_ClientNotRunning
e_Close
e_StartClient
e_StartClient
e_Exit
e_InvalidCredentials
v_LoginPrompted
e_ValidPremiumCredentials
e_Logout
v_Browse

Integrazione con un progetto

1. Definizione di un modello del SUT

2. Generazione dei test
    1. **Offline**: come testo poi da tradurre (vedi test execution)

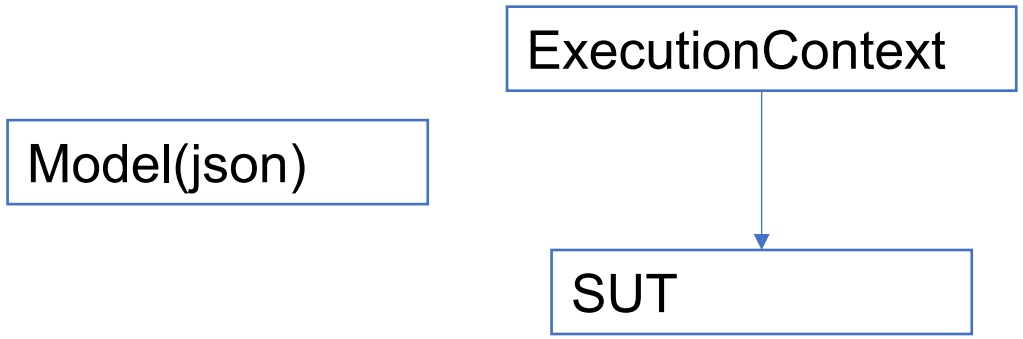
    2. Online: collegamento diretto con il codice (java)

Generazione offline

- A riga di comando
- Oppure con java – vedi demo

Generazione online

- Si collega il graphwalker con un codice java
- Online testing means that a model-based testing tool connects directly to a System Under Test (SUT) and tests it dynamically.
  - GraphWalker will start either as a WebSocket (default) or a HTTP REST server.
  - O codice java



Yakindu/Itemis Create

- Modello come machine di stato (statecharts)
- Bisogna definire l’interfaccia
- Si può simulare (tipo animazione)