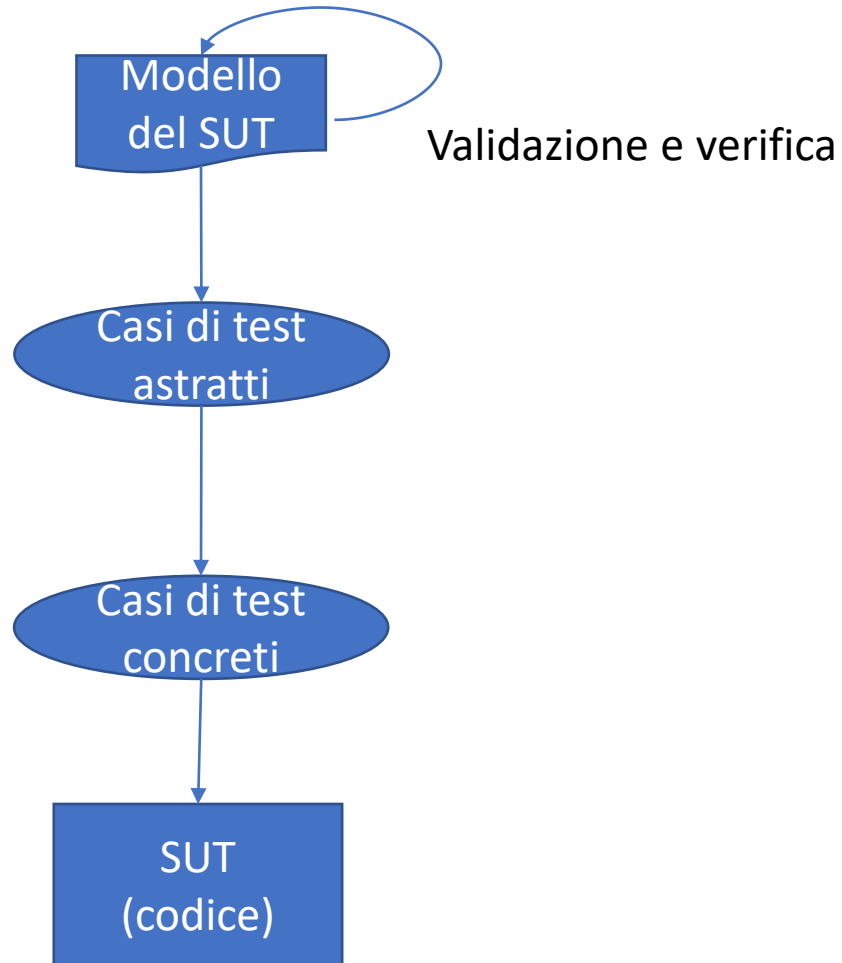


Model based testing approaches e tools

Tetsing e verifica del sw

AA 23/24

Flusso del model-based testing



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

Caso di studio

- Ho una lampada che può essere in tre stati
- OFF – STANDBY – ON

- Inizialmente in OFF
- Se premo il bottone ed è in STANDBY passa a ON
Se ripremo il bottone ed è ON passa in STANDBY
- Se stacco (poweroff) va sempre in OFF
Se attacco lo spina (poweron) e sono in OFF vado in STANDBY

Approcci/tool che vedremo

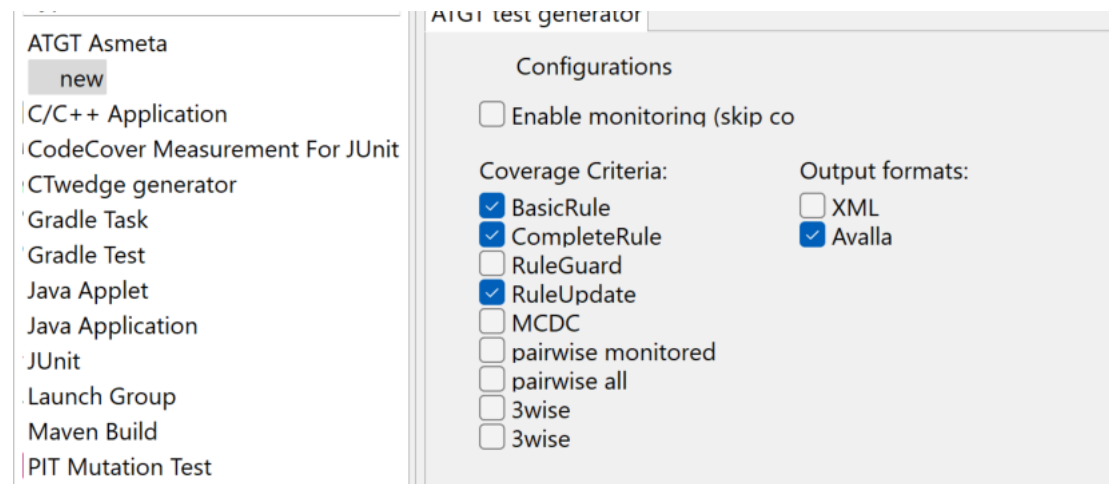
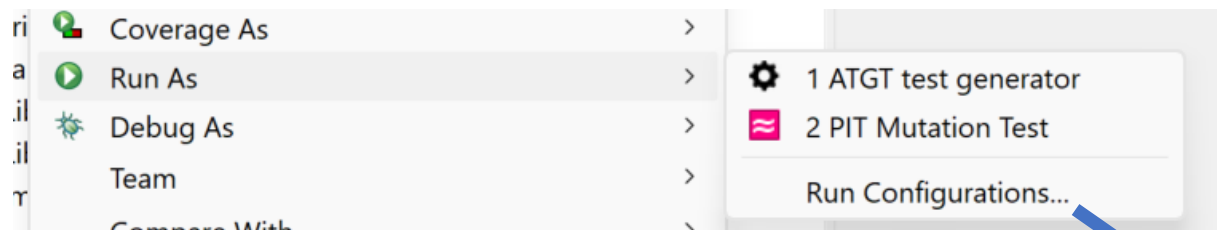
1. Asmeta + ATGT
2. GraphWalker
3. Itemis Create

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
 - In genere
 - Ogni scenario si traduce in un test di Junit
 - Set di una monitorata corrisponde ad un set di un campo della classe
 - 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

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

2. Graph walker

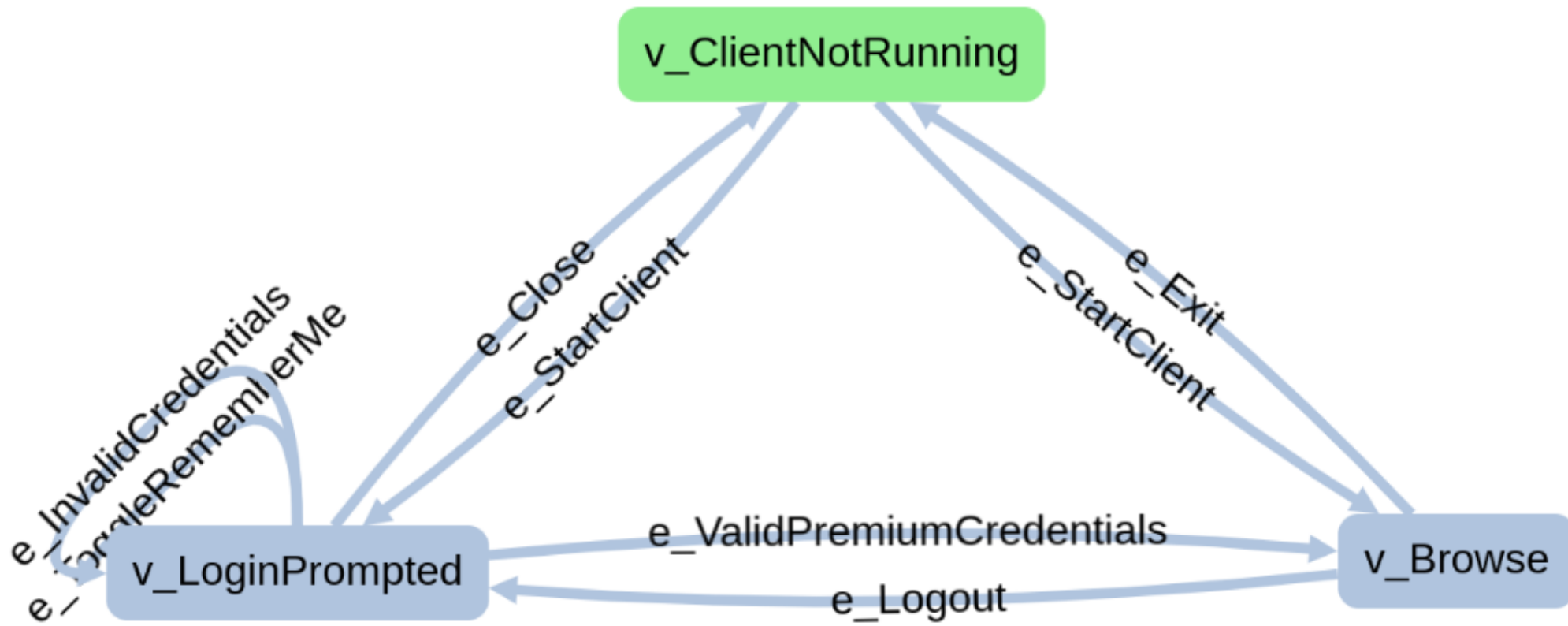
- Modello: macchina a stati con label su transizioni
- <https://graphwalker.github.io/>
 - Wikis con la spiegazione di come funziona

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, GraphWalker 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.



Come creare un modello (studio)

- Start the model editor

java -jar graphwalker-studio-4.3.2.jar

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))**
- Oppure
 - **GraphWalker Player**

Integrazione con un progetto

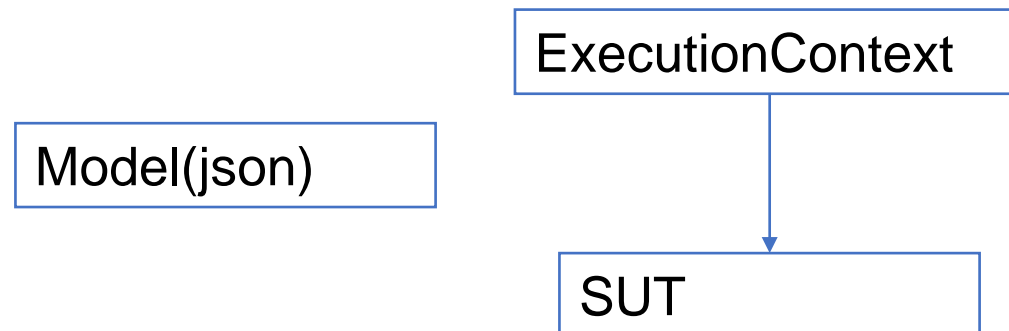
1. Definizione di un modello del SUT
2. Generazione dei test
 1. Offline: come testo poi da tradurre (vedi test execution)
 2. Online: collegamento diretto con il codice (java)

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.
 - 0 codice java



Yakindu/Itemis Create

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

