SOFTENG 2021

On the Composability of Behavior Driven Acceptance Tests

Tugkan Tuglular

Izmir Institute of Technology, Turkey

tugkantuglular@iyte.edu.tr

Assoc. Prof. Tugkan TUGLULAR, Ph.D.

Izmir Institute of Technology, Turkey

April 2021

Tugkan Tuglular received the B.S., M.S., and Ph.D. degrees in Computer Engineering from Ege University, Turkey, in 1993, 1995, and 1999.

He worked as a research associate at Purdue University from 1996 to 1998.

He has been with Izmir Institute of Technology since 2000.

After becoming an Assistant Professor at Izmir Institute of Technology, he worked as Chief Information Officer in the university from 2003-2007.

In addition to his academic duties, he acted as IT advisor to the Rector between 2010-2014.

In 2018, he became an Associate Professor in the Department of Computer Engineering of the same university.

He has more than 70 publications and an active record of duties with international and national conferences.

His current research interests include model-based testing and software quality with machine learning support.

Outline

- Motivation
  - Behavior Driven Development (BDD)
  - Behavior Driven Acceptance Tests (BDATs)
- Proposed Approach
  - Tagged Event Sequence Graphs
  - Elimination by Combination for finding missing BDATs
- Evaluation
  - E-commerce Software

Behavior Driven Development (BDD)

- Information gap between stakeholders and developers to be reduced.
- The scenarios are written in spoken language.
- Acceptance criteria is more understandable by all team members, e.g. Product Owners, Testers, UX Designer, Programmers.
- Comes with a structure : Given - When – Then (Gherkin).

Given some initial context (the givens),
When an event occurs,
Then ensure some outcome.

https://dannorth.net/introducing-bdd/#translations

Behavior Driven Acceptance Tests (BDATs)

- ensure that product is usable.
- ensure that product behaves as expected.
- can be utilized in
  - User Interface Testing (Mobile, Web)
  - Application Programming Interface (API) Testing

Motivation

- By transforming Gherkin clauses written in text to event-based graph models, we can
- Find missing acceptance tests and
- Incorporate advantages of model-based testing.

Advantages of Model-based Testing

- It starts with specifications by reinforcing the idea that QA involvement belongs at the beginning of the discovery stage.
- It forces testability into the product design when talking about the creation of models for a new/modified feature.
- It typically finds design and specification bugs before the code even exists.
- The automatic test suite generation will increase testing thoroughness, test coverage is guaranteed, and there is zero test suite maintenance.

https://saucelabs.com/blog/the-challenges-and-benefits-of-model-based-testing

Model-based Testing Process

- Start by modeling the Software Under Test (SUT)
- Derive test cases from the model
- Execute test cases
  - use model as test oracle
  - record coverage
  - trace to model
- Modify model as needed
- Repeat steps

P. C. Jorgensen, Software Testing: A Craftsman's Approach, 4th Edition, Auerbach Publications, 2013.

Event-based Modeling

- Event Sequence Graphs (Belli, 2001)
- Event-based formal model
  - inputs and actions are merged as events and assigned to the vertices of an event transition graph.
- They can be used for system modeling and test generation.

F. Belli (2001). “Finite-State Testing and Analysis of Graphical User Interfaces“, 12th International Symposium on Software Reliability Engineering, ISSRE 2001, 34-43.

Advantages of Event-based Modeling

- Testability is dominated by two practical problems
  - How to provide the test values to the software
  - How to observe the results of test execution
- Controllability
  - How easy it is to provide a program with the needed inputs, in terms of values, operations, and behaviors
- Observability
  - How easy it is to observe the behavior of a program in terms of its outputs, effects on the environment and other hw and sw components

P. Ammann and J. Offutt, Introduction to Software Testing, 2nd Edition, Cambridge University Press, 2016.

Event Sequence Graphs (ESGs)

An *event sequence graph* ***ESG*** $= (V, E, \Xi, \Gamma)$ is a directed graph where

$V \neq \emptyset$ is a finite set of vertices (nodes),

$E \subseteq V \times V$ is a finite set of arcs (edges),

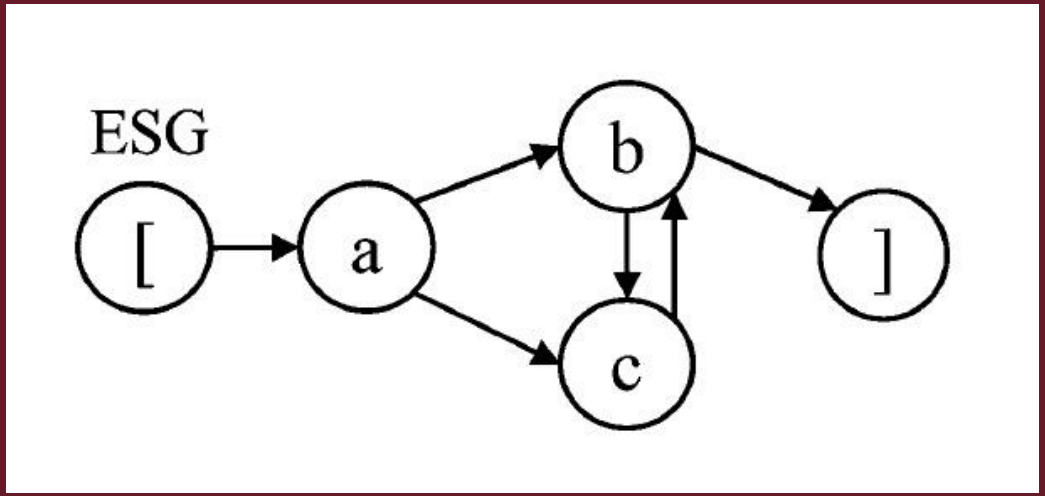
$\Xi, \Gamma \subseteq V$ are finite sets of distinguished vertices

with $\xi \in \Xi$, and $\gamma \in \Gamma$, called entry nodes and exit nodes.

F. Belli (2001). "Finite-State Testing and Analysis of Graphical User Interfaces", 12th International Symposium on Software Reliability Engineering, ISSRE 2001, 34-43.

Event Sequence Graphs (ESGs)

- An ESG with a as entry and b as exit and pseudo vertices [,]
- Each edge marked as a legal event pair (EP).

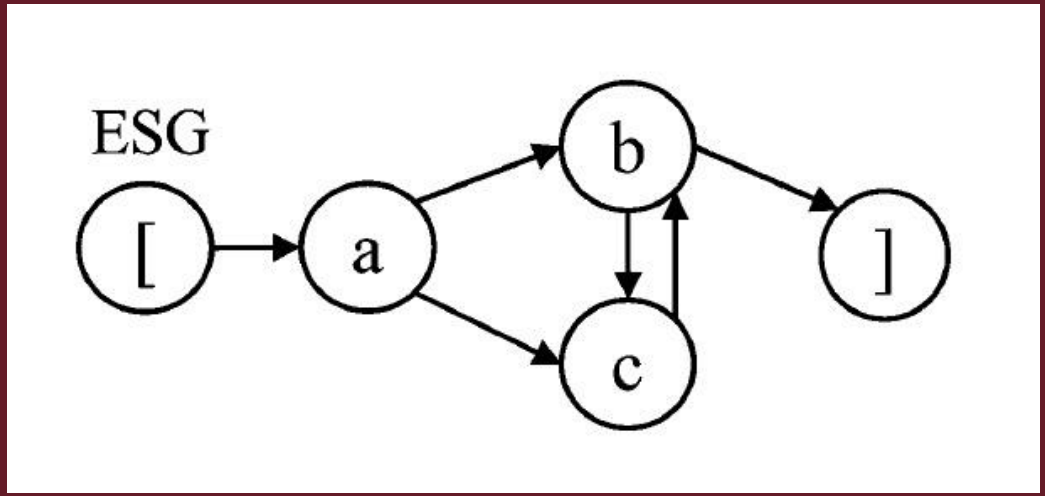


EPs:

a, b
a, c
b, c
c, b

Event Sequence Graphs (ESGs)

- Complete event sequence (CES) represents a walk through the ESG.

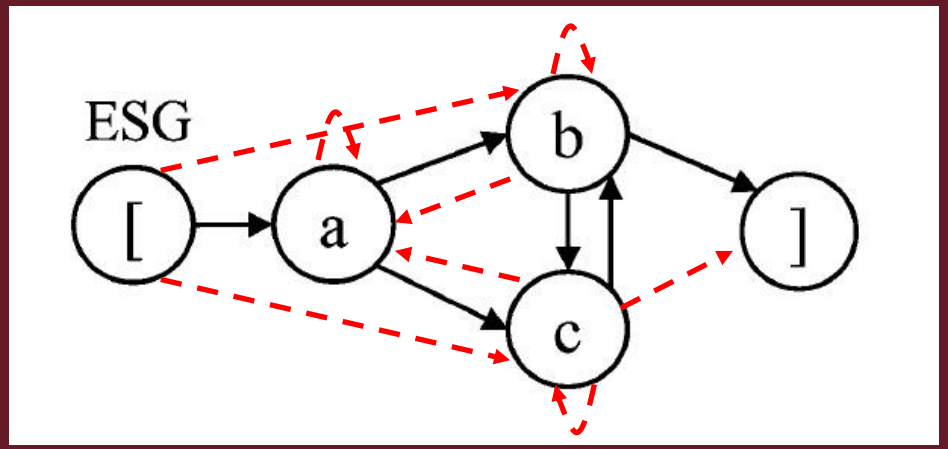


CES:

3: [, a, c, b,],
4: [, a, b, c, b,],

Event Sequence Graphs (ESGs)

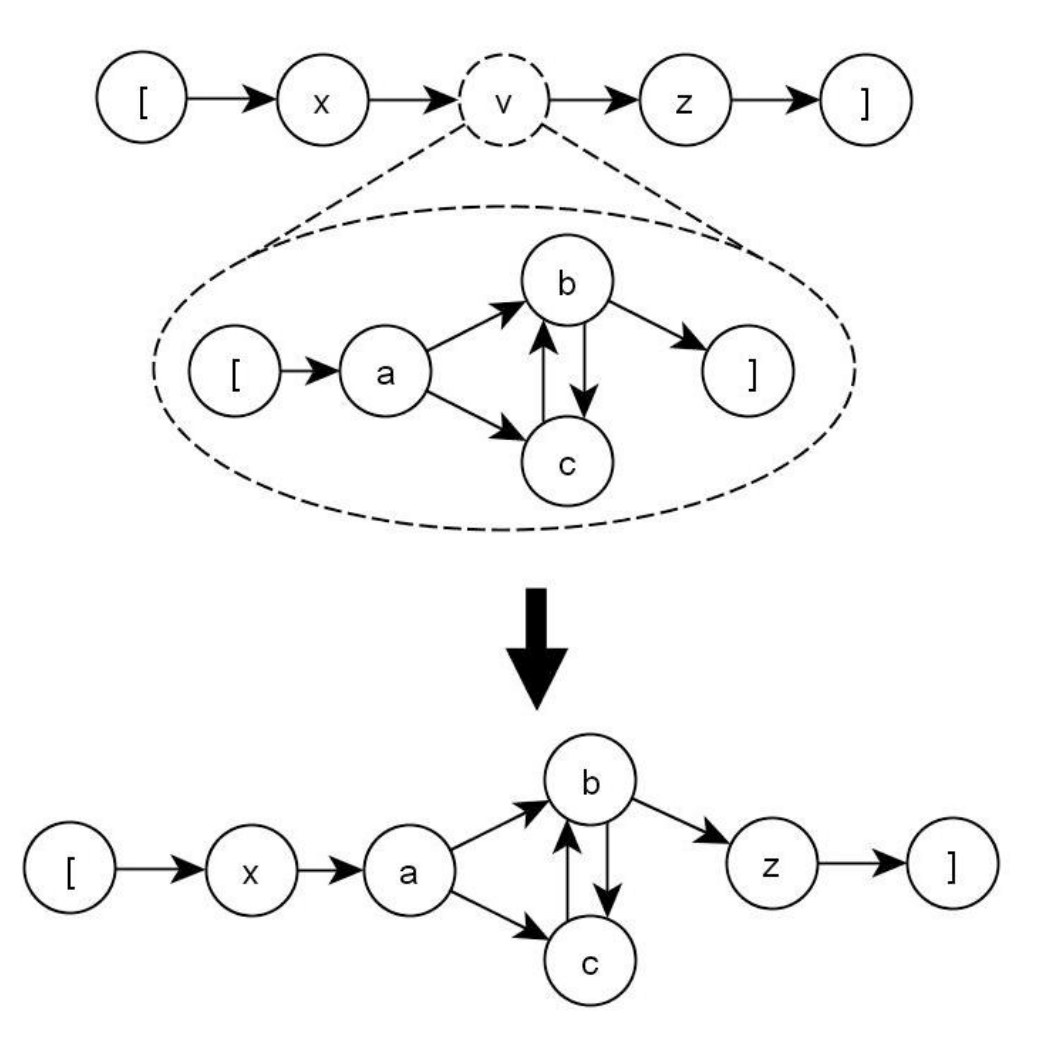
- Faulty (or illegal) event pairs (FEP) are introduced as the edges of the corresponding $\overline{\text{ESG}}$
  - Faulty CESs (FCESs) constructed using FEPs.



FCES:

2: [, a, a,
3: [, a, b, a,
3: [, a, b, b,
3: [, a, c, a,
3: [, a, c, c,
1: [, b,
1: [, c,

Event Sequence Graphs (ESGs)



- Refinement of a vertex v and its embedding in the refined ESG

CES:

6: [, x, a, b, c, b, z,],
5: [, x, a, c, b, z,],

ESG Test Generation

- ***Input:*** ESG
- ***Output:*** Test set with respect to model-based coverage criterion
- Two objectives for the test case generation procedure:
  - generation of CESs,
  - generation FCESs from the complement of ESG.
- Test case generation algorithm generates tests that cover both;
  - All event pairs in ESG,
  - All faulty event pairs of the CESG.

ESG Test Generation Algorithm

Input: an ESG with
n:= number of the functional units (modules) of the system that fulfill a well-defined task
length:= required length of the event sequences to be covered
FOR unit 1 TO n DO
BEGIN
Generate appropriate ESG and $\overline{\text{ESG}}$
FOR k:=2 TO length DO
BEGIN
Cover all ESs of length k by means of CESs subject to minimizing the number and total length of the CESs
END
Cover all FEPs of unit by means of FCESs subject to minimizing the total length of the FCESs
END
Apply the test set given by the selected CESs and FCESs to the SUT.
Observe the system output to determine whether the system response is in compliance with the expectation.

k = 2 means

edge (EP) coverage

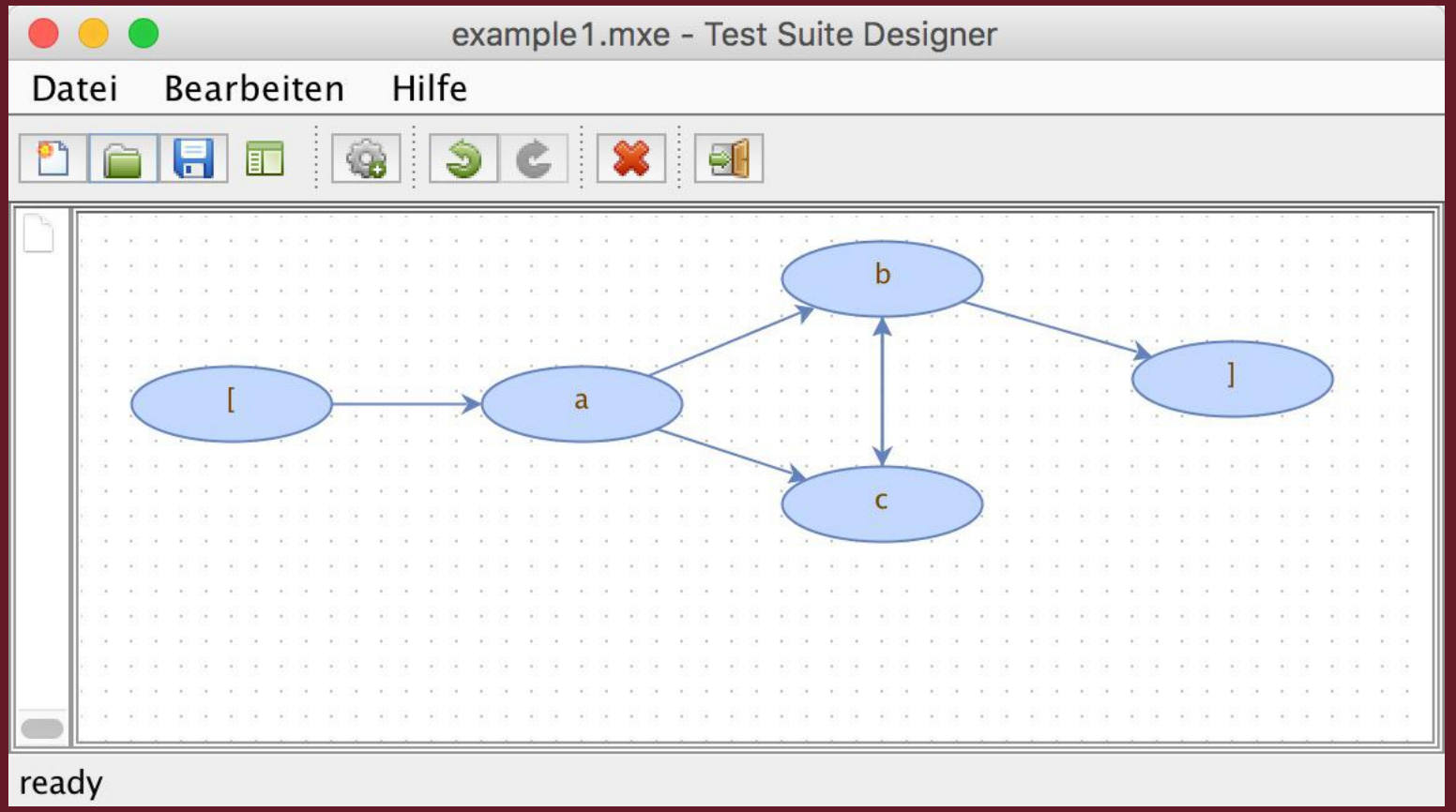
k = 3 means

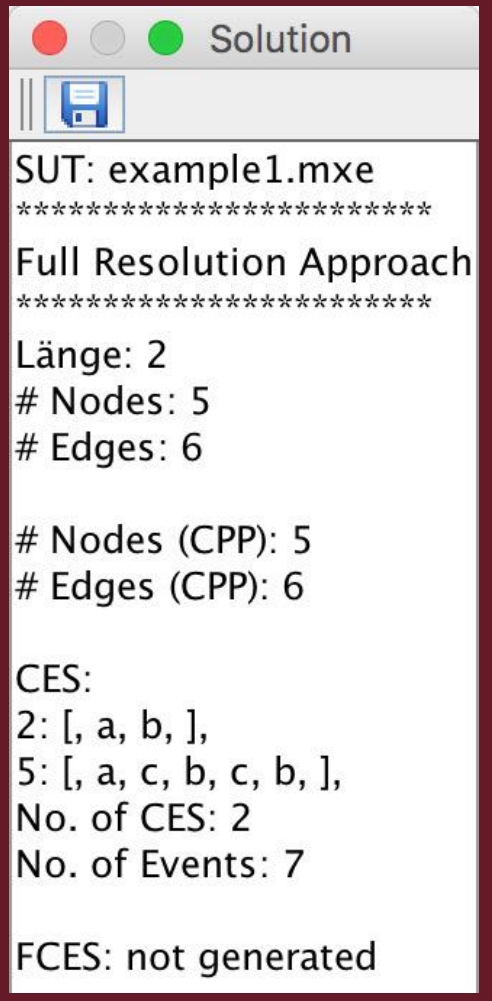
Edge pair (ET) coverage

…

F. Belli, N. Nissanke, C. J. Budnik, and A. Mathur, "Test generation using event sequence graphs", Technical Reports and Working Papers, 2005.

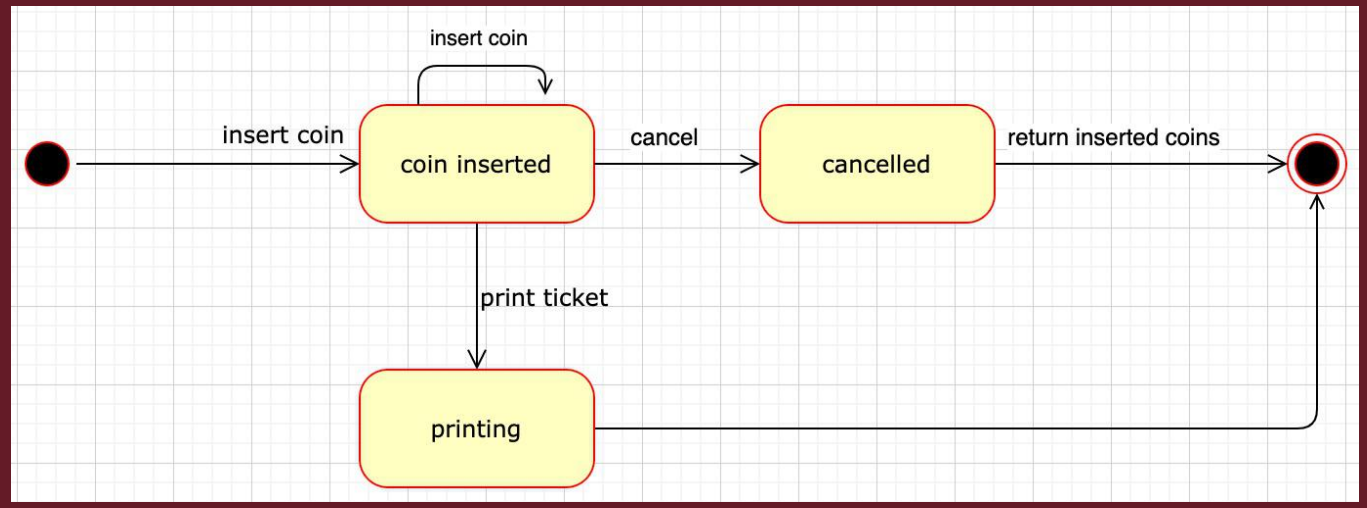
ESG Tool

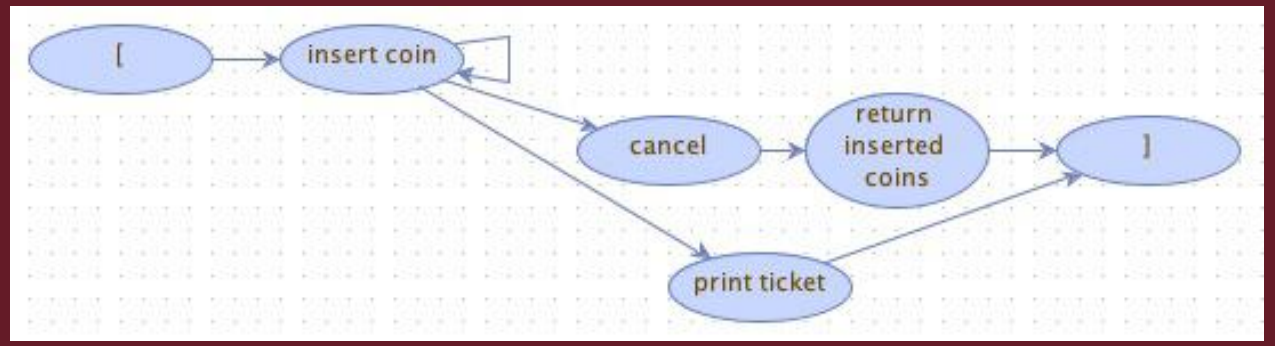




http://download.ivknet.de/index.php

State Machines to ESGs





Solution

SUT: simple_ticket_machine.mxe
****
Full Resolution Approach
****
Länge: 2
Nodes: 6
Edges: 7

Nodes (CPP): 6
Edges (CPP): 7

CES:
3: [, insert coin, insert coin, print ticket,],
3: [, insert coin, cancel, return inserted coins,],
No. of CES: 2
No. of Events: 6

FCES:
2: [, insert coin, return inserted coins,
3: [, insert coin, cancel, insert coin,
3: [, insert coin, cancel, cancel,
3: [, insert coin, cancel, print ticket,
3: [, insert coin, print ticket, insert coin,
3: [, insert coin, print ticket, cancel,
3: [, insert coin, print ticket, print ticket,
3: [, insert coin, print ticket, return inserted coins,
4: [, insert coin, cancel, return inserted coins, insert coin,
4: [, insert coin, cancel, return inserted coins, cancel,
4: [, insert coin, cancel, return inserted coins, print ticket,
4: [, insert coin, cancel, return inserted coins, return inserted coins,
1: [, cancel,
1: [, print ticket,
1: [, return inserted coins,
No. of FCES: 15
No. of Events: 42

Proposed Approach

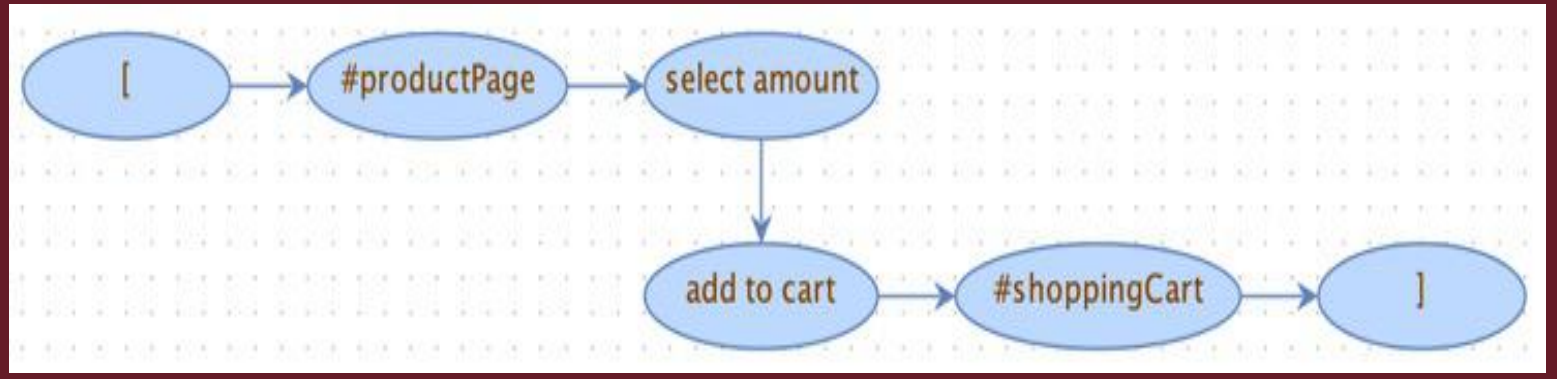
- The proposed approach improves completeness of a BDAT test suite and enables coverage-based test sequence generation.
- With the assumption that Gherkin clauses can be represented by events, the proposed approach suggests use of event sequence graphs (ESGs) for modeling BDATs.
- To model a BDAT as an ESG, **ESGs are extended with tags**.

Tagged ESG

A tagged ESG is an ESG, where a vertex may contain a tag instead of an event.

Scenario: cart02 - Adding a product to cart
Given I am on a product detail page **#productPage**
When I select the amount
And I click the add to cart button
Then the product is added to my shopping cart **#shoppingCart**

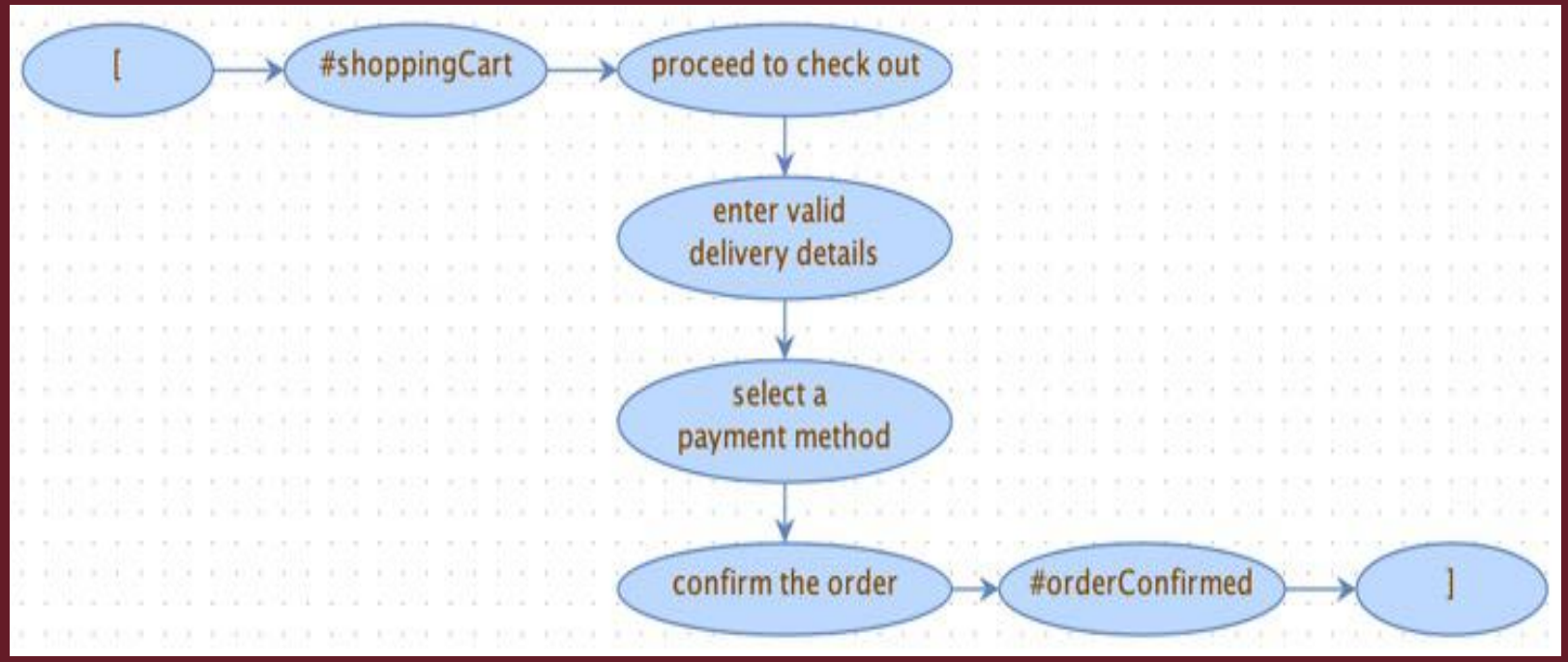
A tagged ESG is used to represent a BDAT.



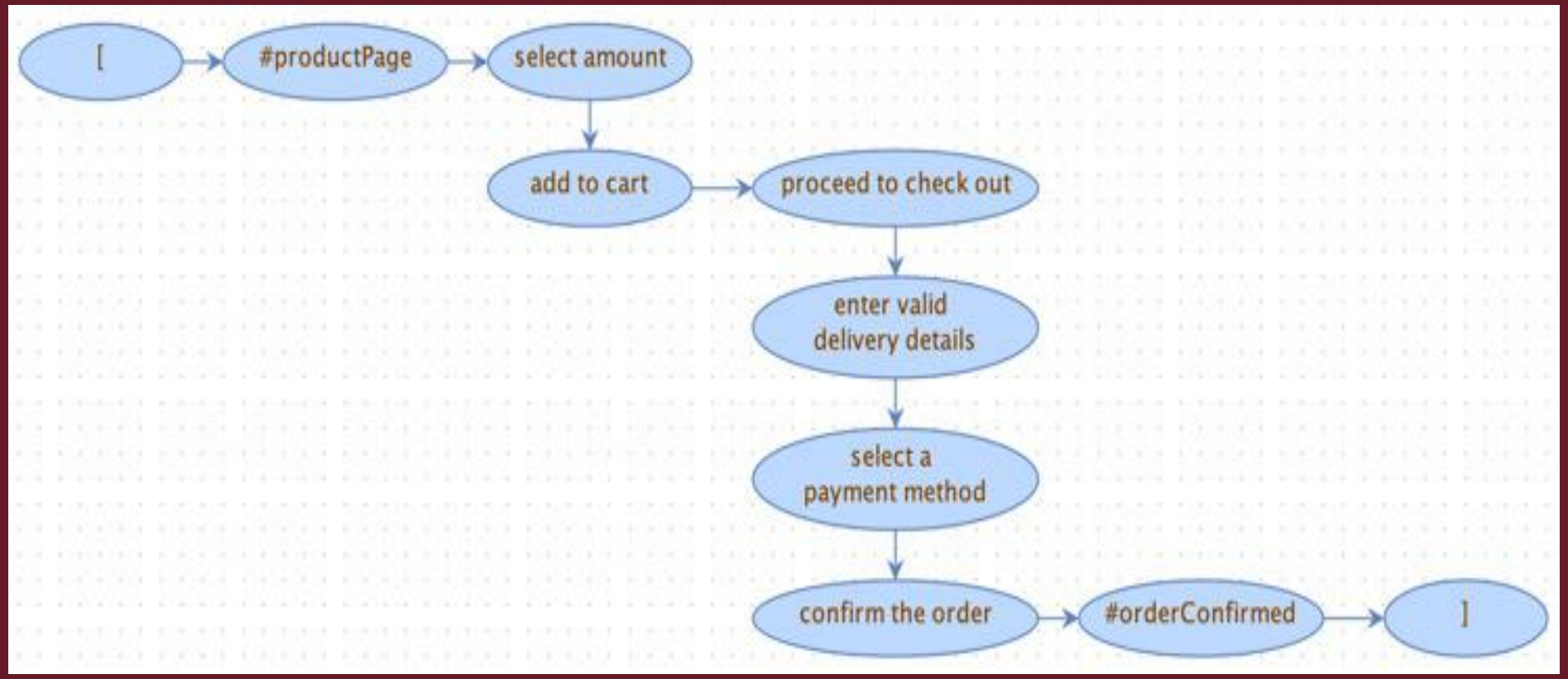
Tagged ESG

Scenario: check01 - Successful checkout
Given I have added an item to my shopping bag **#shoppingCart**
When I proceed to the check out
And I enter valid delivery details
And I select a payment method
And I confirm the order
Then I am redirected to the thank you page **#orderConfirmed**

#shoppingCart tag is used as a connection point and then eliminated.



Combining two BDATs on tagged ESG



Finding missing BDATs

- **Elimination by Combination** for finding missing BDATs
- Once two BDATs are combined using a tag, that tag is eliminated.
- Therefore, all possible tagged scenarios or their graphical representations, i.e., tagged ESGs, are combined.
- A combined tagged ESG may be combined with another simple or combined tagged ESG.
- The goal is to reach an ESG without any tags.

Finding missing BDATs

- **Elimination by Combination** for finding missing BDATs
- After all possible combinations are completed, a tag remained on a tagged ESG indicates that there is a missing BDAT.
- More than one tag remaining may mean more missing BDATs.

Finding missing BDATs

```
Scenario: acc03 - Check orders
    Given I am logged in on the site #atHome
    When I navigate to my orders
    Then I see a list of my orders
    And I can open an order to see the order details #orderDetail
```

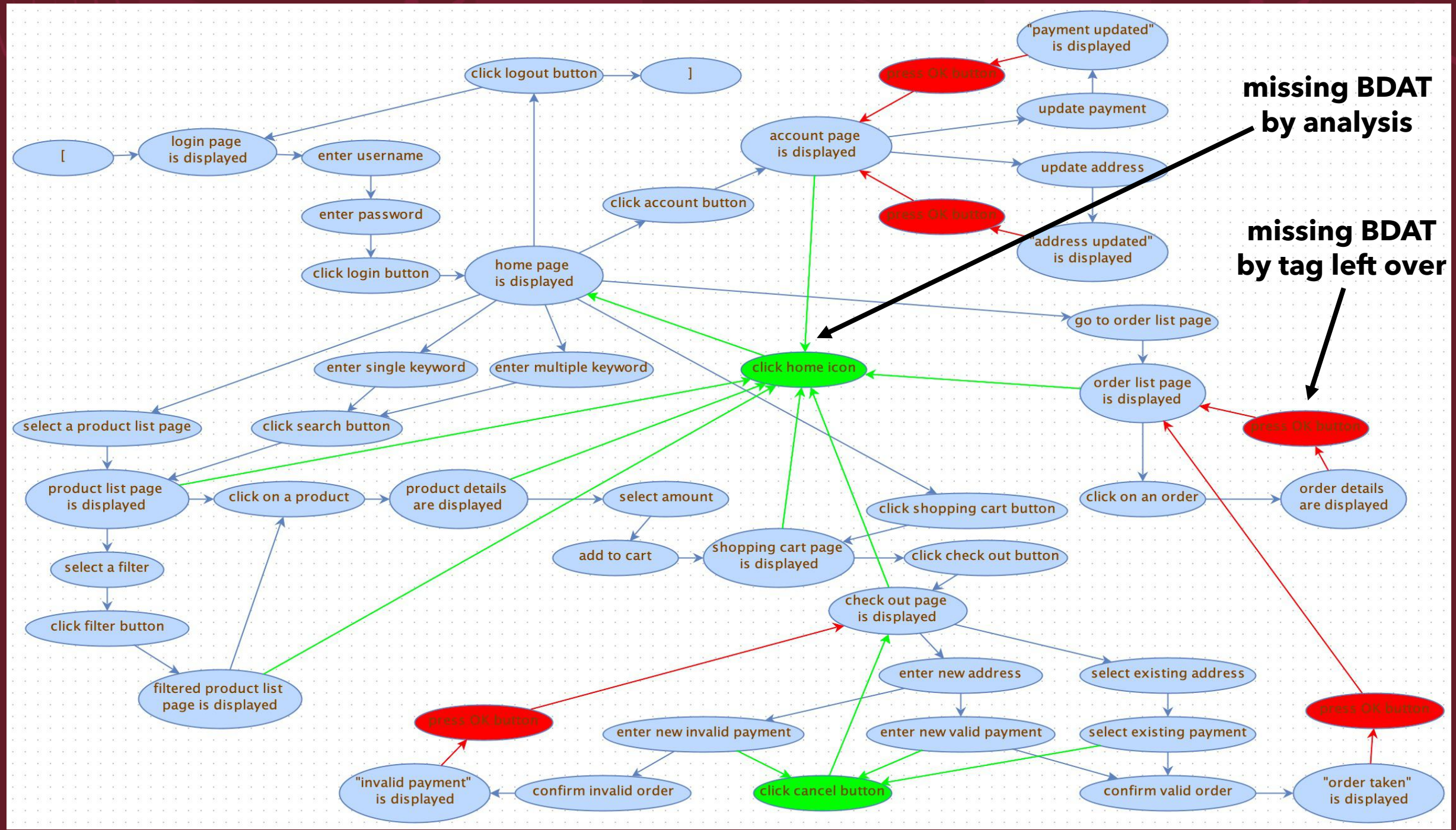
This BDAT is the only Gherkin scenario that has the tag **#orderDetail**.

Since there is no match, it indicates that a BDAT that starts with **#orderDetail** tag is missing. Complete this missing BDAT as follows:

```
Scenario: acc10 - Back to order list page
    Given #orderDetail
    When I press OK button
    Then order list page is displayed #orderList
```

Evaluation

- Proposed approach is applied to an existing test suite for an e-commerce software. https://github.com/spriteCloud/ecommerce -cucumber-web-test-automation-suite
- Existing test suite has **15 BDATs**, or scenarios, with **64 Gherkin clauses**. Clause per scenario ratio is **4.26**.
- After applying the proposed approach, we end up with **24 BDATs** and **85 Gherkin clauses**. There are 9 new scenarios but only 5 of them are missing scenarios. The other 4 scenarios are introduced to simplify and standardize some original scenarios. So, clause per scenario ratio is descended to **3.54**.


missing BDAT
by analysis
missing BDAT
by tag left over
[
login page
is displayed
enter username
enter password
click login button
click logout button
]
home page
is displayed
click account button
account page
is displayed
"payment updated"
is displayed
press OK button
update payment
update address
press OK button
"address updated"
is displayed
go to order list page
order list page
is displayed
press OK button
click on an order
order details
are displayed
click home icon
enter single keyword
enter multiple keyword
click search button
select a product list page
product list page
is displayed
click on a product
product details
are displayed
select amount
add to cart
click shopping cart button
shopping cart page
is displayed
click check out button
check out page
is displayed
select a filter
click filter button
filtered product list
page is displayed
enter new address
select existing address
press OK button
enter new invalid payment
enter new valid payment
select existing payment
press OK button
"invalid payment"
is displayed
confirm invalid order
click cancel button
confirm valid order
"order taken"
is displayed

Test Sequence by TSD Tool

- No. of Nodes: 50
- No. of Edges: 70
- CES with 111 events:
- [, login page is displayed, enter username, enter password, click login button, home page is displayed, go to order list page, order list page is displayed, click on an order, order details are displayed, press OK button, order list page is displayed, click home icon, home page is displayed, click shopping cart button, shopping cart page is displayed, click check out button, check out page is displayed, enter new address, enter new invalid payment, confirm invalid order, "invalid payment" is displayed, press OK button, check out page is displayed, enter new address, enter new invalid payment, click cancel button, check out page is displayed, enter new address, enter new valid payment, click cancel button, check out page is displayed, select existing address, select existing payment, click cancel button, check out page is displayed, enter new address, enter new valid payment, confirm valid order, "order taken" is displayed, press OK button, order list page is displayed, click home icon, home page is displayed, enter multiple keyword, click search button, product list page is displayed, select a filter, click filter button, filtered product list page is displayed, click on a product, product details are displayed, select amount, add to cart, shopping cart page is displayed, click home icon, home page is displayed, enter single keyword, click search button, product list page is displayed, click on a product, product details are displayed, click home icon, home page is displayed, select a product list page, product list page is displayed, click home icon, home page is displayed, click account button, account page is displayed, update payment, "payment updated" is displayed, press OK button, account page is displayed, update address, "address updated" is displayed, press OK button, account page is displayed, click home icon, home page is displayed, click shopping cart button, shopping cart page is displayed, click check out button, check out page is displayed, select existing address, select existing payment, confirm valid order, "order taken" is displayed, press OK button, order list page is displayed, click home icon, home page is displayed, select a product list page, product list page is displayed, select a filter, click filter button, filtered product list page is displayed, click home icon, home page is displayed, click shopping cart button, shopping cart page is displayed, click check out button, check out page is displayed, click home icon, home page is displayed, click logout button, login page is displayed, enter username, enter password, click login button, home page is displayed, click logout button,]

Conclusion

- This paper proposes an approach to represent BDATs using ESGs.
- With the proposed approach, the test designer not only finds and completes missing BDATs but also combines them to know which BDAT can be executed after which BDAT.
- When the final composition is supplied to the TSD tool, it automatically generates a test sequence that covers all BDATs.
- So, the proposed approach improves testability of BDATs.