Using Petri Nets To Test Concurrent Behavior Of Web Applications

Sunitha Thummala & Jeff Offutt

Presented by

Omar Al-Debagy

Outline

- Introduction
- Scope of the Research Paper
- Petri Nets
- Modelling Web Applications with Petri Nets
- Example
- Test Criteria
- Evaluation
- Experimental Design
- Experimental Results
- Discussion
- Conclusion

Introduction

- A major challenge for web applications is concurrency. Concurrency occurs when multiple activities can be run at the same time and possibly interact with each other.

- Donley and Offutt discuss various problems in testing web applications and describe four concurrency related scenarios:

    1. Single user session / multiple tabs

    2. Single user / multiple sessions

    3. Multiple users

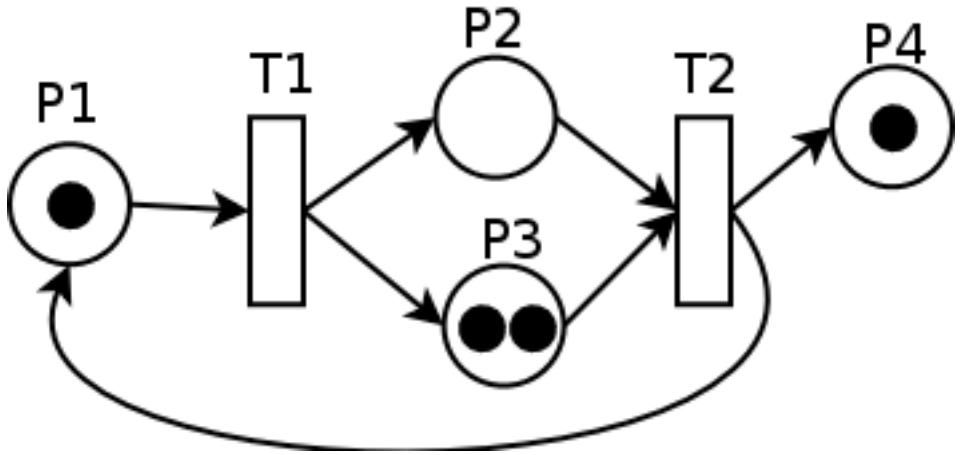
    4. Asynchronous callbacks

Scope of the research paper

- The research focuses on the presentation layer components of a web application
- Their focus is on the synchronous request-response cycle triggered by user activity
- They consider concurrent user behaviors addressed by the first three scenarios in the above list.
- This research paper addresses the modeling of concurrent user behavior of synchronous request-response web applications using Petri nets.
- They define Petri net model- based coverage criteria that try to detect software failures that are potentially visible to end users during normal interaction sequences with web apps.

Petri Nets

- Petri Nets: *Petri nets* (PN) are bipartite graphs introduced by Carl Adam Petri
- They have been used to model asynchronous, distributed, concurrent, non-deterministic, and timed systems.
- The main components of a Petri net are:
  - Places
  - Transitions
  - Arcs
  - Tokens



Petri Nets

- Bernardinello and De Cindio classified Petri nets into three levels.
- **Level one** Petri nets have places that have at most one unstructured token.
- **Level two** Petri nets allow more than one multiple unstructured tokens.
- **Level three** Petri nets use structure tokens that represent data types.
- This research uses level two Petri nets that describe control flow.

Modeling Web Applications with Petri Nets

- This research uses Petri nets to develop tests for web applications.
- Petri nets have features that can be used to represent multiple users browsing a web application simultaneously.
- Petri nets also offer extensions to model time and different data types.

Modeling Web Applications with Petri Nets

- Liveness properties of Petri nets can be used to determine whether any transition will eventually become un-fireable.

- The test method has two major steps.

  - Step one is automated with a tool they developed, the Web Application Petri net Generator (WAPG), which extracts the navigational structure of web apps.

  - Step two is performed manually by the tester.

Step 1

- Step 1: Use WAPG to derive the Petri net model from the web application source code (automated)
- Extract each component (HTML, servlet, or JSP) to form a place in the Petri net.
- Extract links, method calls, or events that cause control transfers between components to form the transitions of the net.
- Extract logic expressions to form guards on transitions.
- Use the places, transitions, and guards to construct the Petri net model.

Step 2

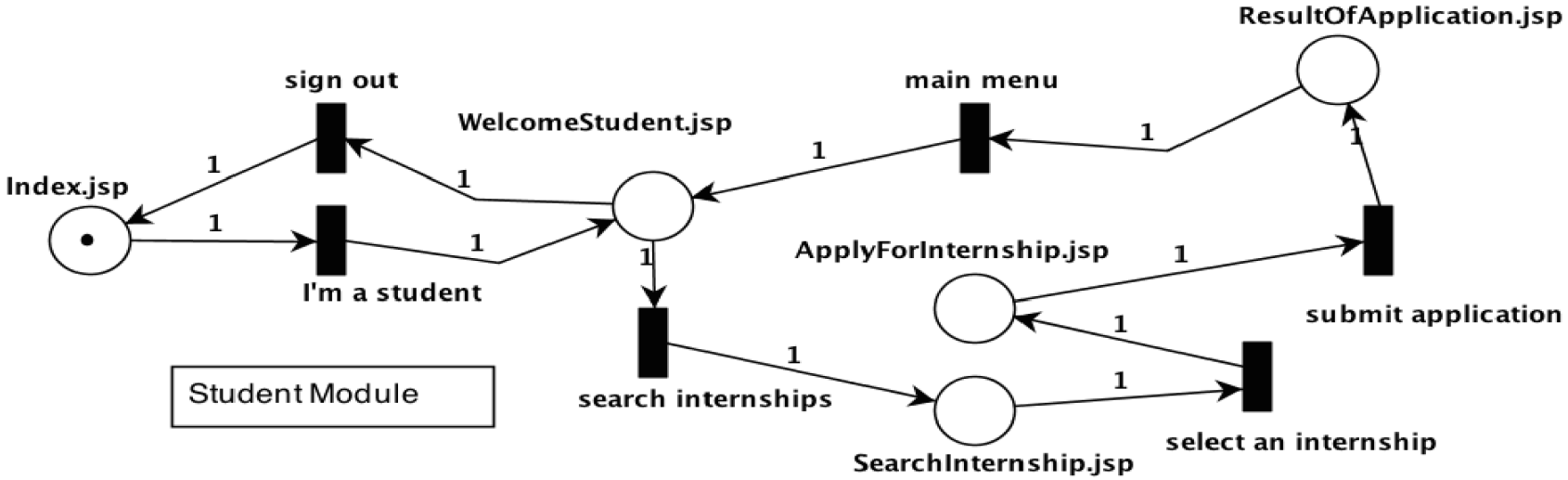
- Decompose the initial Petri net into separate Petri nets

- Add concurrent behavior to the model generated by WAPG.
  - Single Session Concurrent Behavior (SSCB)
  - Multiple Sessions Concurrent Behavior (MSCB)

- Design tests based on the coverage criteria.

Example

- Petri net model extracted for the student module of the internship web app by WAPG



Example – Tester Part

The tester does three things in step two:

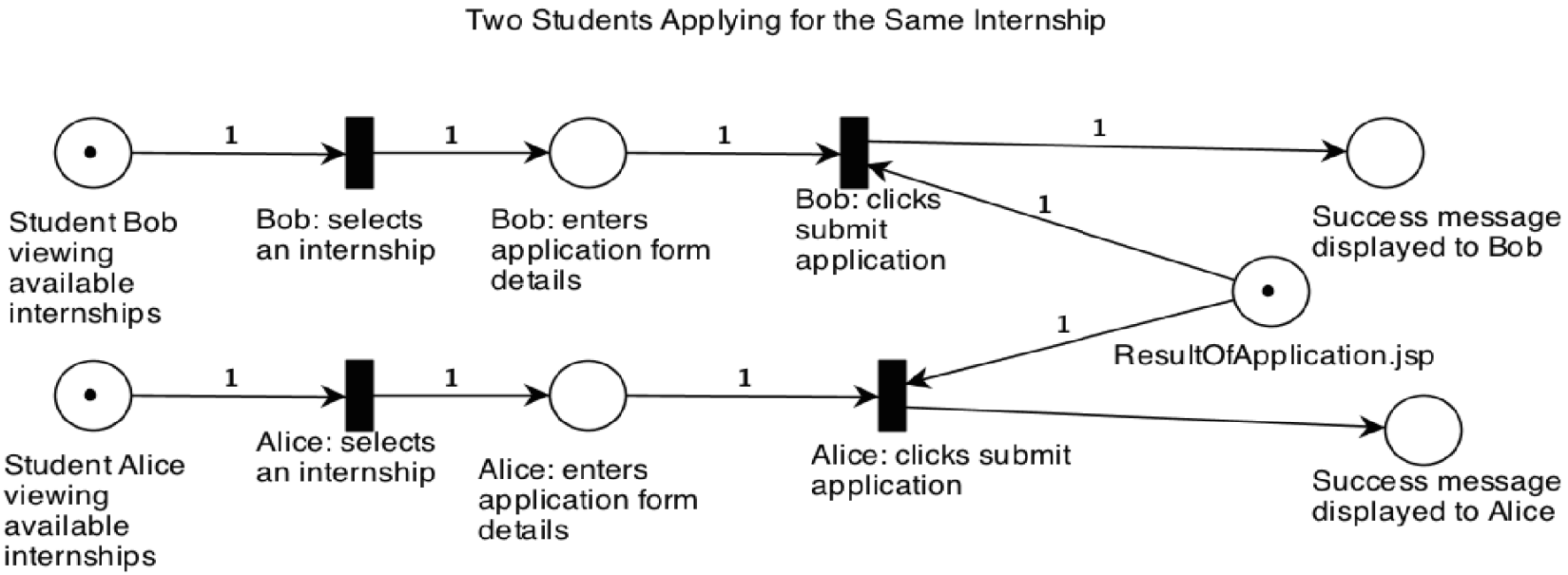
1. Creates two separate Petri nets by dividing the internship web app into two modules, manager and student.

2. Models two types of concurrent behavior
   - Single session concurrent behavior
   - Multiple sessions concurrent behavior

3. Derives tests based on the criteria

Example – Tester Part

- Example of Multiple sessions concurrent behavior



Test Criteria for Petri Net Modeled Web Applications

This research uses three criteria, two that are novel to this paper.

- The first, **Structural and Behavioral Analysis Coverage** (SBAC).
  - Defines tests using the structural and behavioral properties of the Petri net model such as liveness, reachability, and deadlock.

Test Criteria for Petri Net Modeled Web Applications

- The second, ***Concurrent Session Behavior Coverage Criteria*** *(CSBCC)*
  - This criteria focus on session variables, single sessions, multiple sessions, and tabbed behaviors.
- The third criterion is **Restrictive Active Clause Coverage** (RACC): Logic-based tests are designed for the guard conditions of the Petri net.

Evaluation

Research Questions

- Can Petri nets be used to model the concurrent behavior of web applications?
- Can the proposed method help find faults in web applications?

Experimental Design

- Their experiment obtained three separate sets of tests for each web app.
- The tests were designed by two software engineers
- Tests were based on the functional requirements.
- The tests were run by hand. Each tester spent about four hours to generate and execute requirements-based tests.

Experimental Design

- Then they generated three sets of tests for each web app using the coverage criteria.
- Ten J2EE web applications were used.
- Different types of web applications, like, internship application, student survey, employee directory and so on.

Experimental Results

Sub-ject	Comp-onents	LOC	Tests							
			Requirements		Petri Net Model-Based Tests					
			Tests	Failures	SBAC Tests	Failures	RACC Tests	Failures	CSBCC Tests	Failures
1	20	1473	72	2	19	2	11	0	28	3
2	16	1077	33	2	17	3	7	1	12	1
3	9	691	31	0	11	1	4	0	7	1
4	24	1578	57	1	23	2	6	0	16	2
5	13	1196	42	1	17	2	10	1	10	1
6	18	1500	39	1	18	2	11	0	14	1
7	20	1286	35	1	24	2	9	0	13	1
8	29	4554	52	1	25	1	15	0	21	2
9	22	1703	37	0	15	2	6	0	12	1
10	22	2477	61	0	12	2	6	0	11	2
Total	**193**	**17,535**	**459**	**9**	**181**	**19**	**85**	**2**	**144**	**15**
Tests Per Failure			**51**		**9.5**		**42.5**		**9.6**	

- Experimental results of applying four test sets to ten web applications.
- The last row gives the ratio of the number of tests per failure.
- Used WAPG to extract the Petri net models for the ten web applications and generated a total of 410 tests

Experimental Results

- In summary, about 9% percent of the 410 tests found failures.
- All nine of these failures were also detected by the WAPG tests.
- Only about 2% of the requirements-based tests found failures.

TABLE V
TYPES OF FAILURES FOUND USING REQUIREMENTS-BASED TESTS

Failure Type	Failure Description	Number of failures in this category
1	HTTP 404 file not found error displayed	1
2	Duplicate records added	1
3	Did not display expected message for invalid input values	3
4	Transfer was not successful	1
5	Runtime exceptions	2
6	Incorrect information displayed	1
	Total	**9**

TABLE III
TYPES OF FAILURES FOUND USING WAPG

Failure Type	Failure Description	Number of failures found by WAPG tests
1	Runtime exceptions	7
2	Incorrect information displayed	8
3	User id creation failed	1
4	Employee profile creation failed	1
5	Doubled items in cart	1
6	HTTP 404 file not found error displayed	2
7	Duplicate records added	3
8	Liveness issues such as allowing multiple invalid attempts for username and password, or allowing one user to place multiple orders in a single day	5
9	Deadlock in the Petri net structure of the web app	1
10	Did not display expected message for invalid input values	5
11	Transfer was not successful	1
12	Can checkout with standard shipping charge for zero items in cart	1
	Total	**36**

Discussion

- They successfully used WAPG to generate the Petri net model of the web application, then used it to create tests that found 36 software faults. Thus, the answers to RQ1 and RQ2 are clearly yes.

- The majority of the faults found were by their approach and not the requirements-based tests from three different testers.

Conclusion

- Their first contribution is a technique to use Petri nets to model web applications for testing purposes, implemented in a tool that automatically extracts the model.
- Second, they defined novel coverage criteria to test concurrent behavior of web applications involving HTTP browser-based sessions.
- The positive empirical evidence indicates that the criteria-based tests developed by their approach were more effective than requirements-based tests.

Thank You