

Analysis of web system by using model-based testing with selenium tool

Mrs. Nayan Mulla¹, Prof. S.B. Takmare²

*^{1,2}Department of Computer Science and Engg.,
Bharati Vidyapeeth's College of Engineering,
Kolhapur, Maharashtra, (India)*

ABSTRACT

Modern business applications predominantly rely on web technology, enabling software vendors to efficiently provide them as a service, removing some of the complexity of the traditional release and update process. To increasing web application accuracy and speed user process crawler model. Cutting edge business applications transcendentally depend on web innovation, empowering programming sellers to give proficiently them as an administration, uprooting a portion of the multifaceted nature of the customary discharge and overhaul process. While this encourages shorter, more productive and successive discharge cycles, it obliges persistent testing. Having knowledge into application conduct through unequivocal models can to a great extent bolster improvement, testing and support. Model-based testing permits effective test creation taking into account a depiction of the states the application can be in and the moves between these states. As determining conduct models that are sufficiently exact to be executable by a test computerization device is a hard assignment, an option is to concentrate them from running applications.

Keywords—Specification mining; dynamic analysis; model-based testing; web system testing

I. INTRODUCTION

Then again, mining such models is a test, specifically in light of the fact that one needs to know when two states are proportional, and also how to achieve that state. Here introduce ProCrawl (Process Crawler), a device to mine conduct models from web applications that backing multi-client work processes. ProCrawl incrementally takes in a model by creating system runs and watching the application conduct through the client interface. In our assessment on a few true web applications, ProCrawl removed models that briefly depict the actualized work processes and can be specifically utilized for model-based testing [14].

The omnipresence of web programs and progressions in web advances has brought about web applications turning into a prevailing customer for big business programming. Also, the accessibility of system data transfer capacity empowers applications to be worked by the merchant and gave as administrations to clients. Working applications on the seller side evacuates a portion of the intricacy and expenses of the customary programming discharge and

redesign procedure; while this empowers shorter, more proficient and incessant discharge cycles with a littler number of components, it puts more weight on programming advancement and obliges giving careful consideration to operational viewpoints, nonstop Quality Affirmation (QA) and testing. Having knowledge into the conduct of a product segment through unequivocal models can to a great extent enhance the improvement, QA and support process.

The field of determination mining so as to mine plans to encourage these exercises deliberations from projects and their executions; commonly, models of the program's conduct.

On the off chance that these models are sufficiently exact, they can even be utilized as post-facto details of the project and test designers can apply them in a ceaseless joining environment to check for relapses after code changes. Determination mining has been utilized to infer effectively aphoristic details, for example, capacity and information invariants from projects or limited state machines portraying states and moves for individual classes. For such little scale spaces, it is genuinely simple to approve details, in light of the fact that both system code and project state are available and agreeable to typical thinking and thorough testing. Extricating models on framework level is a great deal more troublesome. Project code and system state, for the case, may not be accessible for examination, as the application may be appropriated over a few layers and locales. As a rule, the main suspicion that can be made is that there is some client interface (UI, for example, a web front end that takes into consideration human communication.)

II. RESEARCH DEVELOPMENT CHALLENGES

The Integrating existing tests with Numerous Web applications accompany existing unit and framework tests. Investigating intends to incorporate and adjust these tests into computerized slithering. Research development challenges - utilizing their information for data provisioning, and their association streams for shockingly better scope. The base of our model is absolutel state-based. Here considering utilizing setting free and connection delicate language structures that would permit to express a great deal more perplexing associations and conditions. Besides Web applications, the strategies connected on nonspecific GUI-driven applications, giving model extraction and resulting model-construct testing in light of an extensive variety of stages and projects.

Overview of solution-

Richer models- At this point, the base of our model is purely state-based. We are thinking about leveraging context-free and context-sensitive grammars, that would allow to express much more complex interactions and dependencies.

Alternative platforms- Besides Web applications, the PROCRAWL techniques could just as well be applied on generic GUI-driven applications, providing model extraction and subsequent model-based testing on a wide range of platforms and programs

III. LITERATURE REVIEW

Conduct models can emotionally supportive network comprehension and acceptance. Model-Based Testing (MBT) takes into account productive test creation when a model depicting the conceivable and the normal

application conduct is accessible, and along these lines for expanded computerization. On the other hand, physically making and keeping up conduct models that are sufficiently exact to be executable by a test computerization apparatus all through the product improvement procedure is costly. Web applications commonly come without express models, which suggests for the most part manual and in this manner less proficient test creation, furthermore eases off comprehension and support[1].

There exist many approaches for reverse engineering various properties of software components, PROCRAWL is a dynamic technique mining behavior models of web applications that generalize upon multiple runs (traces) of the application, which makes it applicable where source code is not accessible or amenable to static analysis. The resulting models capture the interaction of multiple users, as well as the interplay between input data and event sequencing affecting application behavior.

PROCRAWL is related to dynamic techniques mining (extended) finite state machines, experimental techniques that systematically generate program runs to learn program behavior, techniques extracting models from GUI applications, and web crawling approaches[2].

There are various approaches that mine FSMs capturing program behavior as sequencing of events, many of them building upon the k-tails algorithm. However, program behavior usually depends not only on the sequencing of events, but also on the provided input data, which led to approaches combining FSM inference with data rule inference[3].

MINER mines parametric specifications capturing the multi-object behavior of Java classes by preprocessing parametric execution traces and using an extension of k-tails to extract an FSM with data parameters. Pradel et al[4]. Also mine parametric multi-object specifications (API usage protocols) from Java classes. While multi-object specifications capture method calls performed on multiple objects, PROCRAWL captures UI interactions performed by multiple actors. One pioneering work was developed by Lorenzo et al. While nondeterministic models may be acceptable for the program comprehension, they are usually too imprecise for test case generation, the transition guard learning approach implemented in PROCRAWL is similar to the approach of Walkinshaw et al[5].

The quality of models extracted by dynamic techniques closely depends on the choice of program runs (traces) these techniques generalize upon. Experimental approaches such as PROCRAWL tackle this problem by systematically generating runs to explore program behavior; this allows PROCRAWL to verify its hypotheses and explicitly control the exploration scope, which would otherwise be implicitly induced by a given set of traces. Such techniques have successfully been implemented for mining behavior models of Java classes[6].

The underlying problem PROCRAWL has to solve is an instance of online exploration of a directed multi-graph by repeatedly selecting an outgoing edge from the current vertex and traversing it, which is a fundamental problem in robotics and has been extensively studied for strongly connected graph. However, ABMs are usually not strongly connected, i.e. PROCRAWL might need to reset the SUT to the initial state to continue graph exploration[7].

PROCRAWL mines program behavior by observing changes on the application's web UI, which is related to GUI Ripping developed by Atif Memon et al. GUITAR reverse engineers Event-Flow Graphs (EFGs,) of Java desktop,

the web and Android applications, which are used for model-based GUI testing. While in ABMs extracted by PROCRAWL user actions are modeled as transitions between nodes that represent abstract GUI states of multiple users, in EFGs they are modeled as nodes with transitions representing the event flow[8].

To explore a web application, PROCRAWL applies techniques similar to web crawling. Gives an overview on the state of the art. A prominent representative and actively developed crawler for AJAX applications is CRAWLJAX, which automatically creates a State-Flow Graph (SFG) of the dynamic DOM states and the event-based transitions between them. SFGs depict the various navigational paths and UI states within an AJAX application. Although ABMs look similar to SFGs, the underlying abstraction in SFGs is much closer to the SUT's UI. Limiting the state abstraction scope to a single DOM tree of a single user, limits the number of actions that can be detected by the crawler and often leads to a nondeterministic FSM, prohibiting effective model-based testing. In ABMs, a node represents an abstraction over multiple DOM trees (views) of multiple users and transitions refer to generated scripts encapsulating sequences of UI commands. Furthermore nondeterminism is effectively eliminated by learning transition guards over the input data[9][10].

In our evaluation apply the behavior models inferred by PROCRAWL for state-based web application testing, similar to and However, due to the wider exploration scope and higher level of abstraction supporting multiple users and views, the test cases generated from PROCRAWL ABMs are more than pure UI tests and suitable for testing workflows. reports the state of the art in web testing[5][11].

Presents an approach for reverse engineering business processes exposed in web applications by inferring an FSM from execution traces and transforming it to Business Process(BPMN) and Recovering and Reducing Business Processes REBPMN [12].Specification mining has been used to successfully derive axiomatic specifications such as function and data invariants from programs [13].

IV. MINING WORKFLOW MODELS

Technically, web applications have a client and a server part. The client code runs in a web browser and is usually implemented in HTML, CSS and JavaScript. The browser parses the HTML code and creates a Document Object Model (DOM) tree with the content and structure of the page, representing HTML elements such as hyperlinks, buttons and text as nodes, which are rendered using the style information defined in HTML attributes or CSS. Content, structure and style can be dynamically modified through client-side JavaScript event handlers that are called on DOM events such as onclick or server-side state changes that are propagated to the client. To improve the user experience and reduce the amount of data to be transferred from the server to the client in order to update a page, modern web applications usually exchange data with the server asynchronously using JavaScript and the XMLHttpRequest object (AJAX), i.e. the UI can be updated without a full page reload.

A. Workflow Exploration

As many other approaches to behaviour model inference, PROCRAWL applies dynamic analysis[17], i.e. generalizes upon observed application executions (traces).

ProCrawl applies a black-box approach that is it mines application behavior without accessing the sources code of the System Under Test(SUT).It determines the configured set of UI views(technical DOM trees)[15],

To fully automatic configurable tool to my workflow models from web applications. It includes implements an iterative approach of executing actions through the UI, observing changes to the UI, and enhancing the model. Design an approach to increases model accuracy by inferring transition guard conditions from the input data

- 1.To Study and Integrating existing tests and implementation GUI-driven system using proCrawl model with context-free grammar and context sensitive grammar.
- 2.To Study context-free and context-sensitive grammars, that would allow to express much more complex interactions and dependencies Besides
- 3.Design an Web applications GUI-driven applications providing model extraction and subsequent model-based testing on a wide range of platforms and programs.
- 4.To integrate and adapt these tests into automated crawling, using their data for input provisioning, and their interaction flows for even better coverage.

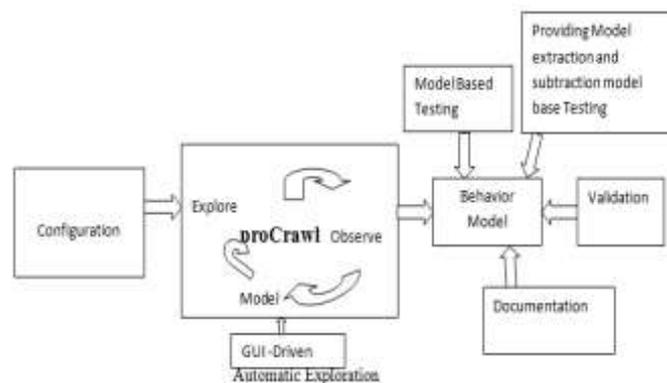


Fig 1. ProCrawl overview

B. Process Crawler Model(PROCRAWL workflow model)

in a shared data store that is manipulated when executing the processes. The data is collaboratively edited through browser-based clients. Depending on the user role these clients may have varying functionality that is typically separated over multiple UI views.

Process Crawler model make the following contributions:

- 1) To the best of our knowledge, PROCRAWL is the first approach automatically learning multi-user workflow models that are complete enough to be executable by a UI automation tool.
- 2) This model present a machine learning approach for inferring decision rules from data input that serve as transition guard conditions for resolving nondeterminism .
- 3) In our evaluation on several real-world web applications, the workflow models mined by PROCRAWL concisely capture the essentials of the underlying process.

4) The resulting workflow models can be directly used as input to model-based testing ,model based-testing show how automated tests following the extracted models detect structural and logical changes in new versions of the applications.

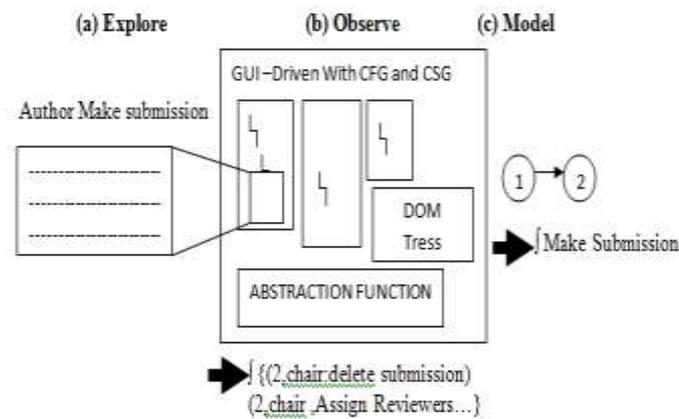


Fig 2. First iteration. PROCRAWL

V. MODELS

To integrate our system described following four models

A. Configuration:

Configured system with the identified actors, views, start action, a test fixture, and the click depth and started system and without transition guard learning;

The participating actors A: in the OPENCONF example an author, a chair and a reviewer, is including their login credentials[16].

- a start action a_0 which is executed by one of the actors (Author: Make Submission in OPENCONF)
- an exploration scope $\forall \alpha$ for each actor $\alpha \in A$, i.e. the views of the System Under Test (SUT) to be checked for changes after executing an action; for each view a sequence of UI commands, such as $h(\text{open, openconf.com/chair}, (\text{click, List Submissions}))$. In OPENCONF this includes the URLs to the reviewer assignments, as well as the submission, review and score list for the chair, and the submissions list for the reviewer.
- a test fixture⁵ to set the SUT to the initial state (test context), which may be necessary to complete workflow exploration after a nonreversible action. Optionally for each actor the type and number of browsers, maximum click depth and DOM elements to be considered or ignored, as well as the input elements for which different values are generated can be specified. Furthermore, the state abstraction can be changed by providing DOM filters and selectors, and test oracles can check the SUT for errors during behavior exploration.

B. Automatic Exploration.

It executes an action with a configured actor, infers the state of the SUT by applying an abstraction function over the DOM trees (views) extracted with the actors, and update the behavior, model.

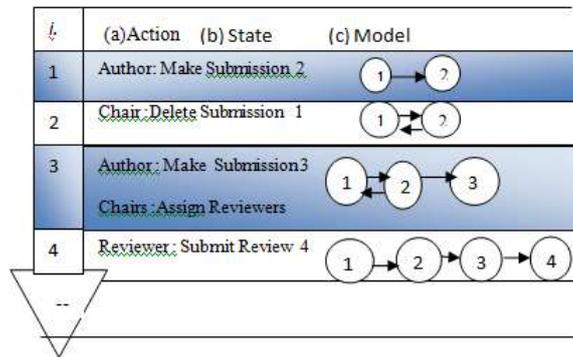


Fig 3. PRO-CRAWL iteratively (a) executes an action with a configured actor (b) infers the state of SUT, and (c) update the behavior model.

Based on this configuration, the system automatically explores the behavior of the web application. each iteration consists of the following steps

1. Run action
2. State abstraction
3. Model.

C. Application Behavior model

PROCRAWL captures application behavior in form of an extended finite state machine si.e. a finite state machine with additional data state variables, transition guards which must hold with respect to the data state variables to enable a transition, and update functions changing the state variables when a transition is executed.

By generating tests from the ABM of the baseline application and executing these tests on the delta build, we are able to detect

- Logical Changes (LC) to the underlying workflow, e.g. removing workflow steps or introducing additional mandatory steps.
- Structural Changes (SC) to the web UI that cause the generated scripts to fail, e.g. removing, replacing or changing attributes of UI elements that are used in the scripts to locate these elements.

D. Model Based Testing and validation

With the aim of increasing test automation, Model-Based Testing (MBT) has expanded the automation of software testing towards the test design phase: behavior models can be used to derive a test suite that is when executed through a test automation framework. The Industrial application has positive effects of MBT on the development productivity .However, wide-scale adoption suffers from the absence of explicit models that are complete enough to be executable by test frameworks.

V.IMPLEMENTATION

PROCRAWL is implemented in Java, using SELENIUM8 for web browser automation and provides a plug-in architecture for writing extensions, which is used for ABM and script serialization, as well as transition guard learning. For each configured actor one or multiple web browsers are started, on which the UI commands are

executed. Each time PROCRAWL detects a new (nonreflexive) transition, the executed UI commands are exported as a SELENIUM script and the behaviour.

Implementing following Algorithms

- Algorithm 1-shows the initialization of the exploration procedure for mining the behavior model. First, a global driver pool is created that is used to interact with the SUT's web UI (Line 3). Then, the initial state s_0 of the SUT is determined (Line 4) and an ABM containing s_0 is created (Lines 2&5). As described before, PROCRAWL keeps a set of pending actions $A(s)$ for each state s , which are executed to explore the SUT's behavior; $A(s_0)$ is initialized with the start action defined in the configuration (Line 6). The plug-ins are initialized (Line 7) and the exploration procedure is called on s_0 (Line 8). After the exploration finished, the plug-ins are notified (Line 9).

Input: config

```
1 global A ← new HashMultimap();
2 global ABMk ← new AppBehaviorModel();
3 global driver ← INITDRIVERPOOL(config);
4 s0 ← DETERMINESTATE();
5 ABMk.initialState ← s0;
6 A(s0) ← {config.startAction};
7 PLUGINS.EXPLORATIONSTARTED(ABMk, config);
8 EXPLORE(s0);
9 PLUGINS.EXPLORATIONFINISHED();
```

- Algorithm 2. a state is defined by the set of elements extracted from the DOM trees of multiple actor/view relations. For each actor α , the DOM trees of the views defined in the actor's exploration scope are retrieved in parallel (Line 3). After removing certain nodes such as invisible HTML elements from the DOM tree (Line 4), elements matching the selection criterion are added to the state (Line 6). Two states s ; s_0 are considered equal, if they have the same set of distinct elements.

Data: config

Output: current state s of the SUT

```
1 s ← {};
2 foreach ( $\alpha, \zeta v$ ) ∈ config.exploration_scope do
3 DOM ← driver.GETDOM( $\alpha, \zeta v$ );
4 fDOM ← FILTER( $\alpha$ , DOM);
5 foreach e ∈ SELECT( $\alpha, fDOM$ ) do
6 s ← s ∪ f( $\alpha, \zeta v, e$ );
7 end
8 end
9 return s;
```

VI. RESULTS AND DISCUSSION ON MODULE

PROCRAWL is a fully automatic, though configurable, tool to mine workflow models from web applications. It implements an iterative approach of executing actions through the UI, observing changes to the UI, and enhancing the model. Its state abstraction is specific enough to capture essential workflow steps, yet sufficiently generic to be applicable to a diverse range of web applications. Through active learning, PROCRAWL increases model accuracy by inferring transition guard conditions from the input data. As our evaluation on several real-world web applications shows, the models mined by PROCRAWL are adequate in size, accurate, cover all to almost all workflow-relevant actions, and are a suitable input to model-based test generation.

INPUT VARIABLES (attributes)		TARGET STATE		
scores_benefit	subs_subaction	scores_submit	scores_subaction	Class
INFO GAIN 0.311	0	0.311	1.0	
1 all submissions	Withdraw Submissions	Go	Reject	05
2 all submissions	Withdraw Submissions	Go	Accept	
3 default	Withdraw Submissions	default	default	
4 all submissions	Withdraw Submissions	Go	Pending	02

TABLE 1. Data for learning the guard conditions of the nondeterministic Restore Submissions transition.

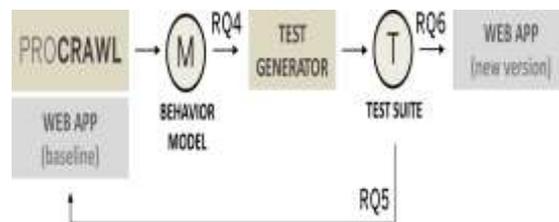


Fig. 5 Test generation.

Process crawler extract a behavior model from the baseline application, use the model as input for a test generator (RQ4), run the generated test cases on the baseline application to ensure they succeed (RQ5) and on the new version of the web application to detect changes (RQ6).

RQ4 Are the extracted ABMs and scripts usable as input to **model-based testing**?

RQ5 What is the level of **over-approximation** of the extracted ABMs? Do they correctly model the target workflow? How many paths in the ABM are infeasible?

RQ6 Are the test cases generated from the ABMs useful for **system regression testing**?

VII. CONCLUSION

It implements an iterative approach of executing actions through the UI, observing changes to the UI, and enhancing the model. Its state abstraction is specific enough to capture essential workflow steps, yet sufficiently generic to be applicable to a diverse range of web applications.

PROCRAWL is a fully automatic, though configurable, tool to mine workflow models from web applications. Its increasing accuracy of web application.

REFERENCES

- [1] M. Utting and B. Legeard. Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007
- [2] M. P. Robillard, E. Bodden, D. Kawrykow, M. Mezini, and T. Ratchford. Automated API property inference techniques. *IEEE Trans. Softw. Eng.*, 39(5):613–637, May 2013.
- [3] N. Walkinshaw, B. Lambeau, C. Damas, K. Bogdanov, and P. Dupont. Stamina: A competition to encourage the development and assessment of software model inference techniques. *Empirical Softw. Eng.*, 18(4):791–824, Aug. 2013..
- [4] C. Lee, F. Chen, and G. Ros, u. Mining parametric specifications. In *ICSE*, pages 591–600, New York, USA, 2011.
- [5] M. Pradel, C. Jaspan, J. Aldrich, and T. R. Gross. Statically checking api protocol conformance with mined multi-object specifications. In *ICSE*, pages 925–935, Piscataway, NJ, USA, 2012. IEEE Press.
- [6] M. Schur, A. Roth, and A. Zeller. Procrawl: mining test models from multi-user web applications. In *International Symposium on Software Testing and Analysis, ISSTA '14*, San Jose, CA, USA July 21 - 26, 2014, pages 413–416, 2014.
- [7] V. Dallmeier, N. Knopp, C. Mallon, S. Hack, and A. Zeller. Generating test cases for specification mining. In *ISSTA*, pages 85–96, New York, USA, 2010. ACM
- [8] B. N. Nguyen, B. Robbins, I. Banerjee, and A. Memon. GUITAR: an innovative tool for automated testing of GUI-driven software. *Automated Software Engineering*, pages 1–41, 2013
- [9] W. Grieskamp, N. Kicillof, K. Stobie, and V. Braberman. Modelbased quality assurance of protocol documentation: tools and methodology. *Software Testing, Verification & Reliability*, 21(1):55–71, Mar. 2011.
- [10] V. Garousi, A. Mesbah, A. Betin-Can, and S. Mirshokraie. A Systematic Mapping Study of Web Application Testing. *Information and Software Technology*, 55(8):1374–1396, Aug. 2015
- [11] M. Schur, A. Roth, and A. Zeller. Mining behavior models from enterprise web applications. In *ESEC/SIGSOFT FSE*, pages 422–432, 2013.
- [12] A. Tomasi, A. Marchetto, C. D. Francescomarino, and A. Susi. reBPMN : Recovering and Reducing Business Processes. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 666–669. IEEE, 2012
- [13] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin. Dynamically discovering likely program invariants to support

program evolution. IEEE Trans. Softw. Eng., 27(2):99–123, 2001.

[14]. M. Utting and B. Legeard. Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[15]. Q. Xie and A. Memon. Using a pilot study to derive a GUI model for automated testing. ACM Transactions on Software Engineering and Methodology, 18(2):1–35, 2008

[16]. <https://www.openconf.com/>

[17]. A. Zeller. Program analysis: A hierarchy. In ICSE Workshop on Dynamic Analysis (WODA 2003), pages 6–9, 2003.



First A. Mrs. Nayan Mulla is a PHP developer at Purestudy Software Services Pvt.Ltd and a M.E student at Shivaji University. She received the B.E. degree in Information Technology in 2013 from Shivaji University. His research concerns the analysis and testing of enterprise web application.



SECOND B. Mr. Takmare Sachin Balawant is working as Associate Professor and Head of Department in computer Science and Engineering Department of Bharati Vidyapeeth's college of Engineering, Kolhapur with teaching experience of about 10 year. He has published about three International Papers and five National Papers.