A Survey on Agile Software Testing Mechanism with Directed Acyclic Graph (DAG) Based Model in Various Platform

1. Alamelumangaiyarkarasi V, 2. Dr Srinath M.V., 3. Dr Omar A. Al Heyasat

ABSTRACT

Background: A Survey on Agile Software Testing Mechanism with Directed Acyclic Graph (DAG) Based Model in Various Platform. Objective: Software testing ensures the quality of the software product. Agile testing is an iterative development methodology, where communication with the customer is more transparent and accurate. Agile relies on technical excellence, simple designs, and create business value by delivering software to the client in a shorter time period. The test cases for agile testing are generated and executed sequentially. Agile includes different methods such as scrum, xp, crystal, etc. In a DAG, the nodes represent the sequential computation tasks and edges indicate data communication. DAG schedules its task in a fully distributed and dynamic way. DAG user mainly concentrates on expressing the algorithm and defines how the tasks should be distributed over the resources. The existing mechanisms concentrate on how to generate the test cases and to execute it consecutively, which may lead to faulty occurrence. This survey paper discusses, Agile software testing mechanism with a DAG based model in various platforms. Results: The results of the survey are shown in table 1. Various approaches on Agile software testing with DAG based model are discussed. A Hybrid Engineering approach for mobile development was used for iterative-incremental development based on the requirements. An MVC approach, analyses of a security problem which includes security functions and scenario attacks. The NORMAP methodology investigates the feasibility of identifying, linking and modeling agile use cases. A use case point estimation model offers an effective effort estimation in testing. A JIC and JIT approach, DAGue framework, CA-DAG model. The existing mechanism only focus on how to generate the test cases for the agile testing. It does not have a proper way to produce the execution of test case in a sequential manner. The result of the survey shows the application of agile testing in various platforms. To overcome the above problem, we focus on the agile software testing with DAG based model. Based on the survey, the authors proposes an agile testing platform with an optimized fault tolerant mechanism to provide the quality software.

INTRODUCTION

Software Testing can be described as the process of validating and verifying the product. Testing also ensures whether the product meets the business and technical requirement. Testing is done manually or by using automated tools. Agile testing is the software testing practice, which involves all members of the cross functional agile team to ensure the business values desired by customer. It includes various methods:

- Scrum
- XP (Xtreme Programming)
- Feature Driven Development (FDD)
- Crystal Clear Methodology (CC)
- Dynamic System Development Methods (DSDM)
- Adaptive Software Development (ASD)

Corresponding Author: Alamelumangaiyarkarasi V, STET Womens College, Department of Computer Applications, Professor, Box 3030, Mannargudi, India.
Tel: 91-4367-255423; E-mail: vsalamu@gmail.com,
Scrum is an agile methodology, it has been developed to manage the process for system development. This approach does not concentrate on software development techniques for the execution state. It basically guides the management that how the team members can perform in order to achieve flexibility where the requirement changes frequently. Scrum team is a cross functional team where each and every one participate in developing the product. It contains a series of iteration called sprints, and each sprint consists of backlog items, where backlog item corresponds to the amount of work to be done.

Xtreme programming is a cluster of different processes, rules and ideas which are inherited from previous methodologies. The main properties of XP are short iterations with speedy feedback, regular participation of customers, regular testing and integration activities. In XP, index cards are used to collect requirements. The user writes the requirement in the form of stories, design the system. Feature Driven Development is a client-centric, architecture-centric software process. It is a software management method which involves planning, up-front modeling and design. Crystal clear methodology is a smallest series methodology in the software development which involves small teams. It is prioritized for project safety, effectiveness and endurable. Crystal family is marked with color, which indicates the heaviest of the methodology (darker the color heavier the methodology). Crystal methodology does not restrict any development practices, tool or work product rather it allows adopting of XP and scrum practices.

DSDM is an iterative and incremental approach that embraces the principle of agile development including customer engagement. Adaptive software development is a design rule for the creation of software products which concentrate on rapid creation and evolution of software. DAG is a collection of tasks arranged in a sequential order which are subjected to certain constraints. In DAG, scheduling is done and the tasks are executed according to the schedules. DAG is independent of problem size, where each process runs their own instance of the scheduler.

This paper is organized as follows, Section 2 illustrates about agile software testing mechanism in various platforms. Section 3 describes about result and discussion. Section 4 explains about conclusion and future enhancement.

**Agile Software Testing Mechanism With Dag Based Model:**

**A) Hybrid Engineering Approach:**

An Agile risk based methodology based on the Hybrid engineering approach (Rahimian, V.R.R., 2008) was used for iterative-incremental development based on a predefined set of requirements and the knowledge gained from existing models/designs. This approach has been formatted as a top-down iterative, incremental process consisting of following tasks:

- **Prioritization of the requirements** - performed at the beginning of the process and repeated at the conclusion of each iteration. At the beginning of the process, abstraction is at a higher level and the scopes confines the entire lifecycle. Therefore, the requirement with lifecycle impacts were given priority.

- **Instantiation** - an already available process meta-model was useful when designing higher level aspects of methodology.

- **Artifact oriented** - formulates a flawless chain of artifacts and building the process around it.

- **Composition** - using the process patterns which were available in the library.

- **Integration** - integrating ideas, features and techniques from existing methodology.

In the first iteration, the practice was detailed through the generic rules and development. Analysis was split into preliminary analysis and detailed analysis to reduce errors. The second iteration focuses on the borrowing activities from the New Product Development (NPD) and design process of the second iteration involves reusability of the NPD process. In the third iteration, the process’s development engine was enriched by integrating ideas from ASD. The requirement based hybrid approach ensures that the requirement were properly addressed and validated. This approach was easily fitted for the production of mobile software system based on agile methodology.

**B) Agile security testing based on Model-View-Controller (MVC) approach:**

This approach provides a unified point of view for analyzing security problems, which include security function and possible scenario attacks (Munetoh, S.Y.N., 2013). In this approach, a tool called RailroadMap was developed to extract the behavior model from the code. The MVC framework comprises of two models:

1. **Navigation model**
2. **Abuse model**

Navigation model simply represents the conduct described by application source code and security characteristics were verified. Abuse model extends the feature of the navigation model by adding security related behavior which is provided by application framework or third party modules, the details were inevitable at the main level code and programmer. In the second model, test cases were generated to verify the individual security features. The MVC web application framework contains the operational flow as follows:

- Analyze the application code and generate a navigation model.
Examine the security design and requirements with the navigation model.
Generate an abuse model by including the behaviors of the security features.
Finally, generate the minimum number of test cases.

In MVC model, the method in the controller class and template file were assigned to independent states. This designation not only keeps the symmetry between the code and model, but also well-suited for representing the Web application behavior. This model was applicable for providing secure web based applications.

Network based systems were becoming significant in many business paradigms. For some businesses, the web interface exhibits the exclusive level of touch between the consumer and commercial enterprise. This arrangement is characterized as any projects which is evolved and deployed through the World Wide Web (WWW). Http-unit is a unit test schema which provides the power to bypass the client GUI by echoing relevant portions of web browser (Tappenden, T., et al, 2005). Http-unit is capable of examining the returned pages in various ways. It also provides modification, where the developer can take advantage in modifying the judgment. This approach elaborates about the performance of security testing using Http-unit.

C) Nonfunctional Requirements (NFR) modeling for agile process (NORMAP) methodology:

The NORMAP methodology investigate the feasibility of identifying, linking, and modeling Agile Loose Cases (ALCs) with Agile Choose Cases (ACCs) and Agile Use Cases (AUCs) (F.W.M., 2012). The NORMAP methodology was validated through NORMATIC, which is a Java based tool (Farid, W.M.M.F.J., 2012). AUCs were newly suggested hybrid use cases and agile stories. The required quality, and the project management metric was used to calculate a risk driven requirement implementation scheme by NORMAP methodology. Agile groups can substantially benefit from the NORMAP methodology by engaging a systematic and risk-driven lightweight engineering process to visually model and plan NFRs in agile environments.

D) Use case point estimation model:

This model deals with reducing the gap between estimated effort and real time implement effort (P.A.W.M.M., 2013). A use case defines the system’s behavior under different condition as it responds to one of the stake holder’s requests, called primary actor. The primary actor initiates an interaction with the system to achieve a goal. This model gathers all the different use cases/scenarios and convert them into numbers and drive. The first step of estimation is to detect size and the next step is to do effort estimation. This model offers an effective effort estimation in testing.

A Model Driven Architecture (MDA) concentrates on the role of models during product development. This is highly effective and prone to less errors (Rumpe, B., 2006). This holds in different strategies, code for MDA is retrieved from a pattern and use cases are generated from an analysis or behavior models. The model based software development suggests the use of the model as the primary artifact for requirements, documentation, code generation and development. To guarantee the caliber of an evolving system intensive set of trial cases are must.

E) Just-in-Case (JIC) and Just-in-Time (JIT) approach:

This approach ensures better interaction, collaboration and an optimal exploitation in the software development effort (Dessai, K.K.M., 2012). JIC and JIT construct resources for making and retrieving the results from social media. In JIC approach, apparent development tasks, requirement gathering, analysis tasks was covered simultaneously. This is possible with the collaboration of analyst and developer at the initial level. The concept here is to originate and keep ready, significant amount of deve

F) Agile interaction in GUI based on ActiveStory Enhanced and LEET tools:

In this approach the tests were recorded from the prototype and replayed on the actual GUI as it was carried out [10]. ActiveStory Enhanced tool allows engineers to create low fidelity prototypes. This paradigm consists of images of different states of user interface and “hot zones”. These prototypes were cheap to make and modify using ActiveStory Enhanced tools. LEET is a capture tool, which was developed based on User Interface Automation Framework. This fabric allows for keyword based testing, since the only attribute of a widget is used to identify automation id. This is capable of simulating the test and replay on an actual GUI.
G) Agile methods in embedded systems:
This paper provides the guidance to support the adoption of agile methods in embedded systems (Srinivasan, J., R.L.K. Dobrin, 2009). Agile approaches are suited for environments in which requirement is evolving. There are two areas need to be addressed to adopt agile methods.

1. Technical issues
2. Organizational issues.
   - Requirement, development and management need to be patched up to support modifiability, maintainability and reliability. Agile methods require organizational support infrastructure ranging from new legal documents and methods. Different software processes had been advised to organize software development activities (Albuquerque, C.O., P.O.N.E.Y. Antonino, 2012). There are industry tested processes aimed at more effective project management, such as RUP and V-model.
   - A MARMOT approach (Bunse, C., H.G.P.C. Gross, 2007) expedites reuse in embedded system development, which is a component based development schema for information systems. A MARMOT modeling principles have two sets of diagrams, one along the specification level and another on the realization level.

H) Service oriented architecture for agile testing:
This architecture provides an interoperable solution where the user can start out with a test plan, deploy a scalable, data monitoring and analysis capability and follow the process from NPI through production (Weir, A., M.A.A. Kulak, 2013). The entire procedure is time consuming and resource intensive. This architecture is scalable and can be deployed in single-site or global applications. This approach (Kroghdahl, P., G.S.C. Luef, 2005) helps the system architects to implement a Core architecture before the start of any SOA-based development used by determining Core Business Process (CBP)

I) Test case generation algorithm using DAG:
In this algorithm, DAG is a directed graph with no directed circuits with a vertex v, and there is no nonempty directed path (Tuglular, T., et al, 2009). Event sequence graphic is an event based model, where input and events are combined and assigned to the vertices of an event transition diagram. A Complete Event Sequence (CES) simply transmits through ESG, where it starts at the entry node and ends at the exit node of the ESG. The testing approach consists of five stages.

- Generating test cases from the firewall rules
- Constructing network test packets from generated test cases
- Sending constructed packets to FUT
- Capturing the packets that pass through the FUT
- Comparing sent and captured packets to find the results.

This approach concentrates on generation of test cases. Firewall testing agenda is automated using test case generation algorithm and Directed Acyclic Graph (DAG).

J) DAG framework for parallel computing:
This framework schedules tasks in a fully dispersed and active environment, which uses all cores of each node (Bosilca, G., 2012). It comprises of runtime engine and a set of tools to build, analyze and pre-compile a representation of DAG. The internal representation of DAG used by DAGue is called as JDF. The JDF representation of the DAG was pre-compiled as C-code and connected in the final binary program. The DAGue library incorporates the runtime environment, comprises of distributed multi-level dynamic scheduler. It is an asynchronous communication engine and a data dependency engine.

The DAGue engine is responsible for moving data from single CPU to another when necessary, tasks are enabled only when all the data are marked as IN. Dependencies of JDF are marked with a modifier, which is a type of qualifier. It instructs the communication engine, how to transmit the data from a remote location to another. In this framework DAGue engine performance has been investigated.

K) Communication Aware model (CA-DAG):
This model defines the communication awareness of cloud applications, which allows separate resource allocations and assigning processors to handle the network resources for information transmissions (Kliazovich, D., 2013). The program is implemented by a directed acyclic graph \( A=(V, E, \omega, \phi) \). The set of vertices \( V=(V_m, V_{com}) \), which possesses two non-overlapping over subsets \( V_m, V_{com} \). The set \( V_m \) belongs to V which indicates the computing tasks and the set \( V_{com} \) belongs to V represent the communication tasks. The DAGs which uses edges to design the communication processes are not capable of modeling certain communication types. CA-DAG model is possible only when an extra link is available.

The resources are implemented with a shared network link, which interconnects the databases and resources. The network topology allows only one communication at a time, while other nodes constrain their
transmissions until the link becomes free. In this approach DAG along with computational vertices are used to indicate the communication.

I) Parallel task model:
This model addresses a deterministic parallel task where each task is assigned as a DAG with different nodes having different requirements (Saifullah, A., et al, 2013). A task decomposition is designed, which splits the DAG into sequential tasks. Following the parallel tasks, decomposition is scheduled using preemptive global EDF. Ultimately, the disintegration of a resource augmentation for non-preemptive global EDF scheduling is shown.

In the general task model, tasks are represented by DAGs where threads can have arbitrary execution requirements. Later, preemptive and node level non-preemptive scheduling are addressed for these DAGs. The decomposition is designed in multi real-time scheduling for unit code DAG. It is applied to a general DAG where every node will be further split into a smaller node. All subtasks of a segment are synchronized at its death, where there is no path of assuring non-preemption of a thread. In scheduling, mixed parallel application with an advanced reservation involves CPA algorithm (Aida, K.C.H., 2009) which comprises two phases. In first phase an algorithm decides how many processors should be assigned to each project. In the second stage, tasks are scheduled in time and place. The theorems are compared over a wide range of applications and scenarios.

M) Sporadic DAG Task model:
This model is a well-known model for projecting the real time system based on a finite number of independent tasks each of which may propagate an unbounded sequence of jobs (Bonifaci, V., 2013). Here, real time workloads can be planned as a cluster of independent sporadic DAGs and are compiled on a platform, which holds n identical CPU. The program is fully preemptive and it permits global interprocessor migration, where each operation may get accomplished on at most one CPU at each moment of time.

The sporadic task model is a model of repetitive process in hard real time systems (Bonifaci, V., A. Marchetti-Spaccamela, 2012). A sporadic task \( S = (W, R, M) \) where \( W \) is worst case execution time, \( R \) is a relative deadline and \( M \) is minimum inter arrival separation. This model clearly explains about the feasibility study of sporadic DAG.

N) Continuous Integration (CI) Model:
In this model, members of a team integrate and shape to minimize the duration and the product is delivered at any minute. It was a development pattern that requires the developer to incorporate code into a shared repository (Stolberg, S., 2009). This model helps to ascertain out the bugs promptly. The CI server supervises the repository and checks out the modifications when they come. The server builds the system and runs unit and integration testing, then it delivers the deployable artifacts for testing. The server allocates a build label to the explanation of the code it just built. If the test fails, server alerts the team and the team fixes the issue. CI allows for greater collaboration between development and delivery.

O) Mobile-D approach:
This process is applied by a team of at most ten developers, turning towards a product delivery. It comprises five stages, namely, Explore, Initialize, production, stabilize and system (Abrahamsson, P., 2004). Each of these phase contains stages, tasks and exercises. The development team generates a plan and establish the project characteristics. Undertakings, in this phase include customer establishment, planning of project, and collection of requirement and process validation. In the initialization phase, the development team and stakeholders develop a key source for product activities. Production contains implementation activities and final two stage deals about product formalization.

P) Pachinko Allocation Model (PAM):
This prototype uses a DAG to capture arbitrary, nested, and possible parse correlation between topics and also produces flexible alternatives (Li, W.M.A., 2006). The model structure consist of an arbitrary DAG, in which each leaf node is connected with a word in vocabulary and each non leaf node corresponds to a topic. In PAM each interior node’s distribution over its children and could be parameterized arbitrarily. PAM connects the words in S and topic in C within an arbitrary DAG, where topic nodes occupy the interior level and the leaves are the words. Each topic \( C_i \) associated with a Dirichlet distribution \( g_i(\alpha_i) \), where \( \alpha_i \) is a vector with the same dimension as the number of children in \( C_i \). In general, \( g_i \) is not confined to the Dirichlet it could be any distribution over discrete children, such as logistic normal.
Q) Recursive method for DAG:

In this method an algorithm is proposed for structure learning of DAG. An algorithm consists of series of operation of a binary tree, where the top node of the tree contains the full set of variables. The leaves of the trees are considered as a subset of variables which cannot be disintegrated (Xie, X.G.Z., 2008). An algorithm involves two procedures: (a) Top-down steps for decomposing the full set of variables into subsections as small as possible, and (b) Bottom-up step is to integrate the local frame into global frame.

RESULTS AND DISCUSSION

The results of the survey are shown in table 1. Various approaches on Agile software testing with DAG based model are discussed. A Hybrid Engineering approach for mobile development was used for iterative-incremental development based on the requirements. An MVC approach, analyses of a security problem which includes security functions and scenario attacks. The NORMAP methodology investigates the feasibility of identifying, linking and modeling agile use cases. A use case point estimation model offers an effective effort estimation in testing. A JIC and JIT approach used for retrieving results from social media.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Year and Reference</th>
<th>Technique</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abrahmsson .P et al.</td>
<td>2004 [1]</td>
<td>Mobile-D approach</td>
<td>It is a viable discipline which ensures the quality of the product in mobile application development.</td>
</tr>
<tr>
<td>Anda .K and Casanova .H</td>
<td>2009 [2]</td>
<td>Critical Path and Area (CPA)</td>
<td>This schedule works well in determining the tasks when the reservation schedule is not tight.</td>
</tr>
<tr>
<td>Bosilca .G et al.</td>
<td>2012 [6]</td>
<td>DAGue framework</td>
<td>This framework is responsible for moving data from one processor to another where tasks are enabled.</td>
</tr>
<tr>
<td>Bunse .C et al.</td>
<td>2007 [7]</td>
<td>MARMOT approach</td>
<td>It implies reuse in embedded systems, which is a schema for information systems.</td>
</tr>
<tr>
<td>Furdi .W.M and Mitropoulos .F.J</td>
<td>2012 [9]</td>
<td>NORMATIC tool</td>
<td>These metrics were used to calculate a risk driven requirement implementation sequence.</td>
</tr>
<tr>
<td>Hellmann T.D</td>
<td>2010 [10]</td>
<td>ActiveStory Enhanced and LEET tools</td>
<td>It allows keyword based testing and capable of simulating the test.</td>
</tr>
<tr>
<td>Munetoh .S and Yoshioka .N</td>
<td>2013 [16]</td>
<td>Model View Controller (MVC) approach</td>
<td>It provides a platform for the agile based testing using a tool called RailRoadMap.</td>
</tr>
<tr>
<td>Rahimian .V and Ramsin .R</td>
<td>2008 [17]</td>
<td>Hybrid Engineering approach</td>
<td>This approach was easily equipped for the mobile software system based on agile.</td>
</tr>
<tr>
<td>Safiullah .A et al.</td>
<td>2012 [19]</td>
<td>Parallel task model</td>
<td>It permits multi real time scheduling using DAG.</td>
</tr>
<tr>
<td>Stolberg .S</td>
<td>2009 [21]</td>
<td>Continuous Integration (CI) model</td>
<td>It admits greater collaboration, between development and delivery</td>
</tr>
<tr>
<td>Tappenden .A and Beatty .P</td>
<td>2005 [22]</td>
<td>Agile security testing via HTTP units</td>
<td>Provides the application with the right functionality and the right level of security.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>--------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Conclusion:**

The paper comprises the survey results in agile software testing based on a DAG model in various platforms. The survey characterizes about hybrid engineering approach, MVC model, JIC and JIT approach, DAgue framework, CA-DAG model. The existing mechanism only focus on how to generate the test cases for the agile testing. It does not have a proper way to produce the execution of test case in a sequential manner.

The result of the survey shows the application of agile testing in various platforms. To overcome the above problem, we focus on the agile software testing with DAG based model. Based on the survey, the authors proposes an agile testing platform with an optimized fault tolerant mechanism to provide quality software.

**REFERENCES**


