1. INTRODUCTION
Incomplete requirements, poor user’s input and stakeholder’s conflicts are commonly known to be the major root causes to impaired software projects. A software project is considered as an ultimate failure when the intended users refuse to use the system. One of the reasons for their refusal to use the system is that the system does not meet their needs and expectations [1,2,3]. In this case, conducting the acceptance test at the earliest stage of software development life cycle (SDLC) is crucial in order to ensure the development of the right requirements that meet the needs of the users. Conducting the acceptance test will also help to avoid late detection of errors. This is because the detection and correction of errors at the later stage of a software development is usually costly and time consuming.

Acceptance test is an important validation process that can ensure the newly developed software meets the needs and expectations of the client-stakeholders requirements. The term “acceptance test” is often confused with the “user acceptance test (UAT)”, which is generally conducted at the last phase of SDLC for the purpose of a final validation of the newly developed software. In contrast, acceptance test or business-facing test is done at the initial phase of SDLC. Commonly used to measure the business value of a requirement, it is a validation (testing) procedure conducted at a higher level between the business logic and the user interface or directly with the user interface [4,5]. However, since the acceptance test is usually performed manually, the process is tedious, expensive and time consuming. Thus, automation of the acceptance tests seem as a promising initiative to ease and improve this process [5].

This paper focuses a part of the agile software development that is a requirements validation at the initial stage of software development. In particular, we intend to investigate and review the usage of acceptance testing and the suitable automated toolkits as an approach in requirements validation. To do so, we have identified few recent related works from existing experience reports and publication on automated acceptance test. We also have selected relevant open source tools and research-based tools for further evaluation and comparison.

The presentation of this paper is organized as follows: While section one presents the introduction, section two presents the research background related to acceptance testing as requirements validation. This is followed by section three that provides the related works on automated acceptance testing tools. Section four discusses the results of the comparison analysis. Finally, the conclusions and future works are summarized in section five.

2. Research Background
Acceptance tests or business-facing test is also known as a “functional”, “customer”, or “story” test. Acceptance tests are used to identify that the software to be developed will meet the customer’s needs and interests. It is a validation activity that should be performed by the customers to ensure that the intended software is acceptable and covers all of the main functionalities as in the initial requirements. However, in the real software development, the implementation of the acceptance tests is often being left to the sole initiative of the developers. As a result, the acceptance tests are often expressed in an overly technical notation, resulting in difficulties for the customers and domain experts to review the requirements. Moreover, manual acceptance testing processes are generally tedious, expensive and time consuming [5]. Therefore, this test is often neglected and done poorly by the development team.

**ABSTRACT:** Acceptance test is a crucial procedure in software development for the development of the right requirements that meets the needs and expectations of the stakeholders and users. Due to the problems faced when conducting this test manually, several automated acceptance tools have been developed. This paper presents a comparison analysis of selected automated acceptance tools from the existing literature and experienced reports, highlighting their requirements representation, programming language, end-user inclusion, usability and output representation. It is found that these tools are still at their infancy stage and there are still more rooms for improvements. This paper serves as a preliminary work for future works to develop better requirements validation tools that facilitate the acceptance testing process, as well as promote collaboration and communication between client-stakeholders and the development team.

**Keywords:** automated acceptance test; requirements validation; agile testing
Sensitive with these problems, some researchers have made the efforts to develop automated acceptance test frameworks. An automated acceptance-testing framework can provide significant value to projects by involving the customers to work together with the development team [6]. The customers are responsible to write the acceptance tests in the language that they understand. This task is usually done with the collaboration with the Business Analyst and the development team. This process is important to ensure that all business stakeholders and the development team have the same understanding of the requirements. Additionally, the defined test criteria can serve as a written “contract” between the development team and the business stakeholders. The acceptance test also can help to identify incomplete or ambiguous requirements.

However, introducing automated acceptance test can be very challenging, especially to a team with no prior experience in test-driven development (TDD) [7]. It takes time and effort to write and maintain these tests. Another issue is about the flexibility in notation and the ability to create automated test. User’s stories are mainly suitable to be used in small-scale software projects only. However, in larger-scale projects, some information workflows are better captured in diagrams, using UML diagrams, such as the use case and activity diagram [7,8].

There are a lot of automated acceptance tests tools that have been developed in order to overcome these issues. However, they are still open for further improvements and enhancement. Motivated from this, we plan to develop a better and robust automated acceptance test tool that can facilitate the acceptance testing process as well as promote better collaboration and communication between client-stakeholders and the development team. This tool is also expected to be able to maintain the consistency, correctness and completeness of the requirements.

3. Related Works
Creating test cases manually requires much effort and consumes time. Therefore, many researchers have proposed automated acceptance testing tools or frameworks to ease the process. It is found that automation of acceptance test is beneficial in many ways. However, it is still not much practiced due to the lack of good tools and innovation [7]. In this section, we present a descriptive review of five acceptance test tools: three research-based and two open source automated acceptance test tools. Our goal is to investigate and evaluate the usefulness of the automated acceptance testing tools as an approach in requirements validation.

1.1 AcceptSoftware
AcceptSoftware [9] extends EasyAccept [10] for automated acceptance testing that allows simple but powerful ATDD software development. The motivation for extending this tool is to enhance the tool to support constant changes made to acceptance tests. AcceptSoftware has the functionality to report failed acceptance test based on the test result history kept in the database. This history is found to be useful for analyzing the existing progress and making improvements.

Fig 1. AcceptSoftware’s Framework. The red-highlighted box is the enhancement to the EasyAccept tool[9].

However, this tool is at its preliminary work and it still has more room for future works. Apart from the acceptance test history report, there is no further enhancement on EasyAccept (refer to Figure 1 and 2 for comparison). This means that all of the other acceptance test automation processes are totally dependable on the EasyAccept’s framework. To write the acceptance test, EasyAccept provides some built-in commands that are closed to natural language. This tool is found useful to produce better quality software where the client and developer can sit together to define the acceptance criteria.

Fig 2. The outline of EasyAccept framework[10].

1.2 UCAT
To realize the benefits of acceptance testing within large-scale development projects, M. El-Attar and J. Miller [8] introduced a technique that utilizes the use case model. The approach utilizes the use case models supported by the robustness of diagrams and domain models. They presented three principle phases for creating acceptance tests from the UCs, as described in Figure 3. In the first phase, high-level acceptance tests (HLATs) for each UCs are developed. Then, a robust analysis is performed in Phase two to identify the object level information that will realize the HLATs. In the third phase, Executable Acceptance Tests (EATs) are developed for each UC, using the object level information identified in Phase two. They also present a supporting tool,
named UCAT (Use Case Acceptance Tester) to realize this approach. UCAT provides automation support for executing acceptance test developed through the proposed approach. It is a supporting tool that depends on FIT/FitNesse to allow analyst to build an entire UC model.

This approach is found beneficial since the acceptance tests are developed for each UC individually. The High Level Acceptance Tests (HLATs) are readable and can be used for high-level validation of the requirements. Moreover, traceability is provided where the test results are associated with UCs and can be automatically run through UCAT (see Fig 4). However, the efficacy of the executable acceptance tests developed is dependable on the quality of the UC model. It depends on the skills and experience of the analyst and customers to produce a high-quality UC model.

This tool has been demonstrated to a domain expert and development team and it has received positive responses. The annotation is found valuable and helps to reduce errors in test. However, this prototype tool is still under developed and needs further improvements.

1.4 Cucumber
Cucumber [12][13] is a BDD automated acceptance test tool that allows users to write the specification of application features and user scenarios in a plain text. It works with Ruby, Java, .NET, Flex or web application written in any languages. Cucumber allows users to describe the specification or features in a plain text in the form of Given-When-Then format, while the executable part of the test cases is written in Ruby code. Figure 6 shows an example of a Cucumber acceptance test or feature:

Cucumber's test case results are presented in an HTML report and are simple to interpret. Cucumber's plain English feature syntax is easy to understand and it facilitates the association of test cases with the use cases or features. However, Cucumber requires a developer or a stakeholder, familiar with Ruby to script the executable portions of the test cases. Cucumber's primary language for implementing the test cases is Ruby and its supports of other implementation languages, such as Java, is problematic.
1.5 Concordion
Concordion [14,15,16] is an ATDD tool for writing automated acceptance tests in Java development environment. It also supports other languages, such as the .NET, Phyton, Scala and Ruby. As stated in their websites, “Concordion is similar to Cucumber, but it focuses on readability”. Therefore, these tools are a relevant comparison in this study.

Concordion [14,15,16] is an open source tool for writing and managing automated acceptance tests in Java project. It is directly integrated with JUnit, which allows for easy use with IDE, such as the Netbeans, Eclipse and IntelliJ IDEA. The concept of using Concordion is illustrated in Figure 7.

Fig 7. Concept of Active Specification [15]

The active specification consists of two parts: (i) a well-formed XHTML document describing the system functionality (Figure 8), and (ii) fixture code (acceptance test) written in Java that implements a Concordion extension of a standard JUnit test case (Figure 9). The test results are highlighted with green and red colors for success and failed test cases respectively (Figure 10).

Concordion is found easy to setup and use. It takes very minimum learning curve for new users. It allows users to write the requirements specification in a plain language. However, the specification test data must be implemented with HTML tag, which may not be easy for inexperience non-technical users. Even so, Concordion is still a good tool that promotes collaboration between end-users and development team to specify and discuss the requirements.

4. DISCUSSION (RESULTS)
Conducting acceptance testing for validating requirements is important in order to get the right requirements before development. This practice will help all business stakeholders and development team to have the same understanding of the intended application. However, manual acceptance testing processes are tedious, expensive and time consuming. Thus, it is necessary to have automated acceptance test tool to facilitate the activity to get the utmost results. In this paper, we presented the review of research-based and open source automated acceptance test tools. Table 1 summarized the comparison of the selected tools.

Based on the review, we found that most of the tools accept informal representation of requirement as the input. For instance, Cucumber and Concordion allow users to specify their acceptance tests using English-like narrative in the format of “Given-When-Then”. This approach allows the non-technical users to implement this tool, as it does not require any programming skills to specify the acceptance tests. The ease of writing and reading the requirements specification also draws the business stakeholders to collaborate in the process. However, expressing the requirements in the form of user’s stories or scenarios is suitable in a small-scale software project only. The larger-
scale software projects usually write the specification using UML diagrams, such as the use case, data flow diagram and activity diagrams. In this case, tools that automate these non-textual requirements are still lacking. Further, we found that the UCAT tool generate acceptance test from the use case diagram only. This means that this approach totally depends on the skills and expertise of the analyst in creating a comprehensive use case diagram. Therefore, the success and effectiveness of this approach is highly dependable on the experience of the analyst. Users also need to learn and apply the concept of robustness diagram, when using this approach.

Table 1 Evaluation Result of the Automated Acceptance Test Tools

<table>
<thead>
<tr>
<th>Type of Tool</th>
<th>Tools Name</th>
<th>Requirements Representation</th>
<th>Programming Language</th>
<th>End-user Inclusion</th>
<th>HTML Report</th>
<th>Output Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Source tools</td>
<td>Cucumber</td>
<td>X X / / / / X / / / / X / /</td>
<td>X X / / / / X / / / /</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research-Based tools</td>
<td>UCAT</td>
<td>/ / X / X X / X / X X /</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acceptware</td>
<td></td>
<td>X / / / X X / / / / X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. CONCLUSION
Acceptance testing conducted during the early phase of a software development is crucial for the production of accurate software that meets the business stakeholders’ needs and expectations. In this regard, several automated acceptance tools have been developed and they are found to be beneficial for both business stakeholders and the development team. This is because these tools have the abilities to overcome the tedious work of the manual acceptance testing process. Drawn from this study, it is found that the active involvement from the customers (intended users) and communication between the customers and the development team are essential during the conduct of the acceptance test for accurate requirements. Specifically, customers should sit together with the development team to validate and confirm their requirements by specifying the requirements in a language that they understand. However, natural language alone may not be enough since it is error-prone and open for misinterpretations. Further, some information workflow can be captured better in diagrams, such as the use case and activity diagrams. However, tools that automate these non-textual requirements are still lacking [7].

Motivated from this condition, we plan to extend this research in order to facilitate the acceptance testing process, as well as promote better collaboration and communication between client-stakeholders and the development team. We plan to integrate some features of the existing tools with our requirements management tool [17] to develop a robust automated acceptance tool as a new approach for requirement validation. We also intend to develop this lightweight tool using HTML5 language to make it compatible with mobile devices. This tool is also expected to enhance the consistency, correctness and completeness of requirements.

ACKNOWLEDGEMENT
We would like to thank Fulgent Corporation, USA for funding this research.

REFERENCES


