Masters Project Final Report

December 2016

Project Title	**Online Training Program on "Acceptance Test Driven Development" for Pearson Lanka (Pvt) Ltd**		
Student Name	**L S DHARMASENA**		
Registration No. & Index No.	**2013/MIT/018** **13550181**		
Supervisor's Name	**Ms. K.M.G.B Nishakumari**		
Please Circle the appropriate	**Masters Program**	**Type**	
	MIT-eLearning	**Research**	**Implementation**

For Office Use ONLY	

Online Training Program on “Acceptance Test Driven Development” for Pearson Lanka (Pvt) Ltd

L. S. DHARMASENA
2016

Online Training Program on "Acceptance Test Driven Development" for Pearson Lanka (Pvt) Ltd

A dissertation submitted for the Degree of Master of Information Technology - eLearning

L. S. DHARMASENA
University of Colombo School of Computing
2016

ABSTRACT

The project course work is designed as an on-the-job training on “Acceptance Test Driven Development” or ATDD for software engineers in Pearson Lanka. At the designing of the course work, it is focused on applying e-learning principles, methods and tools in order to plan, design, develop, implement and evaluate the online training program.

The project is focused on identifying the performance problem in the client’s organization. Some of the issues are project delivery time is exceeding than the estimated, communication gap between key stakeholders, and untestable-unmaintainable codes. In this project, we are trying to analyze the root-cause for the performance problem and find a solution to solve the performance problem. It is also focused on learner analysis, content analysis and work setting analysis, which are essential for designing an effective e-learning course with appropriate content materials and structured courseware. Course work has been designed in order to support software engineers to give practical knowledge and enabling them to apply the learnings on their software development work.

The other main objective of the e-learning project is to provide a flexible learning environment where software engineers can follow the training without affecting their day to day work and other commitments. The course workload is designed to be at minimum level and encourage the software engineers to have discussions among them to share the experience and knowledge. The project report also includes details of the course lessons with overview and detailed storyboards of the course and it also describes the design considerations. Lesson content pages have been designed according to the multimedia principles in order to have higher learner engagement.

Details were included in this project report about course material development process and tools that have used for developed the course materials, such as Articulate storyline as the authoring tool, Adobe premiere and Adobe Audition as the audio editing tool and TechSmith Camtasia as the screen recording. In this report, it is also described the course delivery to the learner and learner activity management.

Next, it is described the delivery methods that have used to deliver the course work to the learners using tools such as Moodle LMS and Managing learner activities and providing help to learners. Pilot test has been carried out with limited audience in order to find out the effectiveness of the learning materials and course delivery.

Finally, it is described the evaluation of the project with the learners’ feedbacks, learner experience and conclusions that were made based on evaluation findings on future improvements of the project.

ACKNOWLEDGEMENTS

There are number of people, I would like to thank for helping me to complete the e-learning project with success.

First, I would like to thank **Ms. K.M.G.B Nishakumari** for her guidance, encouragement over the last year. Thank you so much for guiding and supervising me to complete the project successfully.

I would also like to thank **Mr. Nalina Wijesundara** and the management for allowing me to pilot the e-learning project in Pearson Lanka (Pvt) Ltd. Thank you so much for the support you have provided to me from data gathering to final evaluation.

I would also like to thank **Mr. Sachith Deshan** and **Mr. Laksith Liyanage** for providing me the subject matter expert knowledge in order to design and implement the learning content.

I would also like to thank **Mr. Himesh Kahatapitiya** for his administrative support in managing the e-learning environment.

I would also like to thank my fellow colleagues in Pearson Lanka (Pvt) Ltd, who has participated in the pilot project by giving their valuable time and effort and by providing the valuable feedback and responses.

I would also like to thank my fellow graduate students of MIT-eLearning in University of Colombo for your help and guidance to design, develop and implement the e-learning project successfully.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Information and communication technology has been rapidly improving and advancing in various areas such as economy, health, governance, administration during the past few decades. There is no exception for education and training as well. In fact, education has been the most influenced by the advancement of the Information and communication technology. E-learning can be considered as the evolution of education from traditional teacher centric face-to-face learning.

E-learning provides more interactions between learner-instructor, learner-content than in the traditional learning. With the advancement of the technology, e-learning content has been enriched by multimedia resources and interactive content. Interactivity helps to improve the learner engagement with the learning content. E-learning also empowers the learners to manage their learning mode and create meaningful learning environment.

Collaborative learning experience can be identified as another advancement of e-learning. Leaners can learn through others by collaboratively working on activities or tasks with the help of the communication technology. Social learning is one of emerging e-learning concepts that learners learn through other learners by sharing their experience and knowledge.

With the help of technology, e-learning has removed the temporal and geographical barriers. In general, e-learning environment learners are learning from different locations and not necessarily learnings happens on same time. Learning and communication can be happening in synchronous and asynchronous with various formats ranging from text, voice and video.

E-learning has been identified as effective way of learning, not only in academic area but also in corporate technical trainings. Including correct tools and technologies, online web based training can be the best option to deliver the corporate technical training. In this project we are trying to design, implement and deliver the e-learning solution for technical training, and discuss about the success and effectiveness of the project. Here we also take a look into tools and technology that can be used developing and delivering learning contents.

1.1. PROBLEM

The Client, Pearson Lanka (Pvt) Ltd is a software development company and their primary business is developing education and learning platforms, learning materials and tools for learners, primarily in North America and largely in other parts of the world. In the past few years, company has identified that their software delivery time has been increased in terms of what they have actually estimated and planned. And they need to find out the root cause and solution to the problem.

In order to finding the root cause, performance analysis has been carried out and at the end of the performance analysis, it is found that, (1) The estimations they have made were based on the details in the User story. (2) Most of the time, User stories do not have accurate information. (3) Testers found mismatches of what is expected versus what is delivered at very later stage, quite closer to delivery time. (4) Correcting one scenario requires regression testing across the system. (5) Regression testing currently carried out manually and it is time consuming. (6) Developers are afraid to perform code refactoring due to lack of test coverage.

By analyzing the problem and root cause, it is found that problem is mainly due to lack of knowledge and practices of agile development, such as (1) Writing proper User stories with accurate and sufficient details, which enables accurate estimations. (2) Writing early acceptance test, which improves clarity on requirements. (3) Get Developers, Testers and Product Owners into same understanding and expectation of the delivery. (4) Automated tests to reduce testing time and effort. (5) Promote code refactoring as development practice to improve maintainability and performance.

By conducting a learner assessment, it is found that 4 categories of employee group involve in the process of delivery and development in the company. (1) Product owners: people who represent the end user (Learner, Educator or Institute) and bring the requirement or business needs. (2) Developers: people who develop the software according to the requirement. (3) Testers: people who test the software for desired outcomes. (4) Management – people who facilitate the other groups for seamless operations.

In order to improve the delivery time and the quality, it is suggested to the company, it should provide training on above mentioned areas, (1) Writing proper User stories. (2) Writing early acceptance testing. (3) Give clarity and understanding among the Product owners, Testers and Developers by providing better communication tools. (4) Writing Automated Tests. (5) Writing Unit test to improve code refactoring opportunities.

“Acceptance Test Driven Development” course is designed as a solution to the above identified training needs and it will be developed as an online e-learning course, so whoever later joining to the company can easily follow the course and get train themselves with the relevant knowledge and skills

The main symptoms the client experiencing is the project delivery time is exceeding than what is estimated. The estimation that the team is providing for a particular project is not accurate enough to construct a project road map. This exposes project at risk in terms of revenue as well as the customer satisfaction.

In current development practice in Pearson Lanka, teams follow coding-testing approach, which means developers start the coding first then testers will execute the test plan at the testing stage and after that, product owners will carry out the user acceptance test (UAT) at the final stage. Even though developers write unit-test for the implementation and testers automated the testing, it doesn’t necessarily mean, the implementation has met the acceptance criteria. If there is a mismatch between the acceptance criteria and the actual implementation, it can only be found at very late stage and correcting them will affect the delivery deadline and budget.

It is visible that root cause lies in the software development practice. Main obstacles team facing is (1) Unclear requirements. (2) Communication Gap between Developer, Tester and Product owner. (3) Testing effort is high if the implementation needs to be changed. (4) Because of lack of test automation

1.2. SOLUTION

There can be two options to solve a performance problem (1) Instructional solution. (2) Management Solution.

However for this particular problem which is related to knowledge and skills and there aren’t any cost effective management solution can be found as a solution to the problem. It is not a feasible solution to recruit people who having the knowledge and skills of agile development practice. Even though company has provided tools such as JIRA, teams are still experiencing getting unclear, insufficient information for user stories.

As an Instructional solution for the problem, employees need to be provided proper training on areas, (1) How to write proper User stories. (2) How to writing early acceptance testing.

(3) How to get better clarity and understanding among the Product owners, Testers and Developers. (4) Writing Automated Tests. (5) Writing Unit test for better refactoring.

“Acceptance Test Driven Development” ATDD training is suggested as an Instructional solution for the performance problem.

1.3. GOALS AND OBJECTIVES

The primary goal of this e-learning project is improve software development efficiency by introducing the acceptance test driven development methodologies into the software development process by educating the software engineers through an online course.

And following objectives has been identified in order to support the primary goal

- Train software engineers and product owners to define clear acceptance criteria for user stories and reduce misunderstanding about the project requirements among product owners, testers and developers.
- Guide software developers on how to write testable code with test-driven development
- Guide software developers on how to write effective Unit Tests to validate the code implementation
- Promote effective refactoring techniques that can be used to improve to software code quality.
- Encourage to practice test automation in order to test the applications and for fast release cycles
- Incorporate knowledge of emerging software development techniques, methodologies and best practices in software industry into the software development process of the company

1.4. SCOPE

Introducing agile development process into current software development practice is needs to be done gradually and it also has to be practiced over a period of time in order to fully integrate as the development process. With the limited time we have pilot project is mainly focus on first step of agile development process that is focus on defining the clear acceptance criteria for the software development project requirements. Followings have been identified as deliverables of the e-learning project.

- Identification of target group for the online ATDD training and their current knowledge of the Test Driven development
- Design modules and learning outcomes
- Design course modules with pre-test for personalize learning paths
- Design course materials for course content and assessments
- Development of course outline, course module, course content and assessments
- Setting up e-learning environment, configure course structure and integrate course materials with the e-learning system

Proposed course will contain following learning objectives

- Recognize ATDD concepts, principles and practices
- Write acceptance test with specification steps for user story using gherkin
- Link the steps to step definitions using cucumber and asp.net
- Verify the acceptance criteria of the requirement against the code implementation
- Improve the code quality, system performance and design by practicing code refactoring

1.5. MOTIVATION FOR AN E-LEARNING PROJECT

There are few options analyzed on delivering the instructions. (1) First option is give workshop training by industry experts to all teams. This option has advantages as well as many disadvantages. Main advantage is team is getting hands on experience from SMEs (Subject Matter Experts). They can raise questions and get clarification then and there. But there are many disadvantages were found from this workshop training. (a) Time constrains; not every employee has free time to join with workshop training. (b) Company has to bear the

cost of unproductive time that employees spend on the training while away from their actual work. (c) Company has to facilitate resources for the training. (d) People are joining frequently to the company as well as leaving frequently, means company has to organize more and more workshops in order to training the staff. (e) Workshop training has limited amount of time and not every employee will be able to achieve the best out of the training with that limited amount of time.

(2) Second solution is to have the training as an online web based training. It has many advantages such as (a) No limited time frame to achieve learning (b) Employee can have the training at his own pace. (c) Company doesn't need to organize or facilitate training as people are joining to company. (d) Employee productivity not affected with the training as it can be followed at any preferred time

CHAPTER 2: BACKGROUND

2.1.OVERVIEW

Target group for this ATDD online training program is software developers in Pearson Lanka (pvt) Ltd. Pearson Lanka is part of Pearson PLC, world's leading learning company and Pearson Lanka is a leading provider of software development and remote infrastructure management services to multiple technology groups across Pearson. Its 600 member team based in Colombo office at Orion city is a core part of Pearson's global learning technologies organization, and services technology teams across North America, Europe and Australia.

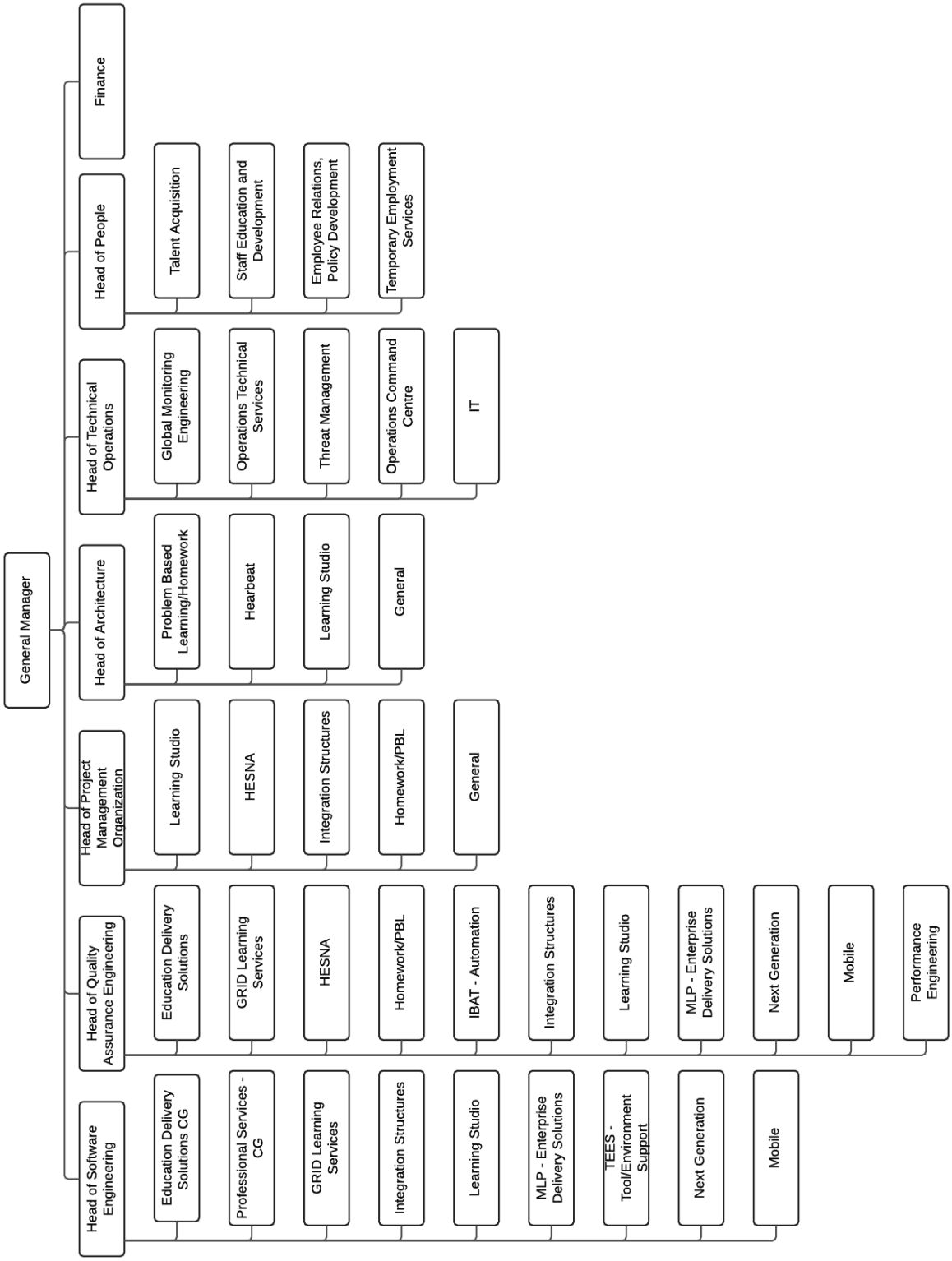
Pearson Lanka development center delivers online education platforms such as MyLab and Mastering, Learning Studio, OpenClass, REVEL with the collaboration of North Americas Product and Development teams. In addition to development work, Pearson Lanka involves in Global Monitoring, Operations and Technical Support around 24x7.

Pearson Lanka follows agile development process as the main development process and along with DevOps culture which brings development, quality assurance and operations into more collaborative, even though the corporate structure has separate reporting hierarchy. In the current development practice, developers and testers are getting the requirements from product owners in the form of User stories. User story contains the details of the requirement and acceptance criteria for the expected implementation. Team discusses the user stories on the priority order and improves the clarity of acceptance criteria or requirement details. After having prioritized the User Story, developers start working on the implementation and validate the functional implementation with the help of unit test. If the implementation satisfies the requirement, it is given to the testers for manual testing. Once the manual testing is over, testers begin to write the test automation based on the implementation what coders have delivered to them. In parallel to test automation, implemented feature is given to the Product owners for the User Acceptance Testing (UAT). Once the user story passes the UAT, it will be deployed to the production environment and will be available to the end-user. If user story failed in the UAT, it is asked to change the implementation, and whole cycle needs to be followed again.

With the current practice, it is found that delivery time will get impacted when there is unclear user story or misunderstanding among team members about the requirement. Pearson Lanka considering changing the current process and doing so, reduce the delivery time and quality

assurance process can be reinforced. Figure 1 represents the organization chart of Pearson Lanka.

Figure 1: Organization chart of Pearson Lanka



2.2. LITERATURE REVIEW

This section discusses about researches, similar course work and online training programs that were carried out and learnings from those works. It is hard to find similar online course works or online trainings within the corporate world in Sri Lanka. However, we can look into similar trainings, research and surveys that carried out on web-based training in the other parts of the globe.

2.2.1. BEST PRACTICES IN THE TRAINING OF FACULTY TO TEACH ONLINE

According to the Wolf [1], Distance education is often used as a cost-efficient way to train employees. Researches revealed the scarcity of scholarly work in this area. To determine best practices in training for teaching online, a faculty training program was examined and experts were interviewed. Both educational organizations and corporations offer online education programs, and there are enough similarities in the way in which these programs are presented to generalize the findings of their study to corporate training. The study found that successful online training programs are led by faculty trained to teach online. Training programs are successful when faculty have computing skills before enrolling in the training, are trained using the course delivery system with which they will be teaching, have ongoing institutional support, and are motivated to work in this environment.

Case study has been carried out on the faculty-training program used at the University of Maryland University College (UMUC) was examined. Data were collected concerning current best practices in training using distance education, and recommendations based on that data were made. UMUC is a nontraditional higher education institution focused on working adults and active duty military personnel. UMUC invests significantly in "the high-technology infrastructure essential to emergence as a world leader in the delivery of online learning" [2]. Like many businesses, UMUC must train greater numbers of employees (faculty) each year. The number of online enrollments at UMUC continues to grow; in 1997, online enrollments were 3,842, whereas in 2004, online enrollments were 126,341. According to Bonk [3], training activities are aligned with key functions, or core competencies, of organizations. Because distance education is a core competency of UMUC, it is appropriate to study how the organization uses distance education to train its faculty.

2.2.1.1. CLASSROOM TEACHING HAS NO CORRELATION ON SUCCESSFUL ONLINE TEACHING

Successful faculty need not have face-to-face teaching experience before teaching online. According to Muirhead [4], there is no correlation between "quality teaching in the classroom and teaching effectiveness online." Muirhead concluded that the best instructors in a face-to-face setting may do poorly in distance learning if they are not comfortable with the technology.

2.2.1.2. MINIMUM COMPUTING SKILLS ARE REQUIRED FOR SUCCESSFUL ONLINE TEACHING

Successful faculty should have a minimum set of computing skills before enrolling in training to teach online. This minimum set of competencies includes use of the computer, the Internet, and online applications. Henning [5] argues that prospective participants are screened for technical skills and that additional training be given where the faculty members need it.

At UMUC, faculty is asked to self-assess their technical skills before beginning the training Although this tutorial has helped increase the success of faculty who take the training, because faculty self-assess they sometimes still overestimate their abilities. Burke [6] described what happens when faculty who do not have the required skills attend the training. In his experience, although instructors were prescreened to see if they had the appropriate Internet skills, some instructors were not deterred from taking the class even though the self-assessment showed that they did not have enough skills to use Internet tools well. These instructors lagged behind other instructors in the course, creating awkward situations where one instructor struggled in front of colleagues. Some instructors dropped the course; others did not, although their lack of skills clearly impeded the progress of others and caused two of the poorly prepared instructors to fail. Effective programs offer separate training for faculty who need to improve their computing skills before training to teach online, and assessment of those skills is conducted by means other than self-assessment.

2.2.1.3. EFFECTIVE TRAINING PROGRAMS USETHE COURSE DELIVERY SYSTEM

Effective training programs are designed so that faculty is trained to teach online using the course delivery system with which they will be teaching. Effective programs also require

faculty to work as learners and access the course delivery system from the learner's perspective. By forcing the teacher to take the role of student, the teacher finally learns to understand students' "fears, stress, frustrations, and joys in learning in the Web-based environment" [7].

At UMUC, faculty participates first as students and learns to use the various features of the system, such as submitting assignments and working in study groups. They are then placed in the role of teachers, with other trainees assigned to their "class." For this portion of the training, instructors learn how to create assignments, manage online conferences, and provide student feedback.

2.2.1.4. SUCCESSFUL TRAINING ENCOMPASSES PEDAGOGY

Successful training encompasses pedagogy, although methods for introducing pedagogy into the training differ. Methods advocated include article review and discussion [8] collegial interaction [9], exploring the various technologies available and discussing the pedagogical basis for their use [10], working in teams [11], and acting as facilitator for class discussions [12]. Pedagogical topics incorporated into training also vary and included effectively managing threaded discussion [8]; encouraging interaction [9][10][13] adult learning principles [13]; understanding and developing a learner-centered mindset [10]; modeling [8]; managing and disseminating disparate views [12]; maintaining contact with learners [7]; evaluating learning at a distance [7]; developing a careful communication style that does not inadvertently offend [7]; selecting, evaluating, and processing the large quantities of information available online [14]; time management [5]; and developing students' cognitive thinking skills [14].

2.2.1.5. ONGOING FACULTY SUPPORT IS NECESSARY

Effective distance education programs provide ongoing faculty support in the form of mentoring, shadowing, continuing education workshops, or some combination of all of these. However, it is not known which of these methods or which content in continuing education workshops is required for effective online teaching. UMUC provides all of these types of support to its faculty, whether they are teaching online or face-to-face and regardless of geographical location. Effective programs survey their faculty to determine what types of support are most desired.

2.2.1.6. MOTIVATION IS PRIMARY FACTOR FOR SUCCESSFUL ONLINE TEACHING

Motivation is the most important factor when choosing faculty to teach online. Successful online faculties have been noted to be willing to make the transition to the new environment, with all the attendant risks and rewards. Kearsley [15] noted that online instructors must spend one to two hours or more every day at the computer reading and responding to students. Instructors must like interacting with students on a one-to-one basis. They must like troubleshooting and problem-solving, because they will do a lot of both. Instructors also need a lot of patience to deal with glitches in technology on a daily basis. Because most interaction will be written, instructors have to like to write. Even being able to type fast is a benefit. Fredericksen [16] note that successful online faculty "have a passion for teaching [and] are willing to rethink how they teach and assess learning".

Successful distance education programs provide incentives for faculty to move to online teaching. Palloff & Pratt [17] point out that not all faculties are suited for the online environment, and academic institutions could make serious mistakes when they make their decisions about who should teach online. The decision regarding who should teach online is often based on faculty criteria--usually someone considered a content expert or someone with a reputation for being popular with students in the face-to-face classroom. Being popular or entertaining face-to-face does not translate to the online environment, where the instructor's personality is reduced to text on a screen. Focusing on faculty who are content experts may present a problem. Knowledge of subject matter alone is inadequate preparation for online teaching.

Additional incentives may be required when a distance education program is new and institutions are trying to encourage existing faculty to move online; however, it is unclear which form of incentive is best. Until spring 2004, UMUC paid faculty teaching online an additional stipend. Because that incentive is no longer necessary to lure faculty to the online environment, it has been discontinued, and faculty teaching online are paid the same as those teaching face-to-face. Successful programs choose incentives that are meaningful to their faculty.

2.2.1.7. FACULTY SHOULD BE INVOLVED IN COURSE DESIGN

Faculty should be involved in course design. If faculty are expected to design courses without the assistance of an instructional designer, instructional design theory is included as part of the training to teach online. Olcott [18] noted that course design is often a collaborative effort, even though individual instructors must assume the major leadership role for the design of their courses. Distance learning courses require working with instructional designers, production technicians, evaluation experts, and support service units. Instructors new to distance teaching often feel that their courses have been taken over by "the experts" and that autonomy and instructional controls have been compromised. However, efforts by all stakeholders to place instructors at the center of the teaching process make distance learning a quality experience for instructors and students. UMUC develops courses using a team, which includes a faculty member, an administrator, an instructional designer, an editor, and a project manager.

2.2.2. SELF-DIRECTED STUDY THROUGH E-LEARNING

E-Learning has been touted as a significant trend in which the independent learning activities of the learner (or organization member) tie in with the goals of the organization, as well as a major revolution seeking a transformation in the notion of human learning. But conversely, perhaps the advantages of e-Learning are not to be found on the side receiving the instruction, but rather on the side that is providing the instruction. To put it more frankly, the alarm has been sounded that while e-Learning has demonstrated its power as a human resource management tool, it may not exhibit effectiveness as a personal management tool. It has to arrange e-Learning once again from the perspective of “self-directed study,” and will examine it from the viewpoint of being a tool for self-management, rather than a tool for managers.

In his critical analysis of the e-Learning revolution, Wesley [19] sounds the alarm over the introduction of learning management system (LMS) as a supervisory outcome in the guise of motivation. To define it as simply as possible, he theorizes that e-Learning has become a means of broadcasting instructional materials through the Internet, intranets, or extranets. In modern society, where emphasis is placed on creativity and innovation within the “Knowledge based Economy,” learning by company employees is stressed. He points out that while this fact lies in the backdrop to the e-Learning boom; e-Learning is becoming commercialized at the same time. A style of purchasing commercial software and commercial

LMS has become entrenched, and there can be no hope that the learning itself will change, with concerns to the effect that this will only serve to strengthen the management side. Wesley introduces readers to the fact that frameworks exist in the form of social and economic theories on supervision (monitoring) and motivation. LMS has three roles: (1) supervision (monitoring), (2) the collective development and management of learning contents, and (3) creating a virtual learning community. The author draws attention to the fact that the strengthening of supervision from simply introducing LMS must not be allowed to hinder creative and innovative ideas.

2.2.3. ESTABLISHING A POSITIVE LEARNING ENVIRONMENT (PLE)

Tobin [20] claims that a company as a whole must be made into a "Positive Learning Environment (PLE)" in order to move forward with self-directed learning. A PLE is defined as having all employees from the president down to regular staff members adopts an attitude of constantly learning in order to achieve business objectives. It describes an organization in which all learning activities are directly tied in with the objectives of the individual, group, and company as a whole, and every member searches for new ideas, tries new approaches, and jointly learns by sharing ideas with others.

2.2.4. ONLINE TRAINING IN AN ONLINE WORLD

In 2001, Bonk [21] has conducted a survey targeting online training in work related settings with 201 respondents. These individuals were asked about their Web-based training practices, experiences, tool preferences, instructional approaches, assessment methods, obstacles, and support structures. Among those completing this survey were corporate trainers, instructional designers, training managers, and Chief Learning Officers. The respondents represented a range of industry types that included information technology, financial services, education, manufacturing, government, consulting, military, and healthcare. Nearly all of them were either users of Web-based training or decision-makers regarding it. In addition, most were active members of training or online learning organizations.

In his finding Bonk has given 15 recommendations for online training environments. First recommendation is related to survey itself for future surveys should be more focus on selected industry or job function. Other recommendations will be useful for building an online training environment

- **Longitudinal Reports:**

  The Web is emerging as one of the preferred methods of employee training. Longitudinal research might explore these trends over the coming years or decades. For instance, such research might track attitudes about organizational support structures as well as employee attitudes and achievement related to these new forms of delivering training. It might also longitudinally explore differences between organizational interest and commitment in Web-based learning, as well as the types of online delivery methods utilized and promoted. Additional research might reveal where and when blended approaches are preferred to either fully online approaches or conventional face-to-face training. Other possible longitudinal variables include the reasons various organizations are interested in Web-based training, the types of training offered, and the principal reasons behind outsourcing the development and delivery of Web-based content.

- **Evaluation and Assessment:**

  Alternative online assessment measures need to be developed that address employee skills and competencies. Given the findings of this survey, organizations should evaluate the completion rates of their courses as well as the motivational characteristics embedded within them. In addition, time to competency measures might be added to, or in some cases, replace traditional ROI measures. Along with changes in assessment practices, there is a need for comprehensive documents that survey the forms of online assessment and evaluation commonly used. Such documents might also provide case examples of success stories and potential problems in assessment.

- **Use of Learning Objects :**

  Organizations should consider how the use of learning objects in instruction relates to their strategic planning, including their knowledge management efforts. Such planning documents are vital since the use of reusable learning objects in online instruction will proliferate during the coming decade. Of course, the growth of this field will depend on the development of effective standards for shareable courseware. Decisions must be made regarding the size and type of objects shared, systems and tools used for sharing, and the ownership and use of learning objects.

- **Online Learning Policies and Procedures :**

  Most organizations still need to develop strategic plans related to e-learning. They might develop guidelines as to acceptable levels of student course completion, skill retention, employee satisfaction, and return on investment. In some instances, they will need to develop clear policies regarding the ownership of online course materials and applicable royalties. Organizations with significant training concerns might adopt policies related to instructors and other employees who provide freelance online instruction for other institutions or organizations. They might also attempt to clearly articulate why certain courseware tools, policies, and expectations have been adopted related to Web-based instruction.

- **High Growth Tool Development Efforts:**

  Few online software tools address the diversity of instructional and learning needs mentioned by participants of this survey. High growth areas revealed in the survey included tools for online course evaluation, instructor demonstrations, student task collaboration as well as storytelling, trainer task collaboration, learner critical and creative thinking, instructor feedback and annotations, and Web resources specific to one's field. As the survey report indicated, there is a dearth of pedagogically interactive and motivating activities within Web-based learning environments. The first organization to develop a suite of pedagogical tools or templates addressing motivation, teamwork, and critical or creative thinking (e.g., tools for debate, role-play, brainstorming, timeline, etc.) will add significant value to the present state of learning management systems and instructional courseware. Finally, as online learning globally extends around the world, tools for language support will be increasingly requested and required.

- **Tool Development Partnerships:**

  Courseware companies might seek partnerships for tool development and testing with universities and institutes that have well-established learning technology, information science, and instructional design departments. In serving as a testbed for emerging tools, technology centers at those universities and institutes can research and showcase product innovations. They might also spearhead significant research grant proposals and help form institutional consortia. With numerous

technologies, content, and service providers, partnerships among firms and universities can bridge knowledge gaps and provide comprehensive as well as competitive solutions.

- **Training the Trainer:**

  Corporations and other learning organizations need to consider not just the learners but, if facilitators, mentors, or synchronous instructors are utilized, the trainers of those learners. It will be difficult to train in the online world without a new skill set. External supports such as Web resources, online "Train the Trainer" courses and institutes, asynchronous discussion forums and communities, online mentoring, and noted experts and consultants can offer instructional assistance. Internally, intranets can provide rich training resources and alternative avenues of such support. In effect, instructional design support and guidelines can help reduce the tension felt by those teaching online for the first time. Of course, adequate time to learn the new systems and tools is vital. While there are masses of available training resources, the use of Web-based training courses and resources is a growing area for e-learning service companies.

- **Freelance Instructors and Designers :**

  The survey respondents predicted fast growth for freelance instruction. How their instruction, training, and consulting wares are bartered online remains an open issue, however. Already one can list e-learning needs using "request for proposal" forms from THINQ as well as hire experts from an array of disciplines listed online at Hungry Minds University. Other innovative organizations might create tools or systems that foster instructor exchange programs, trainer-to-trainer online mentoring, trainer online job-sharing, instructional resource exchanges, and instructor communities in the area of e-learning. Expert pools and knowledge exchange programs might be common in the near future not only for corporate trainers and instructors but instructional designers as well.

- **Organizational Promotion:**

  Employees need to be aware of their online learning options. Marketing new courses with testimonials and up-to-date information will help convince people to take the online course. There should also be incentives for trainers, instructors, and instructional designers for high quality course design and delivery.

- **Organizational Support:**

  An organization must support a range of people within its e-learning initiatives. For instance, online learners need adequate technology access and organizational policies that help them to complete their online course requirements. Instructional designers new to e-learning require training, system support, and perhaps even certification. At the same time, online trainers need new skills as well as access to examples of best pedagogical practices for synchronous and asynchronous delivery systems. Finally, training evaluators need access to data from e-learning courses and events. All these e-learning stakeholders and participants demand attention and support for e-learning success.

- **Information Portals:**

  The survey uncovered a need for online resources such as newsletters, information on training institutes, course catalogs, library resources, survey and evaluation tools, and course design guidelines for online training and instruction. As this area emerges, there is a pressing need to make sense of the available courses, course platforms or learner-management systems, Web-based delivery tools, and online resources. While a number of e-learning information portals and reports are emerging, there remain many areas for development, including the documentation of the companies in this area, the sharing of best practices and online documents, and the generation of online trainer ratings.

- **Online Communities :**

  The survey results also exposed a need for an online community of instructors and instructional designers. Trainers and instructors want expert advice, answers to teaching problems, stories of online experiences, and mentoring services. While primitive forms of such communities exist, none address all these needs.

CHAPTER 3: PLANNING AND ANALYSIS

A needs analysis needs be conducted at the start of any development effort to determine whether:

- training is required to fill a gap in professional knowledge and skills; and
- E-learning is the best solution to deliver the training.

The needs analysis help to identify general, high-level course goals.

Target audience analysis is another important step in analysis. The design and delivery of e-learning will be influenced by key characteristics of the learners (e.g. their previous knowledge and skills, learning context and access to technology).

Similarly, an analysis is needed to determine the course content:

- Task analysis identifies the job tasks that learners should learn or improve and the knowledge and skills that need to be developed or reinforced.
- Topic analysis is carried out to identify and classify the course content.

3.1. NEEDS ANALYSIS

Effective training depends on knowing what is required for the individual, the department and the organization as a whole. With the limited budgets and the need in cost-effective solutions, organization needs to ensure that the resources invested in training are targeted at areas where training and development is needed. Target group for the need analysis is software engineers including product owners, developers and testers. Target population is 600 employees located in Sri Lanka office. In order to identify the need of having “Acceptance Test Driven Development” training, few questions (Table 3.1) given to the selected sample of 50 employees from the target population.

Main purpose of the survey is to identify:

- Any direct relationship between clarity of the user stories with delivery time?

- How many teams/employees are impacted from having incomplete details on user stories?

3.1.1. TARGET AUDIENCE

Primary target audience is software engineers directly involve in software development. The software engineers are ranging from well experienced to novice or fairly fresh to the software industry. They are also ranging from different technology backgrounds. Currently, Sri Lanka office has 600 software engineers and 50 from them is selected as the sample group

Secondary audience is their respective line managers; they are either development managers or quality assurance managers. The primary target audience learnings will be applied to the project work and thus project managers also considered as Secondary audience.

Territory audience is Higher Management. Cost-benefit and final Return on investment is important to the higher management in order to decide the successfulness of the e-learning project.

3.1.2. DATA COLLECTION METHOD

Closed end questionnaire has been used for primary data collection. This questionnaire contains questions about the nature of the project they are working and issues they are facing in terms of process. Employees experience and other work experience details were taken from already existing resources such as employee records from Human resource department.

Table 1 represents the questionnaire that has been given to the sample group and their responses.

Question	Options	Result	Percentage
1)			
How frequently do you deploy your software product into the production environment?	2 weeks	0	0%
	1 month	14	28%
	3 months	8	16%
	6 months	28	56%
2)			
How do you rate your project in terms of on-time delivery?	Within planned deadline with original features	4	8%
	Within planned deadline with adjusted	26	52%

	features		
	Adjusted deadline with original features	8	16%
	Adjusted deadline with adjusted features	12	24%
3)			
How do you rate your project in terms of quality during last 2 years?	0 Hot fixes; 0 rollbacks	8	16%
	1-5 Hot fixes; 0 rollbacks	13	26%
	1 rollback or 5-10 Hotfixes	23	46%
	More than 1 rollback or more than 10 Hotfixes	6	12%
4)			
Most of the defects are due to	Logic errors in the implementation	9	18%
	Corner cases which are not originally mentioned in the user story	22	44%
	Misunderstanding	10	20%
	Other	9	18%
5)			
How do you rate majority of the feature requirements that is given to you in terms of clarity	Poor and vague	21	42%
	Moderately descriptive	18	36%
	Excellent	11	22%
6)			
How often do you need to rework the implementation	Never	4	8%
	1-2 times	15	30%
	3-5 times	17	34%
	6 or more times	14	28%
7)			
Why do you need to rework the implementation	Due to defects	16	32%
	Due to requirement change	18	36%
	Due to performance issue	4	8%
	Other	12	24%
8)			
How frequently do your requirements get changed	Never	1	2%
	1-5 cases	17	34%
	6-10 cases	21	42%
	More than 10 cases	11	22%
9)			
When do your requirements mostly get changed	At backlog grooming	7	14%
	At sprint planning	4	8%
	During the sprint	22	44%
	At UAT testing	17	34%
10)			
Do you get sample data for the requirement	Never	16	32%
	Few occasions	27	54%
	Most occasions	6	12%
	Almost every occasion	1	2%
11)			

How often do you need to communicate with the product owner to get the clarification	Never	0	0%
	1-2 times	16	32%
	3-10 times	23	46%
	More than 10 times	11	22%
12)			
Do your original estimate of implementation is different from the actual	Less than the estimated	10	20%
	not much of a difference	11	22%
	more than the estimated	29	58%

Table 1: Questionnaire for leaner analysis

After analyzing the results, it was given conclusion on:

- Delivery time has directly impacted when user stories are vague and unclear
- Many teams/employees are impacted from not having sufficient details on user stories before starting the implementation.

3.2. LEARNER ANALYSIS

Main target audience is software developers, testers and product owners in Pearson Lanka (pvt) Ltd. Target population is 600 software engineers and to analyze their characteristics survey was carried on a sample of 50 employees and following details have been captured.

3.2.1. AGE DISTRIBUTION

Table 2 represents the Age distribution data captured from respondents.

Age Group	Count
18-25	10
25-35	22
35-45	14
45+	4

Table 2: Leaner distribution by age

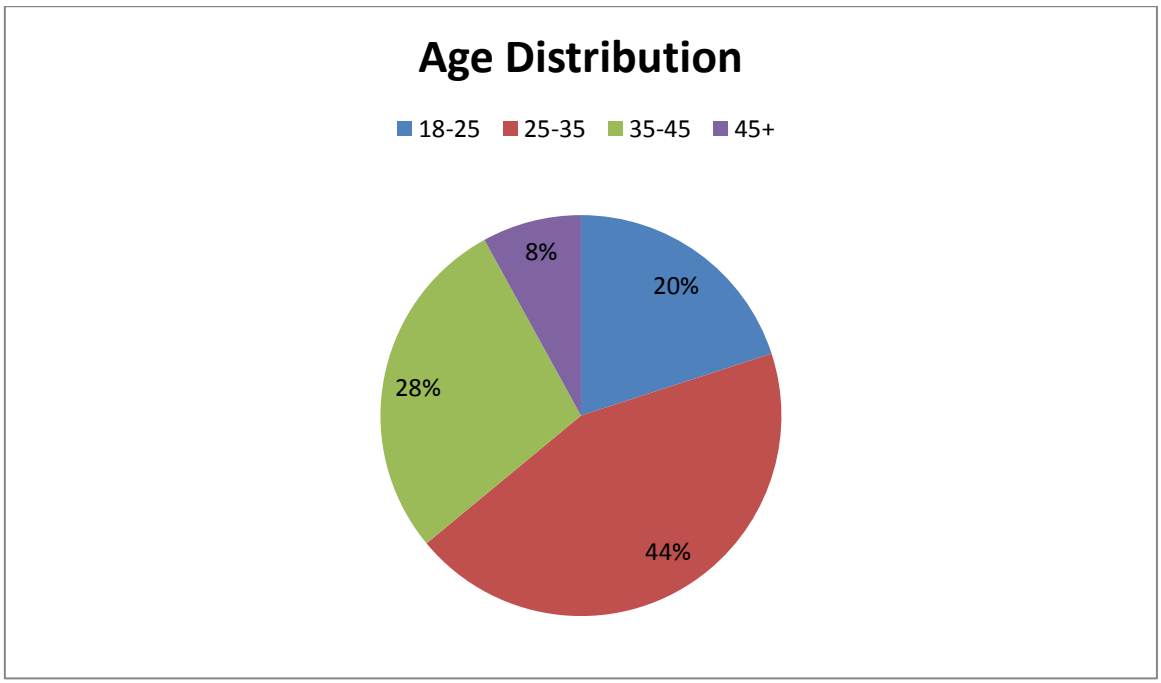


Figure 2: Age distribution percentages

As shown in Figure 2, majority of learners are from age group 25-35.

3.2.2. DISTRIBUTION BY INDUSTRY EXPERIENCE

Table 3 represents the work experience of the respondents in the software development industry

Industry Experience	Count
< 1 Year	7
1-3 Years	17
3-6 Years	14
> 6 Years	12

Table 3: Leaner distribution by industry experience

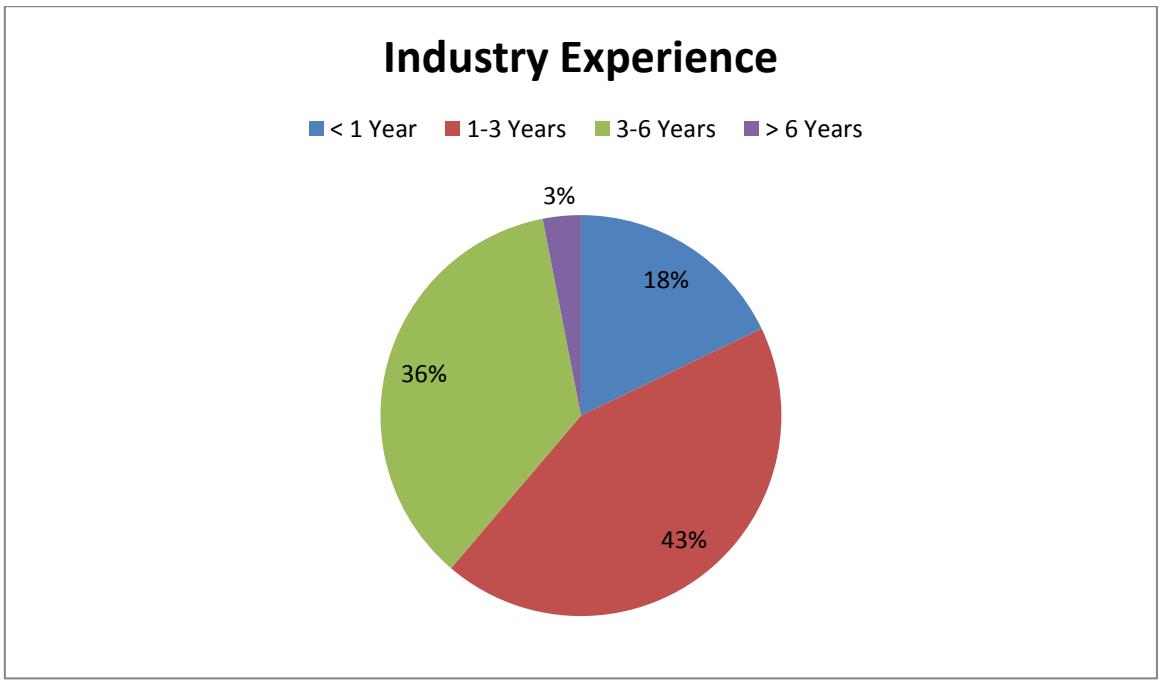


Figure 3: Percentages of distribution by industry experience

As shown in Figure 3, majority of learners have 1-3 years of experience in software industry

3.2.3. DISTRIBUTION BY JOB CATEGORY

Table 4 represents the respondent's job category according to their job title

Job category	Count
Developer	27
Tester	17
Product owner	6

Table 4: Leaner distribution count by job category

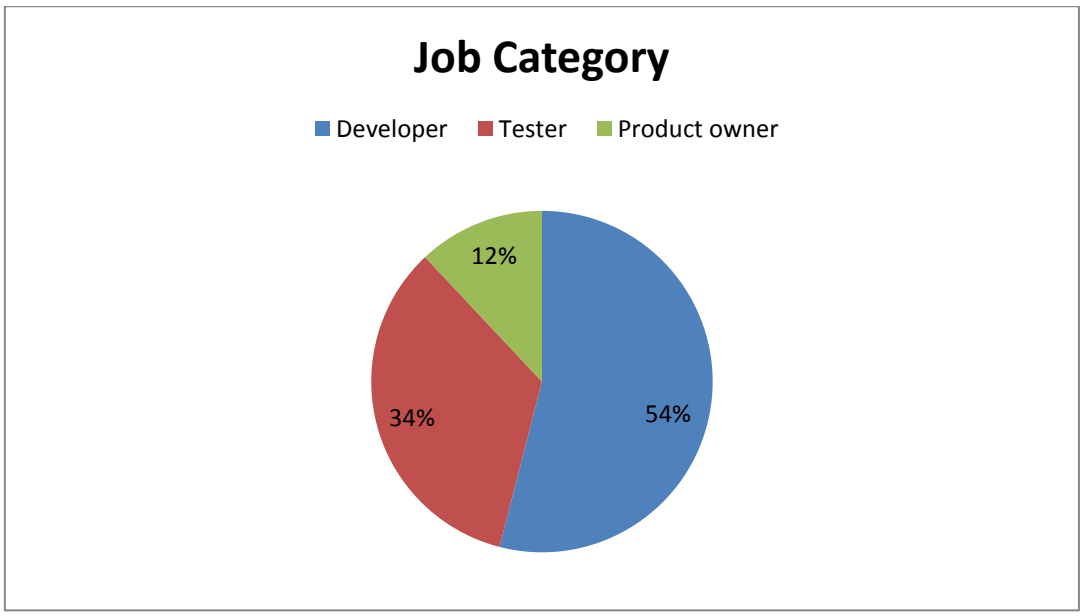


Figure 4: Distribution percentage by Job category

As show in Table 4 and Figure 4, Majority of learner are developers

3.2.4. DISTRIBUTION BY AWARENESS OF ATDD PROCESS

Table 5 represents the respondent's awareness on ATDD process

Level of Awareness	Count
Never Heard	13
Never practiced	27
Practicing	10

Table 5: Leaner distribution by awareness of ATDD

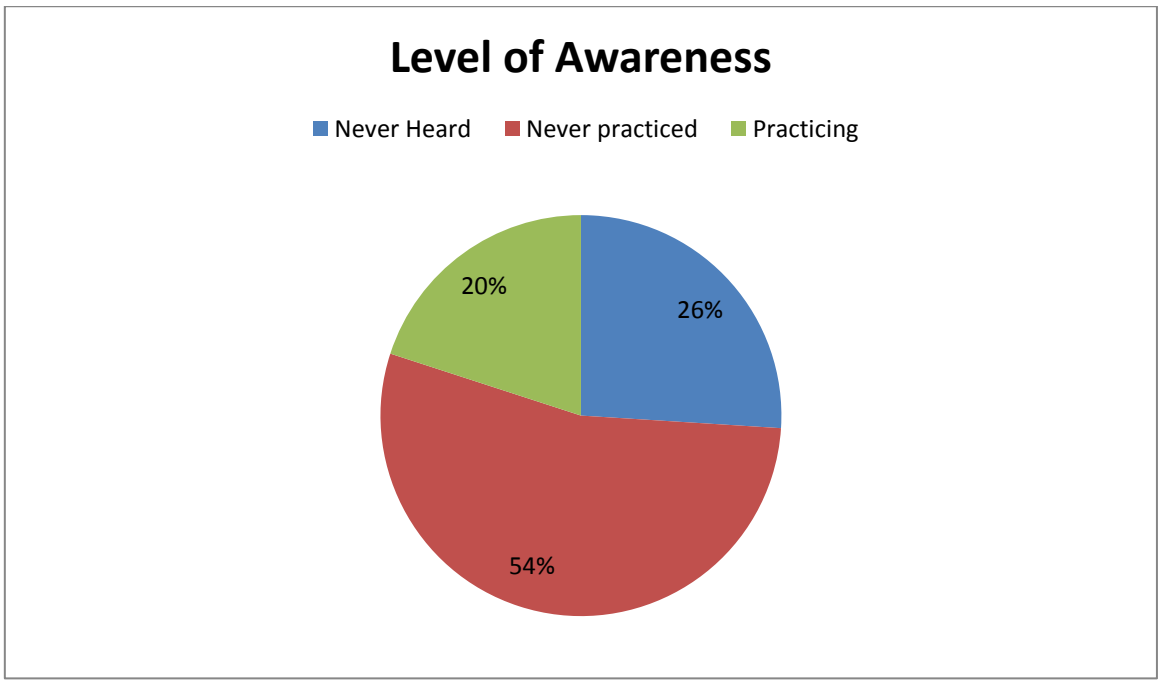


Figure 5: Distribution percentage by Level of Awareness of ATDD process

According to the sample data (Figure 5), majority of learners never practiced ATDD process previously in their software development process.

3.2.5. DISTRIBUTION OF UNDERSTANDING THE BENEFITS OF ATDD

Table 6 represents the distribution of the employees who thinks ATDD will be beneficial for them to achieve their performance gap of on-time delivery and better maintainability.

Understanding the benefits of ATDD	Count
No Impact	11
Will Impact	23
No opinion	16

Table 6: Leaner distribution by knowledge on ATDD values

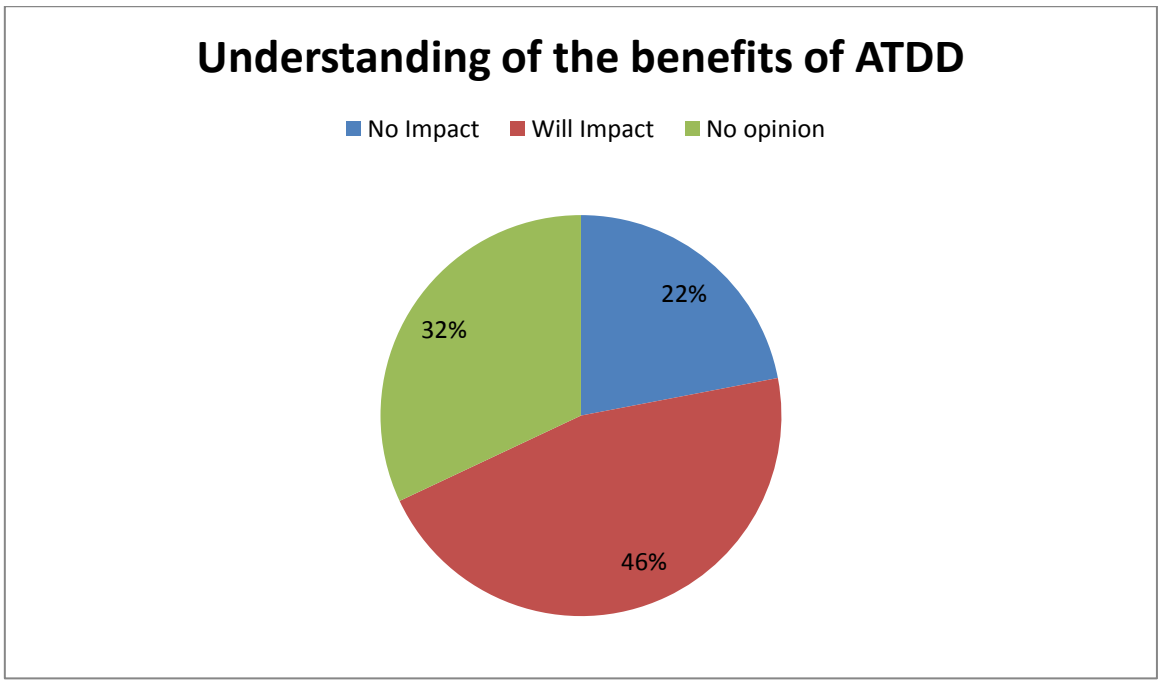


Figure 6: Leaner percentage by knowledge on ATDD values

According to Table 6 and Figure 6, it is shown majority of sample believe ATDD will impact on on-time delivery and maintainability

3.3. LEARNER TASK ANALYSIS

Task analysis is the process of breaking a skill into smaller, more manageable steps in order to teach the skill. As the smaller steps are mastered, the learner becomes increasingly independent in his or her ability to perform the larger skill.

Tasks	**User/Learner Group**	**Importance**	**Difficulty**	**Frequency**
Feature requirement Analysis	PO	High	High	Low
User story writing	PO	High	Moderate	Moderate
User story estimation	DEV/QA	Moderate	Moderate	Moderate
User story prioritization	PO	Moderate	Low	Low
Acceptance criteria creation	PO	High	Moderate	High
Sample data creation	PO/QA	High	Moderate	Moderate
User story review/Grooming	PO/DEV/QA	High	Moderate	Moderate
Code Implementation	DEV	High	High	High
Unit test writing	DEV	Moderate	Moderate	Moderate
Feature testing	QA	High	High	High
Feature test automation	QA	Moderate	High	Moderate
User Acceptance test	PO	High	Moderate	High
Integration testing	QA	High	High	Moderate
Regression	QA	High	High	Moderate
Performance testing	DEV/QA	Moderate	High	Low
Security testing/review	DEV/QA	Moderate	High	Low
Deployment	DEV	High	Moderate	Low

Table 7: Current project development tasks

Legend: PO – Product owner, Dev – Developer, QA – Quality assurance engineer

In Table 7, it is reflected the task that learner group should carried on a project development work and their importance to the project completion with difficulty of performing those task and the frequency. According to the findings Code implementation and Feature testing is most important tasks and those needs to be execute frequently and they are mode difficult task to execute. Even though Unit testing and Test automation not important as Code implementation or Feature testing in terms of actual project completion but those are helpful for the frequent code implementation, Feature testing and Performance testing

3.4. TOPIC ANALYSIS

3.4.1. COURSE OUTLINE

In order to construct a syllabus, various contents has been gathered and analyzed. Content and topic analysis mainly based on resources which were available in the web. Content analyzing has been conduct with the subject matter experts and course outline was created based on the findings. Table 8 represents the course outline details.

Course: Acceptance Test Driven Development	
Unit and Lesson title	**Description**
Unit 1. Introduction to Software Testing	The unit describes overview of software testing principles, Levels of Testing, Testing Process in different software development models and introduction to the agile testing practice and its benefits
Lesson 1.1 – Introduction to Software Testing	The lesson introduce to the Software Testing process of the software development. How it relates to customer experience and business
Lesson 1.2 – Software Testing Principles	The lesson describes 7 software testing principles
Lesson 1.3 – Testing Levels	The lesson describes testing levels in software development process
Lesson 1.4 – Testing Process	The lesson describes several software development models and testing process of each of them
Lesson 1.5 – Traditional Testing practice	The lesson outlines Traditional testing practice and its drawbacks
Lesson 1.6 – Agile Testing practice	The lesson outlines Agile testing practice and its advantages
Unit 2. Transitioning to Agile Software development process	The Unit describes the features of Agile software development process and what needs to be done in order to transition into it from traditional software development
Lesson 2.1 – Introduction to Agile Software development principles and	The lesson describes Agile software development principles and its values.

values	
Lesson 2.2 – Agile Testing Lifecycle	The lesson illustrate Agile Testing Lifecycle
Lesson 2.3 – Test Driven Development	The lesson describes the features of Test Driven Development
Lesson 2.4 – Test Automation	The lesson describes the importance of test automation, suitable and not-suitable test cases for automation and steps in automation process
Unit 3. Writing User stories	The Unit describes concept of User story which represent the specification in agile development. How to write User story with acceptance criteria and definition of done (DOD)
Lesson 3.1 – Introduction to User Story	The lesson describes the features of User story, illustrate the User story card and samples
Lesson 3.2 – Writing Good User Story	The lesson describes how to make good user stories
Lesson 3.3 – Acceptance Criteria	The lesson describes the Acceptance Criteria and its characteristics
Unit 4. Unit Tests	The Unit describes Unit Testing Principles, frameworks, Unit Testing tools and Mocking tools in C# and .NET environment
Lesson 4.1 – Unit Testing Principles	The lesson illustrates the Unit Testing principles and Its importance regards to code quality and test automation.
Lesson 4.2 – Unit Testing Frameworks	The lesson describes Unit test frameworks and its features
Lesson 4.3 – Testing Tools	The lesson describes details into Unit testing tools. particularly MsTest and NUnit, Unit testing framework for .Net Projects
Lesson 4.4 – Mocking and Stubbing	The lesson describes importance of having

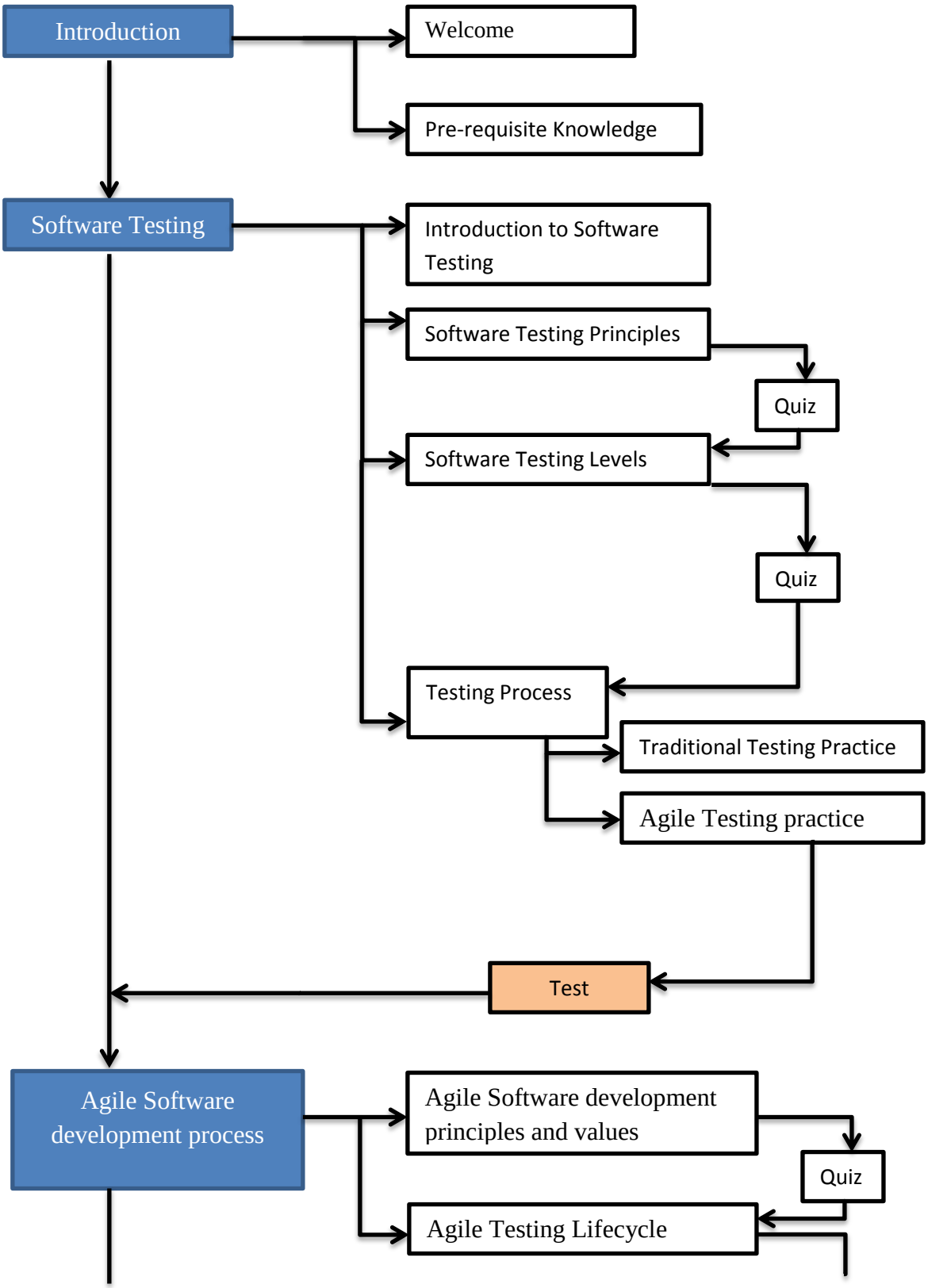
	mocks and stubs in Unit test. And tools for mocks, particularly Moq and NSubstitute, mock tools for .Net Projects
Unit 5. Acceptance Test Driven Development	The unit introduces the Acceptance Test Driven Approach, Difference between Test First and Test Last, Test Driven Development rhythm and refactoring phase of it
Lesson 5.1 – Introduction to ATDD	The lesson describes what ATDD is and how it helps to bridge communication gap
Lesson 5.2 – ATDD Cycle	Describe ATDD cycle of Discuss, Distill, Develop and Demo
Lesson 5.3 – Definition of Done	This lesson describes in-depth details about Definition of Done and how to get it define in Agile setting
Lesson 5.4 - Benefits of ATDD	This lesson points out Benefits of practicing ATDD
Lesson 5.5 - Challenges in ATDD	This lesson points out Challenges in practicing ATDD
Lesson 5.6 - ATDD Tools : Cucumber and Gherkin	This lesson describes Cucumber tool and Gherkin language, Gherkin Syntax
Unit 6. Sample ATDD Project	The unit demonstrates how to start project with ATDD approach, setting up the environment
Lesson 6.1 – Setting up Environment	The lesson demonstrates how to setup the .net environment, visual studio with nUnit and SourceFlow
Lesson 6.2 – Creating a Project	The lesson demonstrates how to create sample project using visual studio
Lesson 6.3 – Creating Specification and Feature for the Acceptance Test	The lesson demonstrates how to create step definition feature file using Gherkin language with SourceFlow
Lesson 6.4 – Add a Failing Test	The lesson demonstrates how to add failing test into the project for a implementation which is yet to be developed

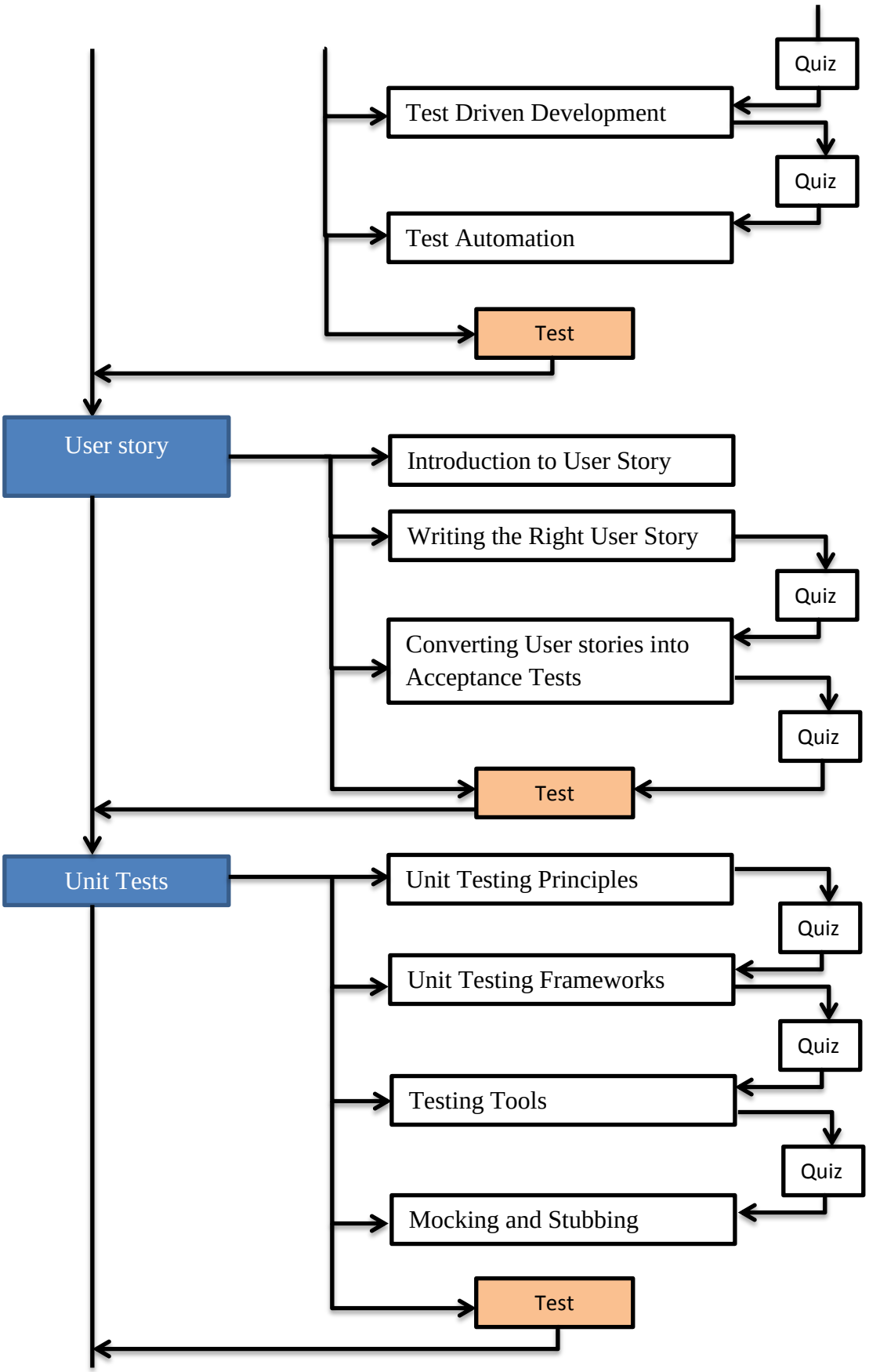
Lesson 6.5 – Implementation to Pass the Test	The lesson demonstrates how to implement the code for the feature which results the Test to be passed.
Lesson 6.6 – Refactor the Code	The lesson demonstrates how to refactor the code with better design and retest for validity
Unit 7. Dealing with Existing Project	The unit demonstrates How ATDD can be applied to an existing Project
Lesson 7.1 – Add Test to Existing Project	The lesson demonstrates how to add unit test for the existing project to test features for it
Lesson 7.2 – Update Test to pass	The lesson demonstrates how to update the test with assertions to pass the test
Lesson 7.3 – Mock the test data	The lesson demonstrates how to use mock object to replace real integration for simplicity and to reduce test execution time

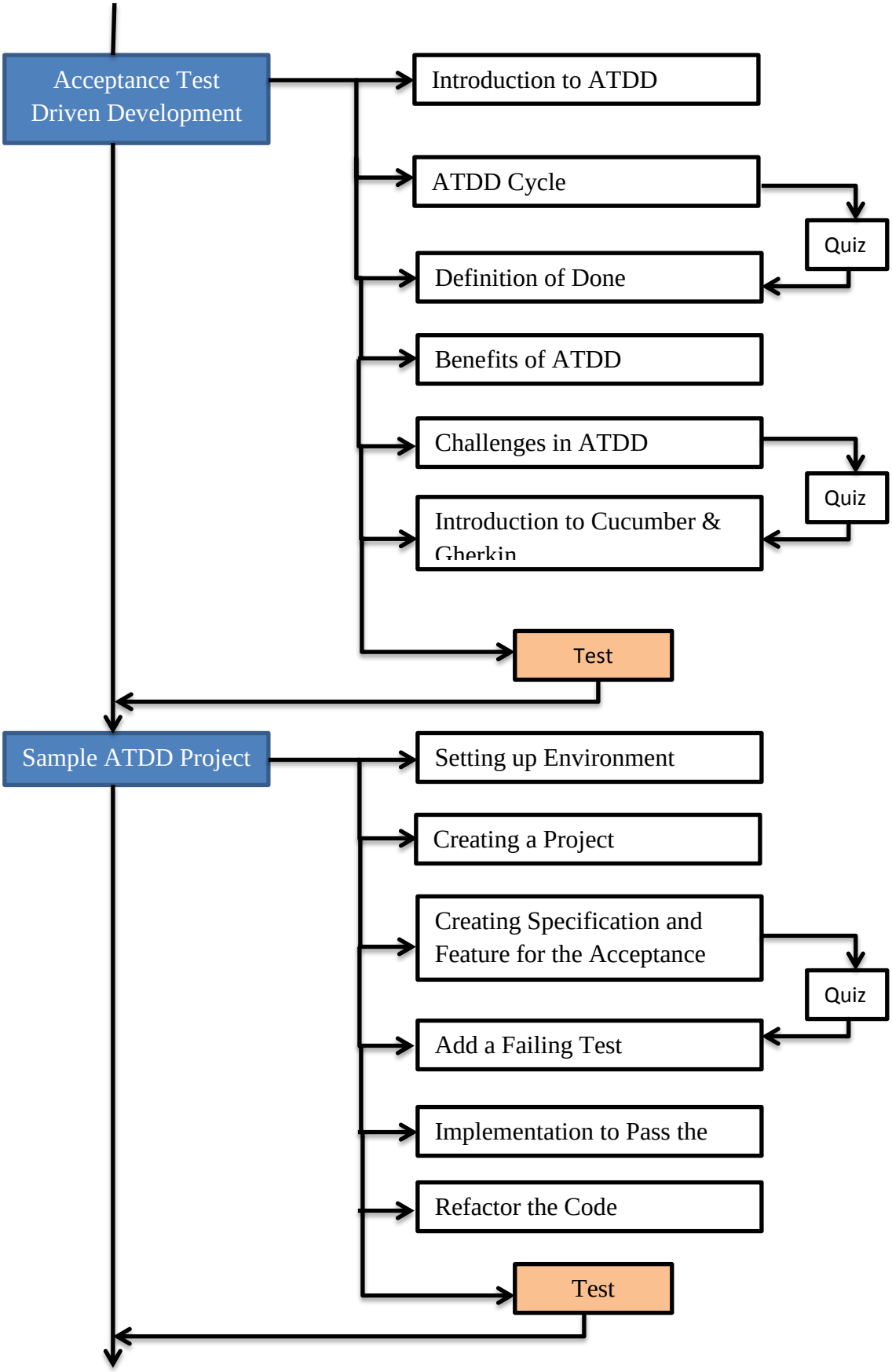
Table 8: Course outline

3.4.2. COURSE MAP

Figure 7 represent the course map created with syllabus. It also indicates the learning path flow between course modules.




Quiz
Test Driven Development
Quiz
Test Automation
Test
User story
Introduction to User Story
Writing the Right User Story
Quiz
Converting User stories into Acceptance Tests
Quiz
Test
Unit Tests
Unit Testing Principles
Quiz
Unit Testing Frameworks
Quiz
Testing Tools
Quiz
Mocking and Stubbing
Test


Acceptance Test Driven Development
Introduction to ATDD
ATDD Cycle
Quiz
Definition of Done
Benefits of ATDD
Challenges in ATDD
Quiz
Introduction to Cucumber & Gherkin
Test
Sample ATDD Project
Setting up Environment
Creating a Project
Creating Specification and Feature for the Acceptance
Quiz
Add a Failing Test
Implementation to Pass the
Refactor the Code
Test

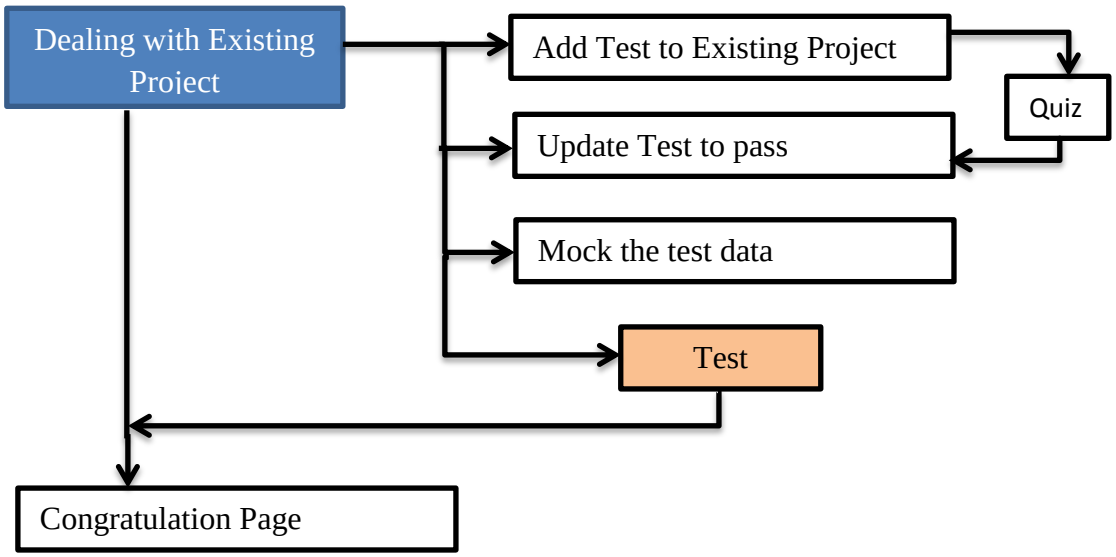


Figure 7: Course map

CHAPTER 4: DESIGN

4.1. COURSE OBJECTIVES

Main goal of delivering Acceptance Test Driven Development course is to improve the quality of the software development and reduce the delivery time by defining clear acceptance criteria and developing clean and testable code.

And it is expected to have following objectives as well

- Learners will be able to define clear acceptance criteria for user story and reduce conflicts between product owners, testers and developers
- Learners will be able to write testable code with test-driven development
- Learners will be able to write effective Unit Tests to validate the code implementation
- Learners will be able to apply mock objects for unit tests
- Learners will be able perform code refactoring effectively
- Learners will be able to create Test Automations.
- Learners will be able to apply new technologies in software development industry
- Learners will be able to reduce code complexity in their projects

4.2. MODULE OBJECTIVES

Unit 1. Introduction to Software Testing
Learning Objective: At the end of this unit, learners should be able to describe the purpose of software testing, different testing levels, processes. And should be able to differentiate agile and traditional practices

Lesson 1.1 – Introduction to Software Testing
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the purpose of software testing • Evaluate significance of finding defects at early stage in terms of cost of fixing
Main Target Audience: Product owners, Testers, Developers

Guidelines for Author: This lesson introduces the Software Testing process of the software development. How it relates to customer experience and business. How the cost of fixing the defect depending on the stage it was found	
Learning Steps	**Scope notes for author**
What is software Testing?	Brief description of what software testing is, importance and purpose of software testing and how quality drives customer satisfaction and business
What are the Stages that defect can be found and how the cost of fixing is depending on the stage	Illustrate the fixing cost and stage dependency

Lesson 1.2 – Software Testing Principles	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe 7 key principles in software testing	
Main Target Audience: Product owners, Testers, Developers	
Guidelines for Author: The lesson describes 7 software testing principles. Testing shows the presence of bugs, Exhaustive testing in impossible, Early testing, Defect clustering, The pesticide paradox, Testing is context dependent, Absence of errors fallacy.	
Learning Steps	**Scope notes for author**
Identifying Main scenario and alternate scenarios of a test case	Illustrate the main scenario of a test case and alternate scenarios of it
Examples for each principle	Illustrate examples for each test principle

Lesson 1.3 – Testing Levels	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the different Testing Levels • Identify Objectives of each Testing Level • Identify execution role of each Testing Level	
Main Target Audience: Testers, Developers	
Guide line for Author: This lesson describes testing levels in software development process. Unit Testing, Integration Testing, System Testing, Acceptance Testing	
Learning Steps	**Scope notes for author**
What are Testing Levels?	Illustrate 4 Testing Levels and Objectives

Lesson 1.4 – Testing Process	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe different software testing processes with respect to different software development models • Identify key features in each software testing process	
Main Target Audience: Testers, Developers	
Guide line for Author: This lesson describes several software development models and testing process of each of them	
Learning Steps	**Scope notes for author**
What are different software development models which used in the industry?	Brief Descriptions for each software development models and its testing process
Identify features of each testing process	Illustrate key features of each testing process and phases of the process

Lesson 1.5 – Traditional Testing practice	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the Traditional Testing practice • Identify the drawbacks of Traditional testing practice	
Main Target Audience: Product owners, Testers, Developers	
Guidelines for Author: This lesson describes Traditional testing practice and its drawbacks	
Learning Steps	**Scope notes for author**
What is Traditional Testing?	Describe Traditional Testing Approach
What are the drawbacks of the Traditional testing?	Explain drawbacks of Traditional Testing Approach with respect to time and cost

Lesson 1.6 – Agile Testing practice	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe Agile Testing practice and features • Differentiate Agile Testing practice vs Traditional Testing practice • Evaluate Advantages of Agile Testing practice	
Main Target Audience: Product owners, Testers, Developers	
Guide line for Author: This lesson outlines Agile testing practice and its advantages.	
Learning Steps	**Scope notes for author**
What are the features of Agile Testing practice?	Describe Agile Testing practice and its features
What are the Advantages of Agile Testing Practice	Illustrate advantages of agile testing practice

Unit 2. Transitioning to Agile Software development process
Learning Objective: At the end of this unit, learners should be able to mapping agile principles and values to testing, describe agile testing life cycle, identify benefits of TDD

Lesson 2.1 – Introduction to Agile Software development principles and values	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe agile development principles • Recognize values delivers by agile development	
Main Target Audience: Product owners, Testers, Developers	
Guide line for Author: This lesson describes Agile software development principles and its values	
Learning Steps	**Scope notes for author**
What is agile manifesto?	Illustrate the agile manifesto and core values
What are agile development principles?	Illustrate agile development principles
How testing should happen in agile development model	Describe Agile testing in agile software development model

Lesson 2.2 – Agile Testing Lifecycle	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe Key points in Agile Testing • Describe 3 Phases in Agile Testing Lifecycle • Identify output of each phase • Describe Agile Testing approaches	
Main Target Audience: Product owners, Testers, Developers	
Guide line for Author: This lesson illustrate Key points in Agile Testing, Phases of Agile Testing Lifecycle and output of each phase	
Learning Steps	**Scope notes for author**
What are Key points in agile Testing	Illustrate key points in agile testing
What are the phases in Agile Testing lifecycle and how it's different from Traditional Testing lifecycle	Illustrate phases in agile testing lifecycle and describe the differences between traditional testing lifecycle
What are the different Agile Testing approaches currently practicing?	Describe different agile testing approaches. Along with development approach and Sprint + 1 Approach

Lesson 2.3 – Test Driven Development	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the features of Test Driven Development • Describe the benefits of Test Driven Development	
Main Target Audience: Product owners, Testers, Developers	
Guide line for Author: This lesson describes features of Test Driven Development	
Learning Steps	**Scope notes for author**
What is Test Driven Development?	Describe features of Test Driven Development
Why Test Driven Development? What are the benefits?	Illustrate benefits of Test Driven Development

Lesson 2.4 – Test Automation	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the goals of Automation • Describe why automated testing is important • Identify suitable test cases for Automation • Identify non-suitable test cases for Automation • Describe steps in Automation process	
Main Target Audience: Testers	
Guide line for Author: This lesson describes the importance of test automation, suitable and not-suitable test cases for automation and steps in automation process	
Learning Steps	**Scope notes for author**
Why Test Automation?	Describe goals of Test automation and why test automation is needed in software development
What to Automate?	Illustrate the criteria of suitable test cases for automation and criteria of non-suitable test cases
How to Automate?	Illustrate steps in automation process

Unit 3. Writing User stories
Learning Objective: At the end of this unit, learners should be able to describe components of user story, differentiate bad user story from a good user story, demonstrate how to write proper user story

Lesson 3.1 – Introduction to User Story
Learning Objectives: At the end of this lesson, learners should be able to: • Identify components of a user story • Describe how user story fits into requirement specification
Main Target Audience:

Product owners, Testers, Developers	
Guide line for Author: This lesson describes the features of User story, illustrate the User story card and samples	
Learning Steps	**Scope notes for author**
What is User story?	Illustrate components of a user story
How User story different form requirements	Illustrate how and why user story is different from requirement specification
Lesson 3.2 – Writing Good User Story	
Learning Objectives: At the end of this lesson, learners should be able to: • Identify characteristics of a good user story • Demonstrate writing a good user story	
Main Target Audience: Product owners	
Guide line for Author: This lesson reviews the examples of bad user stories and describes how to write good user stories with clear acceptance criteria	
Learning Steps	**Scope notes for author**
What are the characteristics of a good user story	Illustrate components of a good user story
How to write a good user story	Demonstrate examples of writing clear user stories

Lesson 3.3 –Acceptance Criteria	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe structure of acceptance criteria	
Main Target Audience: Product owners, Testers	
Guide line for Author: This lesson describes structure of Acceptance criteria, introduce to the Gherkin language and describes how to write user stories in feature files using Gherkin language with Give/When/Then Statements	
Learning Steps	**Scope notes for author**
What is Acceptance Testing?	Describe Acceptance Testing and how to derive acceptance criteria from a user story
How to write an Acceptance Criteria?	Illustrate writing sample acceptance criteria using Gherkin language

Unit 4. Unit Tests
Learning Objective: At the end of this unit, learners should be able to describe unit testing principles, how unit test improves quality of the software, recognize the unit testing frameworks and tools, demonstrate writing unit test

Lesson 4.1 – Unit Testing Principles	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe Unit Testing principles • Recognize importance of unit test with respect to code quality	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson illustrates the Unit Testing principles and Its importance regards to code quality and test automation	
Learning Steps	**Scope notes for author**
What is Unit Testing?	Describe definition of unit Testing
Why Unit Testing?	Describe the importance of writing unit test
What are the Unit testing principles?	Describe Unit testing principles

Lesson 4.2 – Unit Testing Frameworks	
Learning Objectives: At the end of this lesson, learners should be able to: • Recognize Unit Testing Frameworks of Visual Studio	
Main Target Audience: Developers	
Guide line for Author: This lesson describes Unit test frameworks and its features	
Learning Steps	**Scope notes for author**
What are the features of Unit test framework in VS?	Demonstrate Unit test framework using Visual Studio

Lesson 4.3 – Testing Tools	
Learning Objectives: At the end of this lesson, learners should be able to: • Recognize Unit testing tools for .NET projects and Visual studio • Demonstrate installing unit testing tools	
Main Target Audience: Developers	
Guide line for Author: This lesson describes details into Unit testing tools. particularly MsTest and NUnit, Unit testing framework for .Net Projects	
Learning Steps	**Scope notes for author**
What are available Unit testing tools?	Describe Unit testing tools available for .NET environment
How to install the Unit testing tools	Demonstrate installing NUnit using Visual Studio

Lesson 4.4 – Mocking and Stubbing	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe usage of having Mocks and Stubs in Unit Tests • Demonstrate installing and configuring Mocking tools with .NET environment	
Main Target Audience: Developers	
Guide line for Author: This lesson describes importance of having mocks and stubs in Unit test. And tools for mocks, particularly Moq and NSubstitute, mock tools for .Net Projects	
Learning Steps	**Scope notes for author**
What is Mock and Stub	Describe the Mocks and Stubs and its role in Unit test
How to install Mock tools?	Demonstrate installing Moq into Visual Studio

Unit 5. Acceptance Test Driven Development
Learning Objective: At the end of this unit, learners should be able to describe what are acceptance tests, who creates acceptance test, when created, where to used, why use them

Lesson 5.1 – Introduction to ATDD	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe how ATDD helps to bridge the communication Gap • Recognize the value of ATDD	
Main Target Audience: Product owners, Testers, Developers	
Guide line for Author:	
Learning Steps	**Scope notes for author**
What is ATDD	Introduce the ATDD approach

Lesson 5.2 – ATDD Lifecycle	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe ATDD cycle	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson illustrates the Acceptance Test Driven Development cycle	
Learning Steps	**Scope notes for author**
ATDD cycle Discuss, Distill, Develop, Demo	Describe ATDD cycle of Discuss, Distill, Develop and Demo

Lesson 5.3 – Definition of Done	
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the purpose of definition of Done • Apply step of clarify the definition of Done for User story, Iteration and Release	
Main Target Audience: Developers	
Guide line for Author: This lesson describes in-depth details about Definition of Done and how to get it define in Agile setting	
Learning Steps	**Scope notes for author**
What is DoD	Describe briefly DoD
What isn't DoD	Describe deference between DoD and Acceptance Criteria

Lesson 5.4 – Benefits of ATDD
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the Benefits of ATDD
Main Target Audience: Developers
Guide line for Author: This lesson points out Benefits of practicing ATDD

Lesson 5.5 – Challenges in ATDD
Learning Objectives: At the end of this lesson, learners should be able to: • Describe the Challenges in ATDD
Main Target Audience: Developers
Guide line for Author: This lesson points out Challenges in practicing ATDD

Lesson 5.6 – ATDD Tools : Cucumber and Gherkin	
Learning Objectives: At the end of this lesson, learners should be able to: • Identify the Gherkin Syntax • Write feature files using Gherkin	
Main Target Audience: Developers	
Guide line for Author: This lesson describes Cucumber tool and Gherkin language, Gherkin Syntax	
Learning Steps	**Scope notes for author**
Identifying the feature file structure	
Write feature file using Gherkin for a given user story	

Unit 6. Sample ATDD Project
Learning Objective: At the end of this unit, learners should be able to demonstrate setup a fresh project using ATDD approach

Lesson 6.1 – Setting up Environment	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate setting up .net environment	
Main Target Audience: Developers	
Guide line for Author: This lesson demonstrates how to setup the .net environment, visual studio with nUnit and SourceFlow	
Learning Steps	**Scope notes for author**
How to install nUnit?	Demonstrate how to install nUnit using Visual studio and NuGet Package Manager
How to Install SourceFlow	Demonstrate how to install SourceFlow using Visual studio and NuGet Package Manager

Lesson 6.2 – Creating a Project	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate how to create a fresh Project	
Main Target Audience: Developers	
Guide line for Author: This lesson demonstrates how to create sample project using visual studio	
Learning Steps	**Scope notes for author**
How to Create a .net Project?	Demonstrate the steps to create .NET project using Visual Studio

Lesson 6.3 – Creating Specification and Feature for the Acceptance Test	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate How to create Acceptance Criteria	
Main Target Audience: Product owners, Developers, Testers	
Guide line for Author: This lesson demonstrates how to create step definition feature file using Gherkin language with SourceFlow	
Learning Steps	**Scope notes for author**
What is feature file?	Describe feature file in SpecFlow
How to create feature file?	Demonstrate creating feature file for acceptance test using SpecFlow
How to write Acceptance Test into	Demonstrate writing Acceptance Test using Gherkin language

feature file?	

Lesson 6.4 – Add a Failing Test	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate how to create failing test file from feature file	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson demonstrates how to add failing test into the project for a implementation which is yet to be developed	
Learning Steps	**Scope notes for author**
How to create failing Test file?	Demonstrate how to generate code using acceptance scenarios in feature files
How to run the Test?	Demonstrate test execution which will fail the test initially

Lesson 6.5 – Implementation to Pass the Test	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate steps needed to pass the acceptance test	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson demonstrates how to implement the code for the feature which results the Test to be passed.	
Learning Steps	**Scope notes for author**
How to pass the test?	Demonstrate code implementation which will eventually pass the Acceptance Test

Lesson 6.6 – Refactor the Code	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate code refactoring and retesting the acceptance test	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson demonstrates how to refactor the code with better design and retest for validity	
Learning Steps	**Scope notes for author**
How to refactor the code?	Demonstrate code refactoring, retesting and validating the result

Unit 7. Dealing with Existing Project
Learning Objective: At the end of this unit, learners should be able to demonstrate ATDD implementation to an existing project

Lesson 7.1 – Add Test to Existing Project	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate how to add test for existing project	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson demonstrates how to add unit test for the existing project to test features for it	
Learning Steps	**Scope notes for author**
How to add Test for existing project	Demonstrate steps to add test of existing Project

Lesson 7.2 – Update Test to pass	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate Test modification to pass the Test	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson demonstrates how to update the test with assertions to pass the test	
Learning Steps	**Scope notes for author**
How to update the test in order to pass	Demonstrate how to update the test for pass the test execution

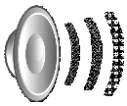
Lesson 7.3 – Mock the test data	
Learning Objectives: At the end of this lesson, learners should be able to: ▪ Demonstrate using mock object to replace real object or service	
Main Target Audience: Developers, Testers	
Guide line for Author: This lesson demonstrates how to use mock object to replace real integration for simplicity and to reduce test execution time	
Learning Steps	**Scope notes for author**
How to use mock object?	Demonstrate using mock object to replace real database integration

4.3. INSTRUCTIONAL STRATEGY

4.3.1. OVERALL STORYBOARDS FOR COURSE: ACCEPTANCE TEST DRIVEN DEVELOPMENT

This section represents the overall storyboard of a selected module of ATDD course. Complete overall storyboard for the all modules can be found in "Appendix A". The selected overall storyboard represent learning content types such as text, multimedia, interactive content, activities and assessments include in a particular course module.

Visual	Explanation
Introduction to ATDD Objectives: <Learning objectives of this module> Structure of the Module: <Topic List> Image	This page describes objectives of this module, and topics covered by the module <Topic List> What is ATDD Difference between ATDD and TDD Process of the ATDD Details of ATDD stages Benefits and challenges in ATDD Image: shows ATDD process figure
What is ATDD Definition of ATDD Image	This page describes the definition of ATDD. Image: shows figure of transforming TDD to ATDD TDD -> ATDD
ATDD vs TDD TDD Focus ATDD Focus Image	This page describes TDD focus area and difference between ATDD focus. Image shows ATDD process and TDD in its core

ATDD Process Animation	This page contains an Animation of ATDD process stages of Discuss, Distill, Develop, Demo When user hover over any stage, explanation will be provided for that stage.
Activity (Drag & Drop)	This page contains a Drag & Drop Activity in order to identify learners' knowledge on ATDD process. Collections of Activities given to the user. User must select correct activities related to ATDD process and should match into correct stage Activities: Create User case diagrams (not related) User Story Unit Test Coding Requirement Analysis Performance Testing (not related) Architecture reviews (not related) …
Definition of Done Image 	This page describes What DOD is. And the importance of having clear DoD Image: show questions to ask in order to get clear DoD How will user use the solution? (examples) How we can demonstrate it? How will we test it? Audio: Clip will play DoD and each of above questions to get to the DoD
Benefits of ATDD For Business Team: For Developers: For Testers: Image	

Challenges of ATDD Cultural Challenge: Slicing Requirements: Defining DoD:	
Quiz MCQ	Based on DoD, Benefits of ATDD and Challenges
Introduction to Gherkin Image	
Gherkin Syntax Feature: User Registration Check for home page See of the registration is working Also verify if the register user is displayed Background: Given: Clear already created user before begin Scenario: Register user with minimal password combination Given I've opened the website And I'm in the homepage When I click the register link Then I should see the register page And I fill the form with details |user name | password | cPassword| |lahiru | abc@123 | abc@123 |	
Few Examples:	
Test	Test is based on use cases. Learner will be give use cases in real software requirements and asked to provide Acceptance Test scenarios using Gherkin Syntax

4.3.2. DETAILED STORYBOARD FOR COURSE: ACCEPTANCE TEST DRIVEN DEVELOPMENT

This section includes the limited storyboard from selected course modules of the ATDD course. Full detailed storyboard of all modules can be found in the Appendix B. Selected storyboards represent various content format that are ranging from text based content to multimedia content to interactive content. It is also included the activities and assessments.

Course Name: **Module 1: Software testing**		Storyboard File no. 01.01.01.00
Course section: 1.1		
Lesson Name: **Introduction To Software testing**		ID's name:
Objective(s):		SME's name:
Page Title: Introduction To Software testing	Page no. 01.01.01.00	CD's name:
Date Designed:	Date SME contributed:	Date verified:
Design		

Introduction To Software testing

Lesson Structure

- What is Software testing and why it's important?
- Software testing principles
- Software testing levels
- Testing processes
- Traditional software testing
- Agile software testing

Objectives

- Describe the purpose of software testing
- describe different testing levels and processes

Special Comment(s):

Figure 8: Storyboard of an introduction content page

As shown in the Figure 8, introduction page for a typical module contains text and relevant graphics to gives a fast impression about the course module. This screens contains introductory audio narrations to explain the module content and structure

Page Title: Software Testing Principles	Page no. 01.02.01.00	SME's name:
Date Designed:		Date verified:
Design		

Special Comment(s):
This is an interactive animation

User should be able to click any item to get more details about the principle. Please refer next screen (01.02.01.01) for detail view

Hand icon should be flashing to indicate "click" here for more action.

Figure 9: Storyboard of interactive content

Figure 9 represents the storyboard of an interactive content where learner can click and get more details. Learner will see the summary of the content first, then he can have detailed information of each items.

Page Title: Writing Good User Stories	Page no. 03.02.01.00	SME's name:
Date Designed:		Date verified:
Design		

Writing Good User Stories

Special Comment(s):
This is a video

"TOP 10 Tips" should be displayed in a "Red" (# FE4040) circle. Until the narration is on this display should be there. Refer next screen and script (05.03.01.01) afterwards.

Transcript:

"User stories are probably the most popular agile technique to capture product functionality.

Working with user stories is easy.
But telling effective stories can be hard.

The following ten tips help you create good stories.

So, let's look at them individually”

Page Title: Writing Good User Stories	Page no. 03.02.01.01	SME’s name:
Date Designed:		Date verified:

Design

Special Comment(s):

As narration goes “Number 1”, #1 should appear in the middle of the Red Circle. And when narration goes “Users Come First”, world should appear from the bottom of the circle and slowly come to the bottom of the #1 as narration goes. When the narration goes “Number 2”, all the remaining text should be cleared and “#2” should appear in the middle of the Red circle and repeat the same behavior for the rest of the items

Transcript:

Number 1: Users Come First

As its name suggests, a user story describes how a customer or user employs the product; it is

written from the user's perspective. What's more, user stories are particularly helpful to capture a specific functionality, such as, searching for a product or making a booking

If you don't know who the users and customers are and why they would want to use the product, then you should not write any user stories. Carry out the necessary user research first, for example, by observing and interviewing users. Otherwise, you take the risk of writing speculative stories that are based on beliefs and ideas—but not on data and evidence.

Number 2: Use Personas to Discover the Right Stories

A great technique to capture your insights about the users and customers is working with personas. Personas are fictional characters that are based on first-hand knowledge of the target group. They usually consist of a name and a picture; relevant characteristics, behaviors, and attitudes; and a goal. The goal is the benefit the persona wants to achieve, or the problem the character wants to see solved by using the product.

But there is more to it: The persona goals help you discover the right stories: Ask yourself what functionality the product should provide to meet the goals of the personas.

Number 3: Create Stories Collaboratively

A user story is not a specification, but a communication and collaboration tool. Stories should never be handed off to a development team. Instead, they should be embedded in a conversation: The product owner and the team should discuss the stories together.

You can take this further and write stories collaboratively, for instance, as part of your product backlog grooming process. This leverages the creativity and the knowledge of the team and results in better user stories.

Number 4: Keep your Stories Simple and Short

Write your stories so that they are easy to understand. Keep them simple and short. Avoid confusing and ambiguous terms, and use active voice. Focus on what's important, and leave out the rest.

Number 5: Start with Epics
An epic is a big, vague story. It is typically broken into several user stories over time leveraging the user feedback on early prototypes and product increments. You can think of it as a headline and placeholder for more detailed stories.

Starting with epics allows you to sketch the product functionality without committing to the details. This is particularly helpful for describing new products and features: It allows you to capture the rough scope, and it buys you time to learn more about how to best address the needs of the users. It also reduces the time and effort required to integrate new insights. If you many detailed stories in the product backlog, then it's often tricky and time-consuming to relate feedback to the appropriate stories and you have to be careful not to introduce inconsistencies.

Number 6: Refine the Stories until They are Ready

Break your epics into smaller, detailed stories until they are ready: clear, feasible, and testable. All development team members should have a shared understanding of the story's meaning; the story should not too big and comfortably fir into a sprint, and there has to be an effective way to

determine if the story is done.

Number 7: Add Acceptance Criteria

As you break epics into smaller stories, remember to add acceptance criteria. Acceptance criteria complement the narrative: They allow you to describe the conditions that have to be fulfilled so that the story is done. The criteria improve the story, they make it testable, and they ensure that the story can be demoed or released to the users and other stakeholders. As a rule of thumb, use three to five acceptance criteria for detailed stories.

Number 8: Use Paper Cards

User stories emerged in Extreme Programming, and the early XP literature talks about story cards rather than user stories. There is a simple reason: User stories were captured on paper cards. This approach provides three benefits: First, paper cards are cheap and easy to use. Second, they facilitate collaboration: Everyone can take a card and write down an idea. Third, cards can be easily grouped on the table or wall to check for consistency and completeness and to visualize dependencies. Even if your stories are stored electronically, it is worthwhile to use paper cards when you write new stories.

Number 9: Keep your Stories Visible and Accessible

Stories want to communicate information. Therefore don't hide them on a network drive. Make them visible, for instance, by putting them up on the wall. This collaboration, creates transparency, and makes it obvious when you add too many stories too quickly, as you quickly start running out of wall space.

Number 10: Don't Solely Rely on User Stories

Creating a great user experience requires more than user stories. User stories are helpful to capture product functionality, but they are not well suited to describe the user journeys and the visual design. Therefore complement user stories with other techniques, such as, story maps, workflow diagrams, storyboards, sketches, and mockups.

Additionally, user stories are not good capturing technical requirements. If you need to communicate what an architectural element like a component or service should do, then write technical stories or use a modeling language like UML.

Finally, writing user stories is worthwhile when you develop software that's likely to be reused. But if you want to quickly create a throwaway prototype or mockup to validate an idea, then writing stories may not be necessary. Remember: User stories are not about documenting requirements; they want to enable you to move fast and develop software as quickly as possible and not to impose any overhead.

Figure 10: Storyboard of a video content

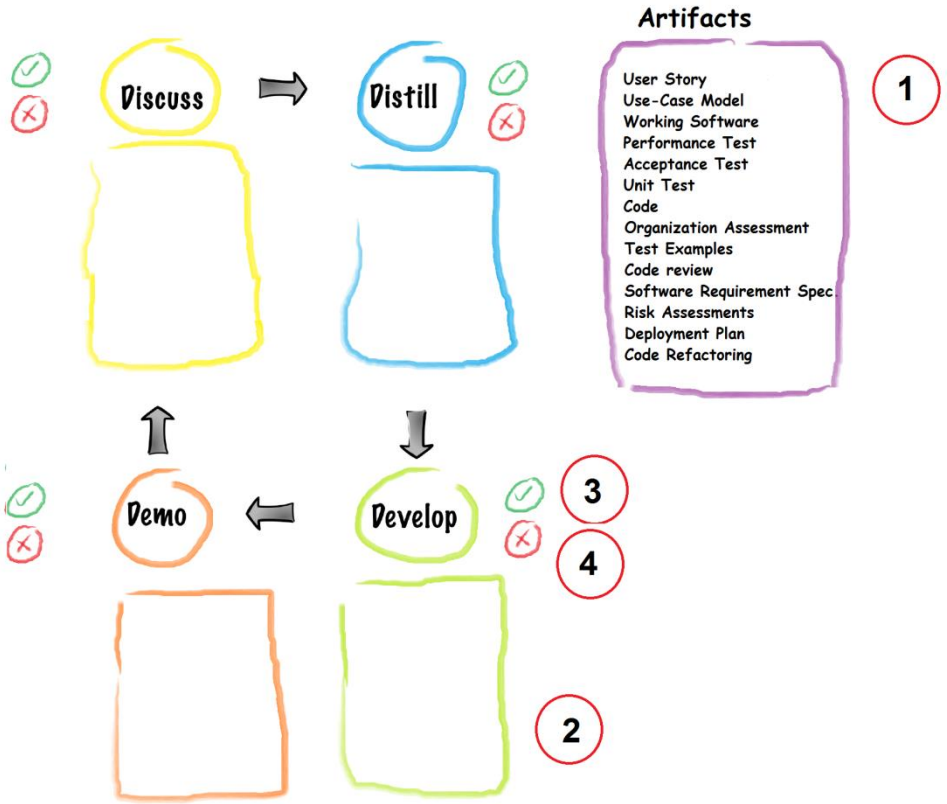
Figure 10 represents a storyboard of a video content. In the storyboard it's describe about the features and narration text. Giving the design details will be helpful for content designers to build the content as intended by the instructional designer.

Page Title:	Page no. 05.03.01.00	SME's name:
Date Designed:		Date verified:

Design

ATDD Process Cycle

Activity: Map artifacts into correct stage.
You need to drag and drop activities from "Artifacts" Box into correct ATDD process. If an artifact does not related to ATDD process, place them in "Not Related to ATDD" Box.



Special Comment(s):

User can select and drag and drop artifacts for "Artifacts" Box into the correct cage to get marks

Should be able to select "Artifacts" from 1

Should be able to Drag selected artifacts into chose cage 2

If the Artifact is not belong the Drop cage, it should go back to the "Artifacts Box" and

Indicate "Incorrect Move" and Should increase the Wrong count 4 by 1

If the Artifact is placed in correct cage, Right count 3 should increase by 1

Figure 11: Storyboard of a drag and drop activity

Figure 11 represents a storyboard of a drag and drop activity. In the comment section, instructional design has given with intendent behavior. These instructions will be helpful for content developers to have an understanding about the requirement.

Page Title:	Page no. 05.08.01.00	SME's name:
Date Designed:		Date verified:
Design		
Assignment: (10 Marks) Due date: 2016-08-10 No submission after: 2016-08-17 Late penalty: 40% will be reduced from the marks for late submissions Look at the given user story and convert it into .feature file using Gherkin syntax. Upload the .feature file using "Choose file" button and submit the assignment. As a shop visitor I want to collect books in my shopping basket So that I can purchase multiple books at once. Books can be added to the shopping basket Books can be removed from the shopping basket Shopping basket is initially empty The same book can be added multiple times to the shopping basket File: Choose File no file selected Cancel Submit Assignment		
Special Comment(s):		

Figure 12: Storyboard of an assignment

Figure 12 represent the typical assignment which will be given to the learner in order to evaluate the learning.

CHAPTER 5: DEVELOPMENT

5.1. CONTENT DEVELOPMENT

Content Development has been started after finalizing the detailed storyboard with Stakeholders. Subject matter experts and Instructional designer verified that lesson content is aligned with learning objectives. Reviews also have been taken to verify assessment tests and exercises aligned with lesson objectives at every step in the lesson flow. Content of every lesson has written in simple and clear wording and keeping the sentences short. Bullet points were used whenever appropriate in order to make the content clearer to the learner. If there's an acronyms used in the content, it was read in full the first time. Personal pronounce (e.g. "you") has been used to refer to learners in order to personalize the instructions to the learner.

5.1.1. STRUCTURE OF LESSONS

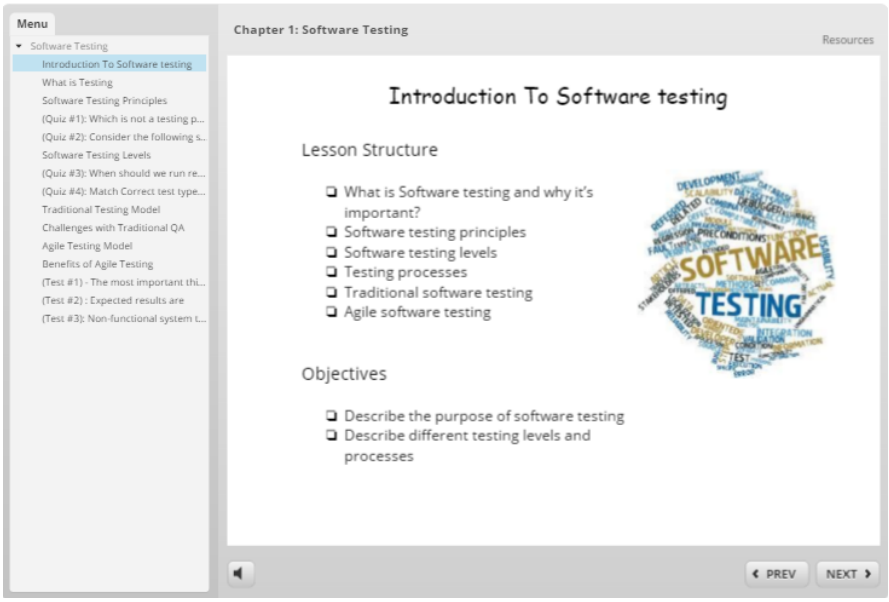
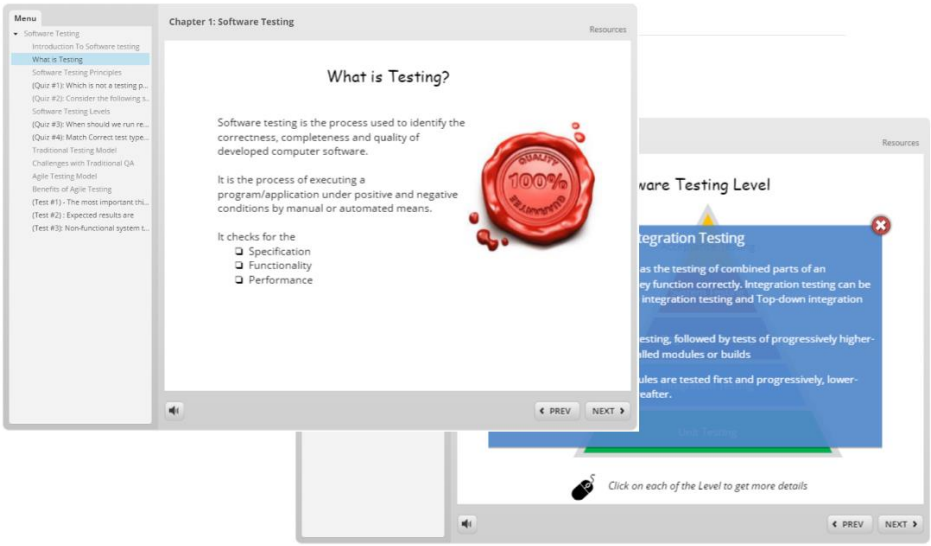
Common lesson structure was used in every lesson in order to maintain the standard in the lesson flow.

E-lesson structure:

Introduction > Content > Test > Result

Introduction is limited to one screen and it also describes the learning objectives as well. The Introduction screen is audio narrated. Next screens, typically limited to 20-25 screens are from lesson topic of the module. Lesson content is mixed with quizzes to give formative feedback to the learner and keep the learner engaged with the content. At the end of each lesson learner will get test and learner will be presented with the test results. Summary of the lessons is displayed at the end of each lesson to summarize what learner has learned through the lesson.

Table 9 represents the major lesson component details. In the current course follows standard structure through the different course modules in order to have unique look and feel. These major components can be categories as introduction page or learning objective page, main content pages, test or quizzes pages and result display page.

Lesson Component	Example
1) Learning Objectives First screen contains the lesson structure and clear and informal description of learning objectives	
2) Content (core of the lesson) A set of screens (from 4 to 25) which make up the core of the lesson. These combine: ▪ text ▪ media elements ▪ examples ▪ Practice questions (quizzes)	

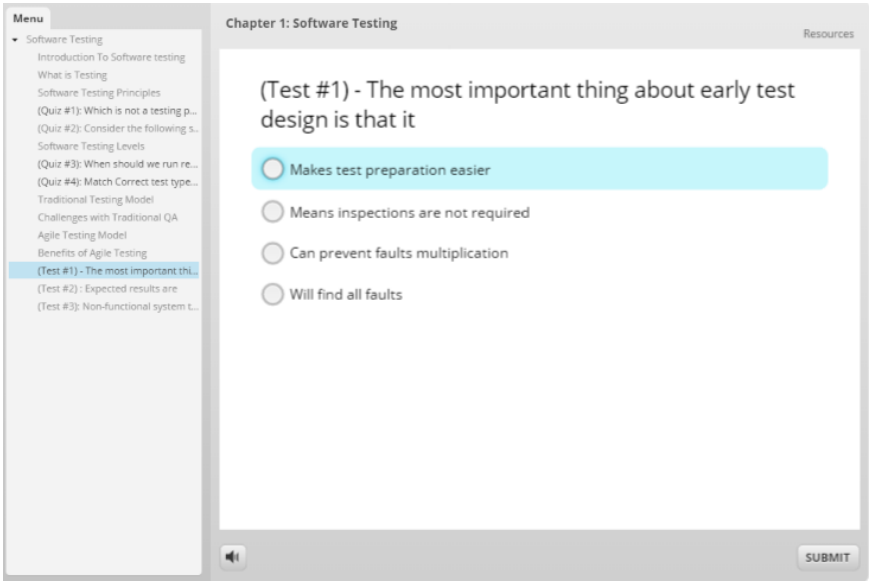
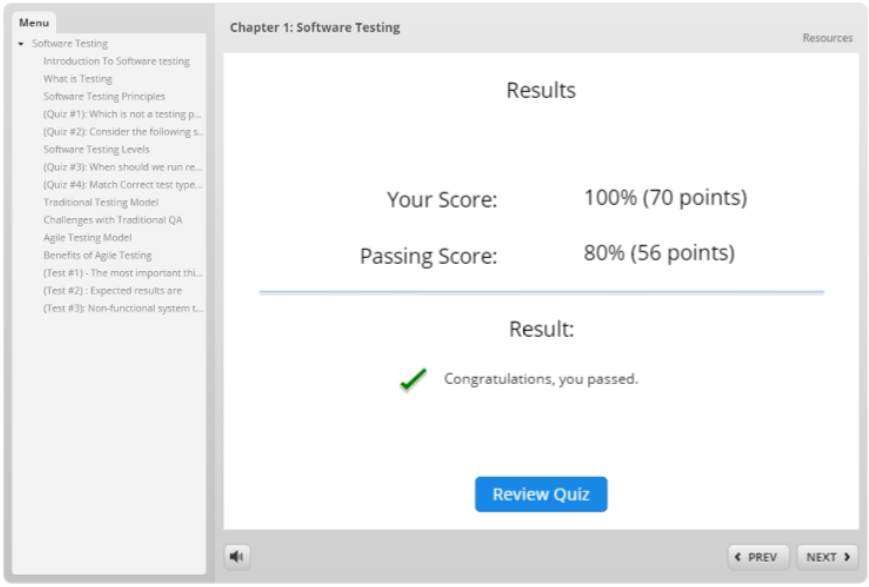
3) Test Test is used as summative assessment to check the learner's knowledge of the learnings. Test format is very from each lesson. Results of the test will be show at the end of the test	
4) Result Result page will show the learner's pass/fail status and give review option to view and compare the correct responses.	

Table 9: Lesson components

5.1.2. INTEGRATING MEDIA ELEMENTS

There are a number of different kinds of media elements have been combined to create e-lessons. Media elements have been used carefully not to overload learner's working memory.

Media Element: Text

Written text is an important "media" for communicating course content. Great attention has been given to its graphic display and integration with images.

The following principles were applied when displaying text on a lesson screen:

- Display on-screen text to provide the best readability and clarity.
- Use graphic conventions consistently; for example, italic style has always been used for the same purpose.
- Use lists or tables used to organize the information.
- Use list points or blank spaces to separate items in a list or focus the attention on them.
- Considered word and row spacing to improve text readability.

Media elements: Graphics

Graphics include illustrations, pictures, diagrams and icons. Graphics has been used for different communication functions, including the following:

- decorative: to add aesthetic appeal
- representational: to represent an object in a realistic fashion
- mnemonic: to provide retrieval indications for factual information
- organizational: to show qualitative relationships among content
- relational: to show quantitative relationships among two or more variables
- transformational: to show changes in objects over time or space
- Interpretive: to illustrate a theory, principle or cause-and-effect relationships.

Graphics has been used in order to promoting learning. It has not only been used to add visual interest to a screen. In e-learning, relevant graphics has facilitated learning by:

- drawing attention to a specific content element
- suggesting analogies between new content and familiar knowledge
- supporting the understanding of concepts
- simulating the work environment and real situations

- motivating learners by making materials more interesting

Table 10 represents few examples of graphics serve some of the communication functions list.

Example: Graphics with representational function	
Graphics used to illustrate the concepts	
Graphics used to describe the lifecycle of agile testing	
Graphic is used to grab the attention to the	

content	

Table 10: Use of multimedia in lessons

Media elements: Animations

Animated illustrations and interactions were used for series of procedural steps or transformations.

Animations were used to

- Allow learners to focus on only one object at a time.
- Use arrows to steer attention to selected details or motion direction.
- Segment long or complex animations and allow learners to access each chunk at their own pace rather than playing all the steps continuously
- Limit the use of animation effects on text because they do not have any instructional function and can irritate learners.

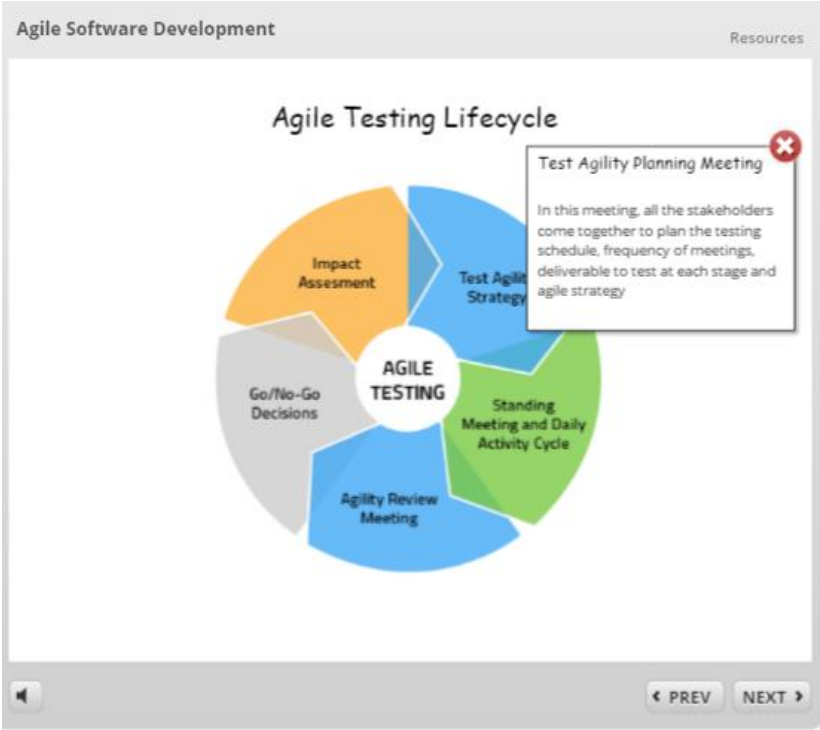
Examples: use of animation to illustrate more details of the concepts	
Display details of a stage in lifecycle	
Display details of software testing level	

Table 11: Usage of animation

Table 11 represents the various content pages that includes the animation in their lesson pages

Media elements: Audio

Audio narrations have been used whenever appropriate because it's greatly increases the effectiveness of a course. Audio is used in combination with on-screen text to summarize or expand key points because audio narration is more effective than printed text when providing comments on animations sequences or a series of static frames showing a transformation. Option has been given to the learner to silence the audio, because learners' visual channel can become overloaded with audio narrations if they have to process graphics and the printed words that refer to them.

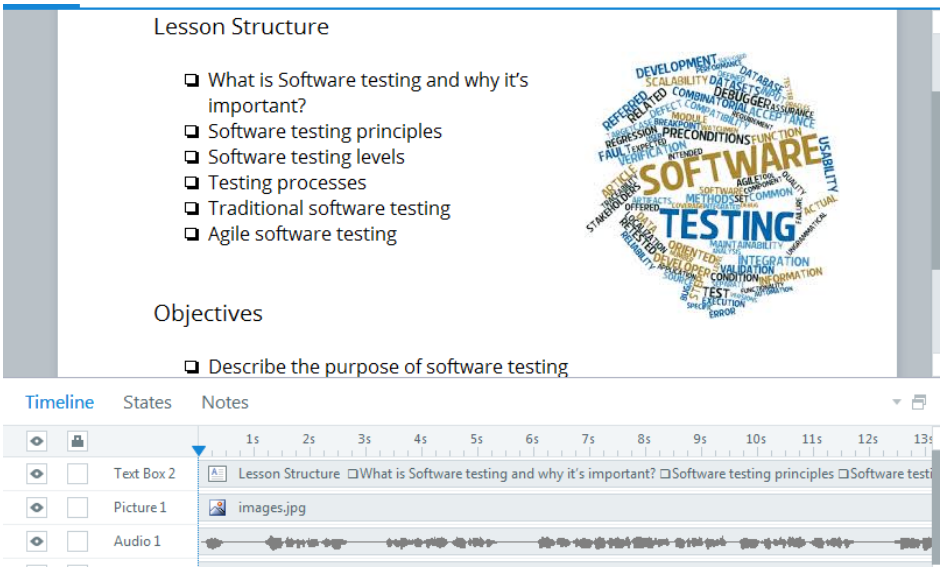
Examples of using audio narrations	
Audio narration has been embedded into introduction of lessons	
Audio narrations embedded into interactions	

Table 12: Usage of audio

Table 12 represents various content pages which use audio and narrations in their content.

Media elements: Video

Video is the only media that makes it possible to reproduce behavior, processes or procedures the way they appear in real life. It has been used for demonstration of installations and configuration of ATDD related software components and how the software use in real work setting.

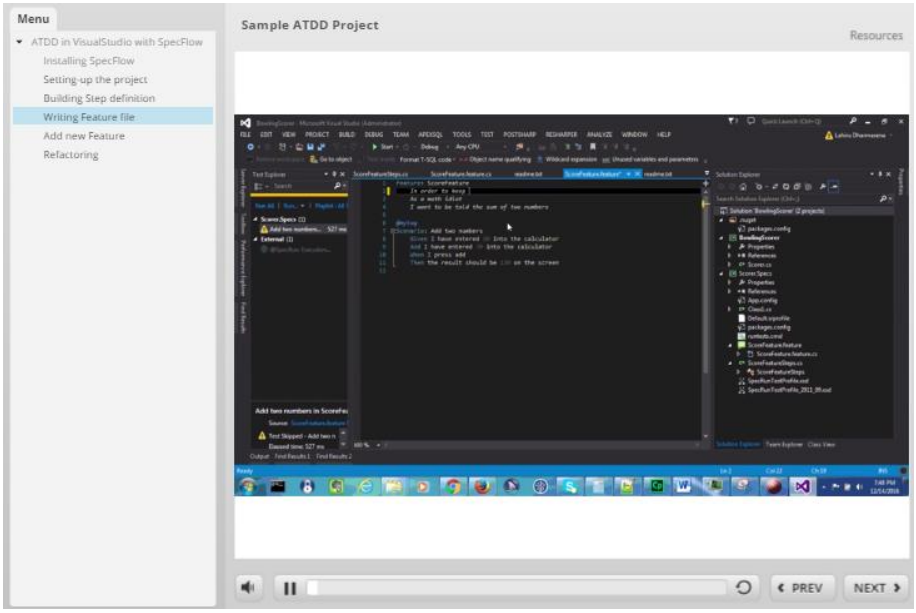
Example of using video in lesson	
Demonstration of creating specFlow feature file with Microsoft VisualStudio	

Table 13: Usage of video

Table 13 represents content pages which uses video in their content such as demonstrational videos.

5.2. COURSEWARE DEVELOPMENT

This chapter provides information on the last step of the development stage, which is the creation of the final interactive courseware. The chapter will illustrate work done using Authoring tools for producing e-learning courseware

5.2.1. SELECTING THE AUTHORING TOOL

Even though there is no right and wrong authoring tool for developing the courseware, few factors were considered selecting correct authoring tool for this project. Those are basically,

- Editing/updating capabilities – ability to do rapid editing through a content publisher. Fast editing and easier updating is time efficient.
- Delivery outputs
    - LMS – Course will be deployed on a learning management system. This requires courseware to comply with SCORM technical standards
    - Web browser - interoperability has been considered
- Learning curve –Amount of time needed to learn how to use the tool should be minimal.
- Training opportunities – Should be able to learn about the tool through online guides, webinars, online support and forums.
- Integration – Ability to integrates well with leading LMS or/and other software
- Creative freedom –Ability to express and accommodate interactions, navigation elements, quizzes and other features into course design.
- Industry and community support – room for get support is essential for troubleshooting, problem solving and getting useful tips. It has to be widely used tools are better supported by online forums and user groups

By considering above factor Articulate Storyline has been chosen as the primary authoring tool. For screen demonstration capturing is done by Microsoft Screen Recorder and video/audio editing is done using Adobe premiere and Adobe Audition.

Example: Articulate Slide

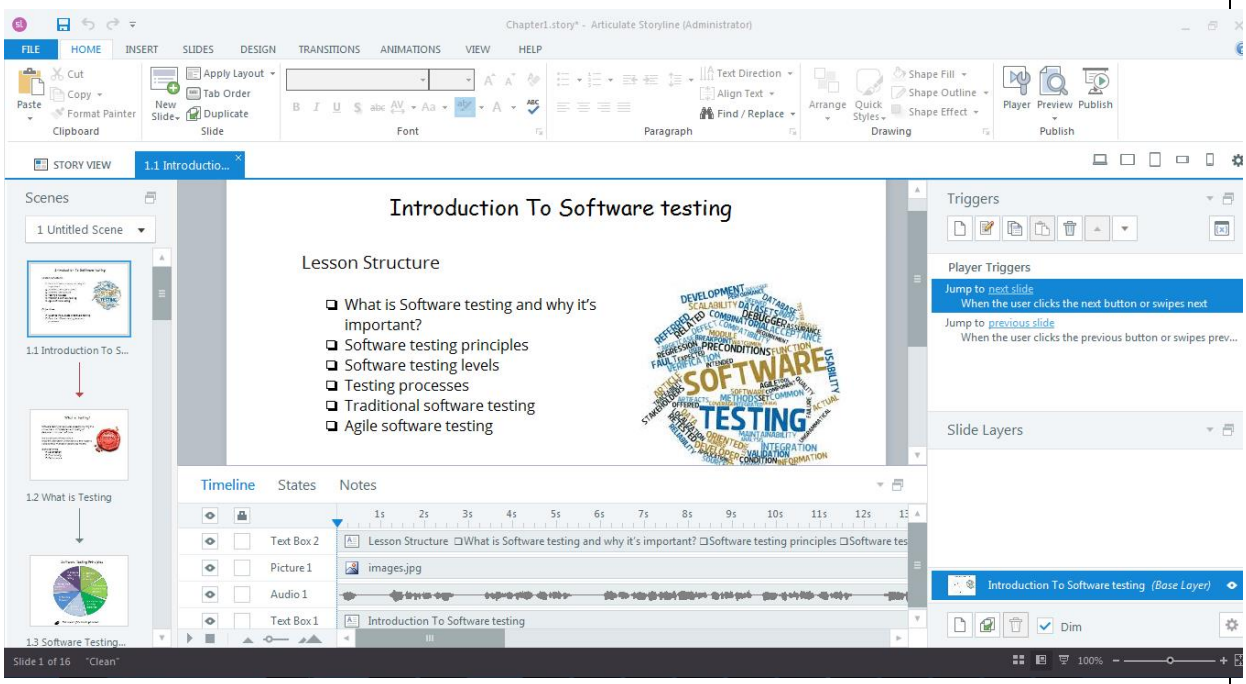


Figure 13: Authoring tool

Figure 13 represents the articulate storyline tool which was used as the main authoring tool to build the lesson content in SCROM format.

Adding Interactions / Triggers

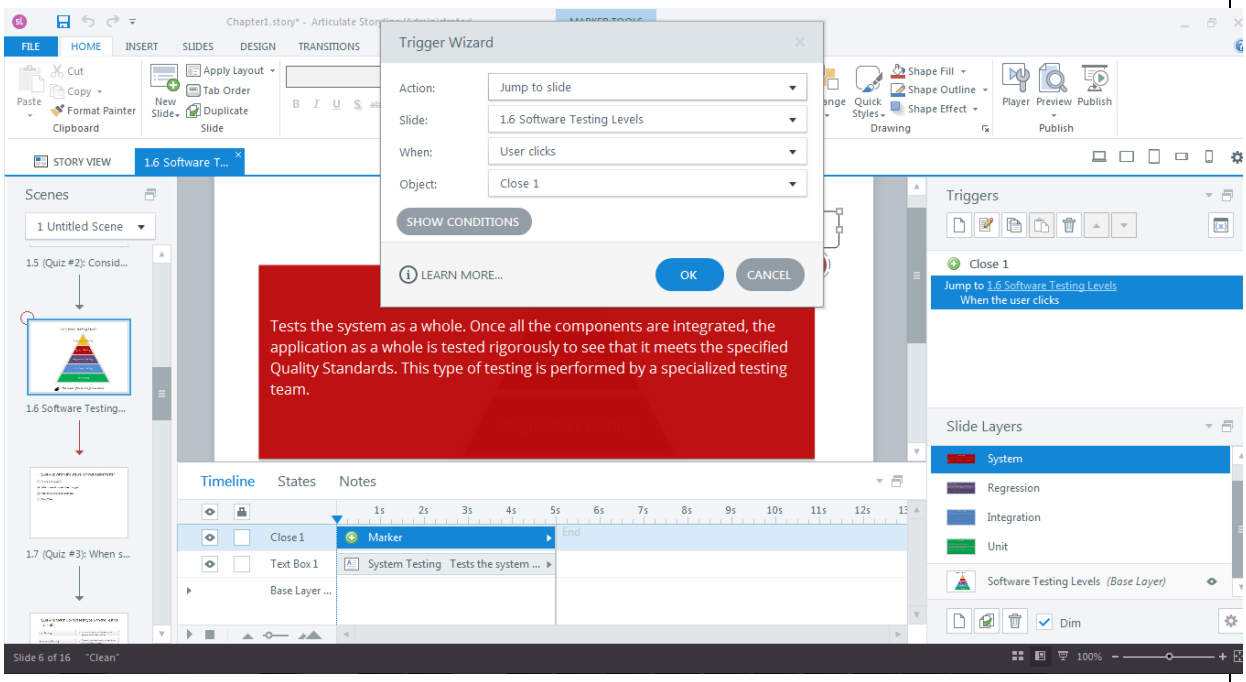


Figure 14: Using triggers in storyline

Figure 14 represents the screenshot of inserting trigger in interactive content.

Adding Quiz/Test

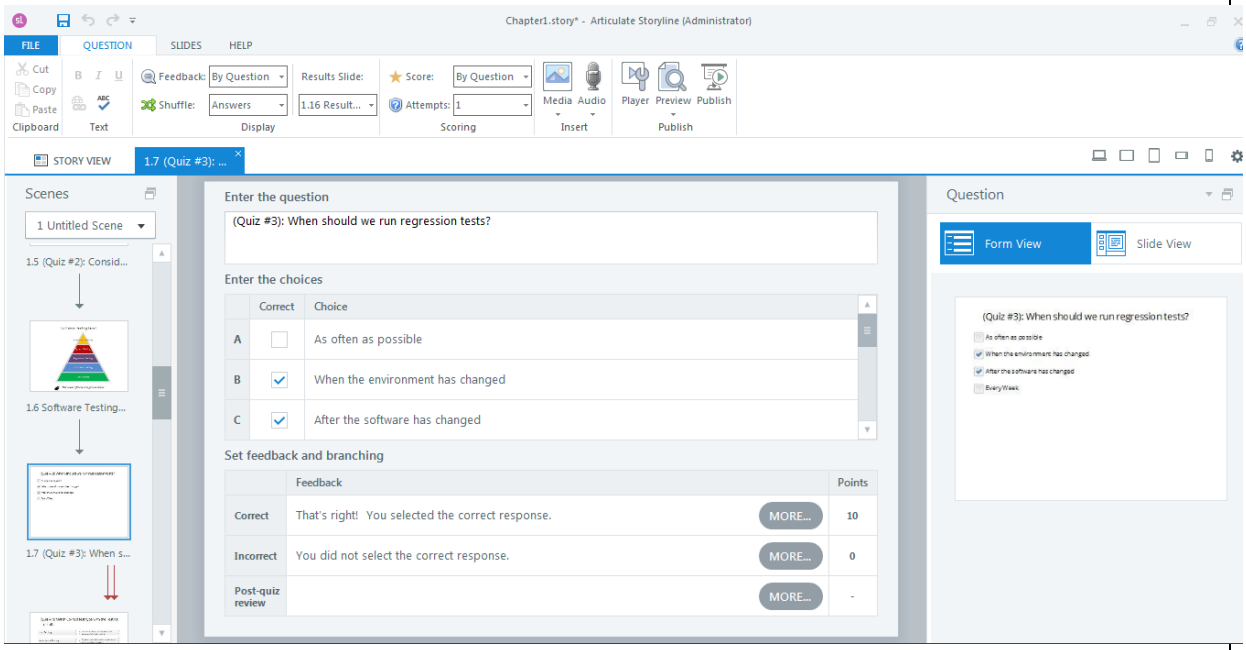


Figure 15: Adding quizzes

Figure 15 represents the screenshot of inserting quizzes to the assignment

Publishing SCORM

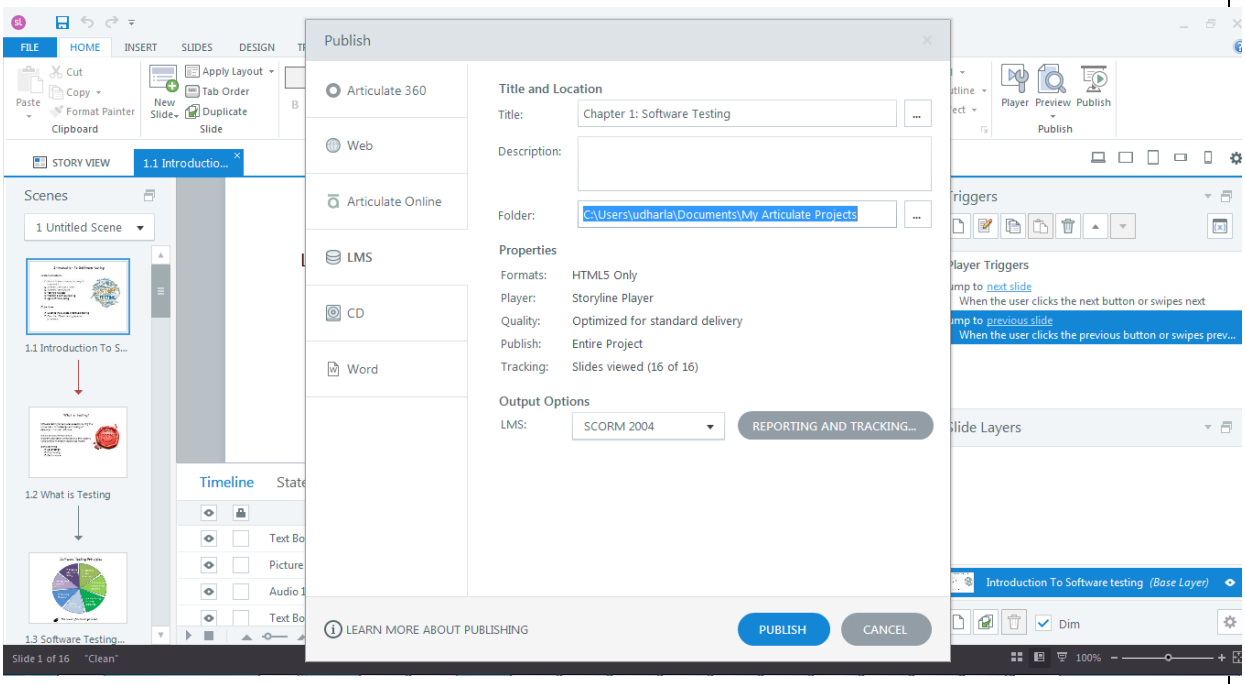


Figure 16: Publishing SCORM package

Figure 16 represents the screenshot of publishing the designed content as SCORM package. This SCORM package will be uploaded to the LMS in later stage.

Video Editing Using Adobe Premiere

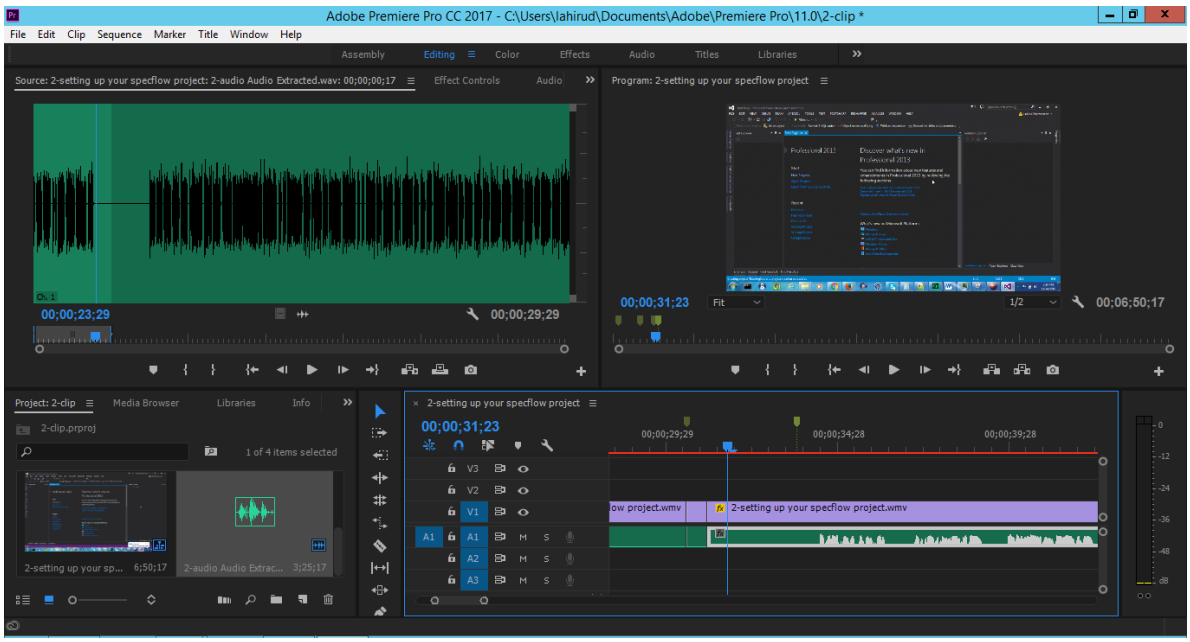


Figure 17: Video editing

Figure 17 represents a screenshot of adobe premiere software that has used as video editing tool

Audio editing using Adobe Audition

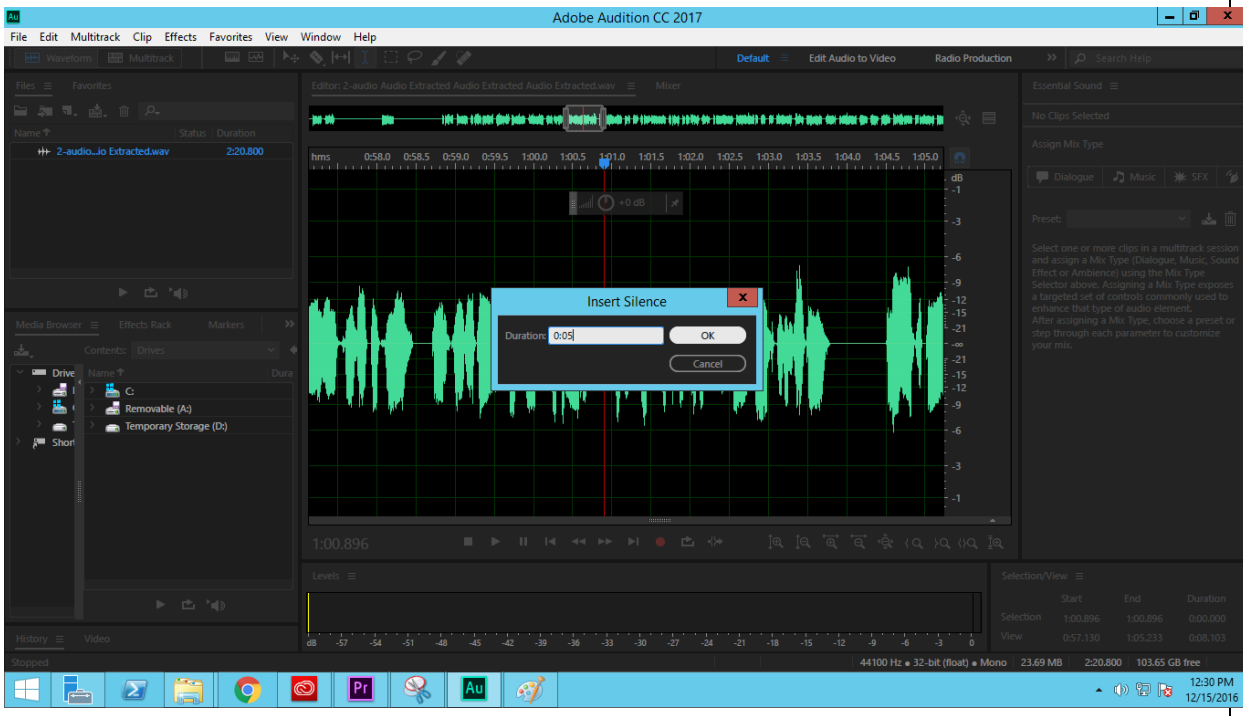


Figure 18: Audio editing

Figure 18 represents of a screenshot of adobe premiere which used as an audio editing tool.

CHAPTER 6: IMPLEMENTATION

This chapter discusses the implementation of the course using Learning Management System (LMS). In section 6.1 it is described about technical information about the system and infrastructure. In section 6.2 and 6.3 it is described about student enrollment process and managing learning activities, learner support provided to maintain the services related to learner support.

6.1. SYSTEM AND INFRASTRUCTURE

6.1.1. LEARNING MANAGEMENT SYSTEM

Learning management system (LMS) is a learning platform which provides interactive learning services to leaners with access information, tool and support educational delivery and management through internet. There is verity of learning management system available with different level of complexity and features such as managing course content, manage learners and keep track record of their progress and provide support for learning activities and administrative tasks.

Few factors considered in our course implementation in order to choose the right learning management system for the purpose.

- Feature support for the course delivery
- Technical knowledge required
- LMS license cost
- Hardware and software cost
- Maintenance and upgrade cost

Considering these factors, it has been decided to use Moodle is the right LMS for the course delivery

6.1.2. MOODLE

Moodle is free and open-source learning management systems widely use to deliver the course content and managing learners. Moodle provides collaborative tools such as forum discussion, chats and instant messaging and wiki based activities in addition to lesson and assessment delivery features. It also provides announcements and other communication tools

to bridge the communication with learner and instructor. Moodle has very large community who has been using the Moodle for quite some time and the support on technical or feature wise can be easily found. Moodle also supports SCORM and AICC standard contents integration and tracking the progress and learner activities. It also has self-enroll and instructor led enrollment options, which is very easy to managing user access control.

Moodle LMS in Student view

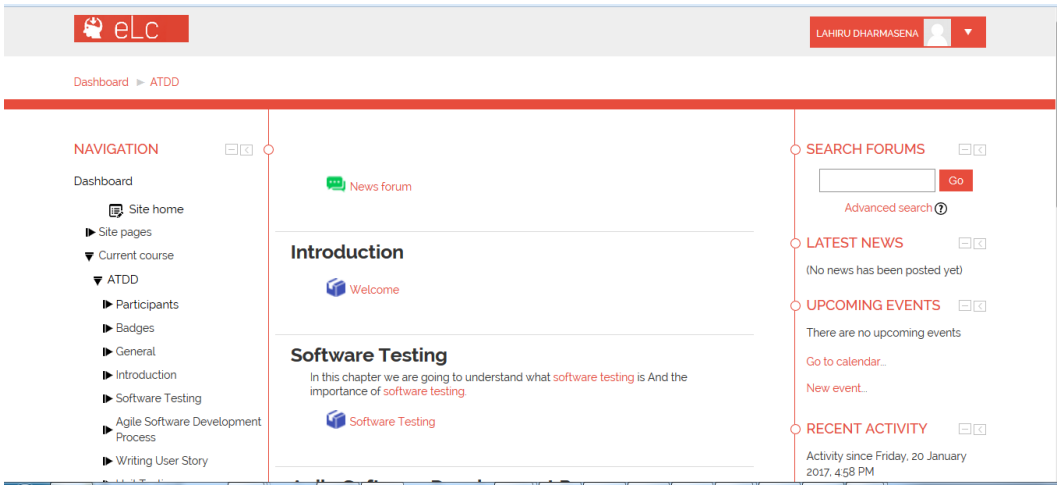


Figure 19: Moodle LMS

Figure 19 represents a screenshot of student view in Moodle LMS. Course module list can be seen in the first screen.

SCORM Lesson in Moodle

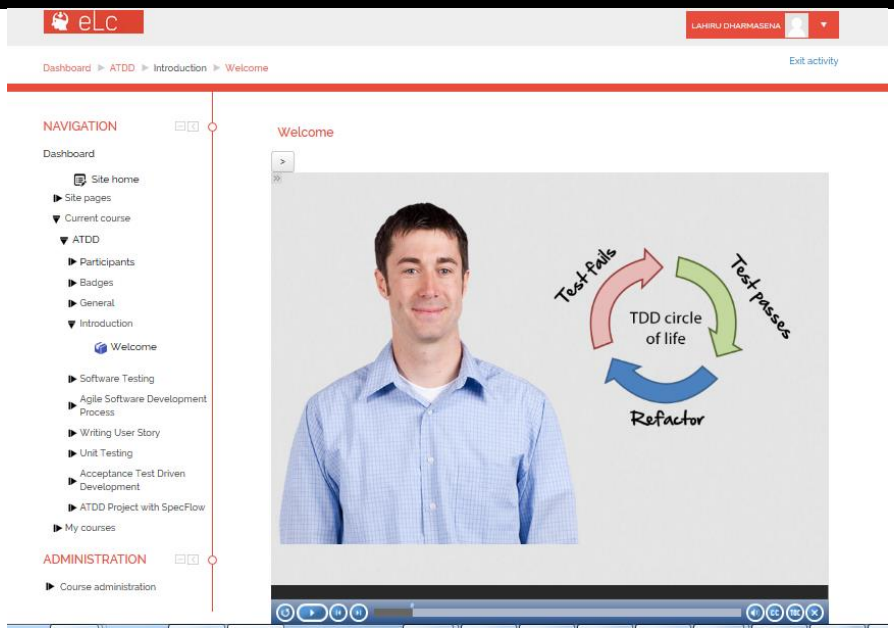


Figure 20: Lesson in moodle

Figure 20 represents the screenshot of a lesson content page in moodle LMS.

Moodle: Adding SCORM lesson in Administrator mode

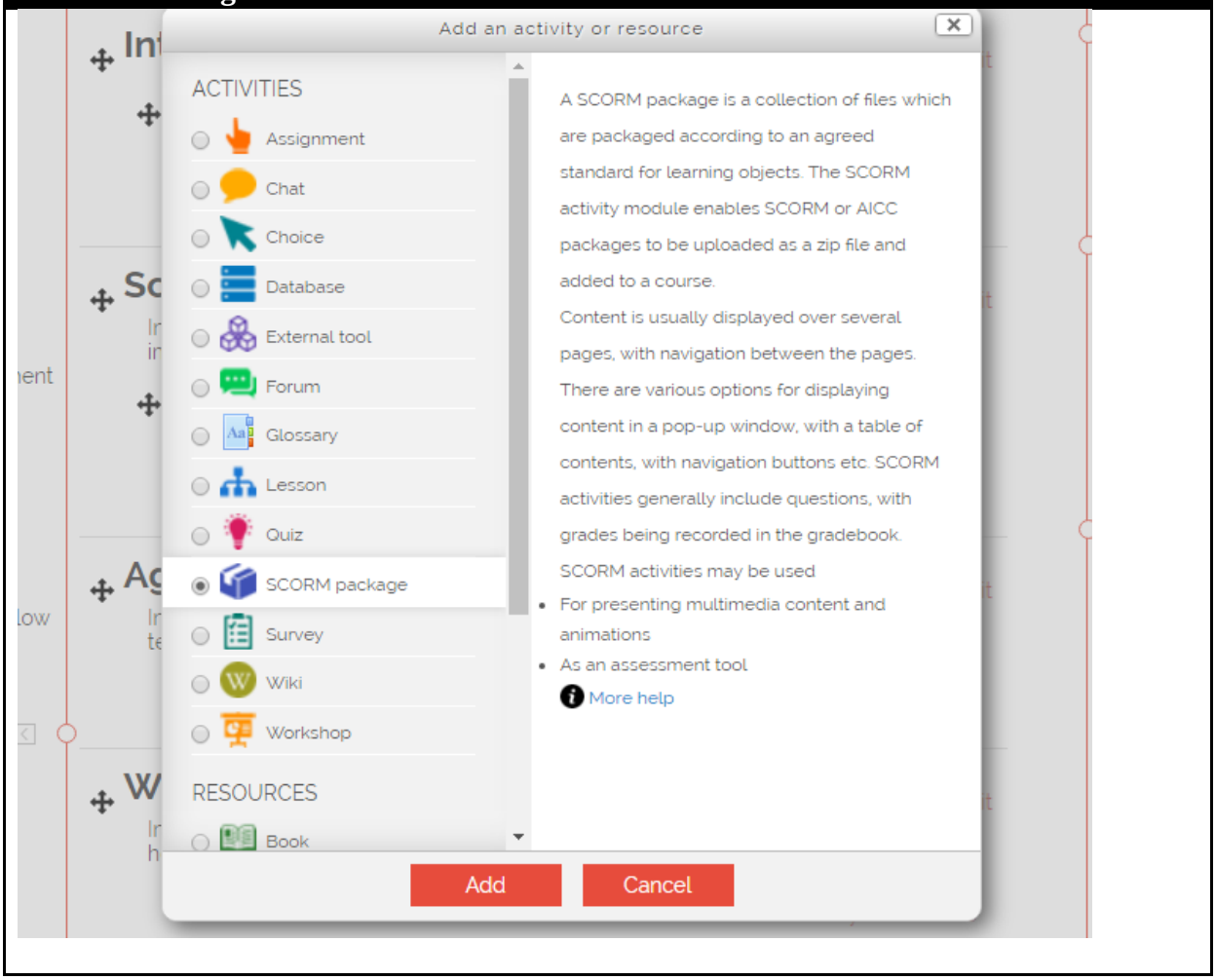


Figure 21: SCORM package upload to moodle

Figure 21 represents the screenshot of moodle lesson creation. In here, it is show where instructor adding SCORM package into the moodle lesson.

6.2. STUDENT ENROLLMENT PROCESS

Student enrollment process begins with selecting the candidates on voluntary basis. Awareness emails were distributed throughout the teams to find out potential candidates who have interested in the e-learning pilot project as learners. We received fair amount of responses and 24 individuals have been selected to follow the e-learning course.

Next step after selecting the individuals was providing them with the login details and Moodle url link in order to access to the learning environment. After successful login, they were instructed on how to enroll to the “Acceptance Test Driven Development” course through email. And weekly reminders have been sent to the candidates who have not been

enrolled to the course. Weekly emails have been sent for slowly progressing users and remind them about the progress and encourage them to follow the activities of the course in order to achieve the learnings.

6.3. MANAGING LEARNERS' ACTIVITIES

In collaborative online-learning, a group of learners creates interaction around common learning goals. As an online facilitator, it is our responsibility to ensuring that this process is organized, stimulating and efficient. The online facilitator has been performed the following tasks in order to support learners' activities

- provides information on tasks, deadlines and places to upload or download files
- accompanies participants during their work by checking workflow and individual or group results, composing working groups
- answers questions concerning tasks, deadlines or use of learning tools
- motivates participants to produce, reflect, exchange ideas and initiate discussions

The facilitator is the person that learners will approach with any questions; therefore, the facilitator has been allocated. Facilitator was available throughout the course and respond to questions as quickly as possible so that learners can proceed with their work and remain motivated.

6.3.1. USING COMMUNICATION TOOLS

E-learning activities have been managed by using range of communication tools, synchronous and asynchronous. Synchronous tools such as chats and instant messages were used. Emails, Announcements and Forum discussions were used as asynchronous tools.

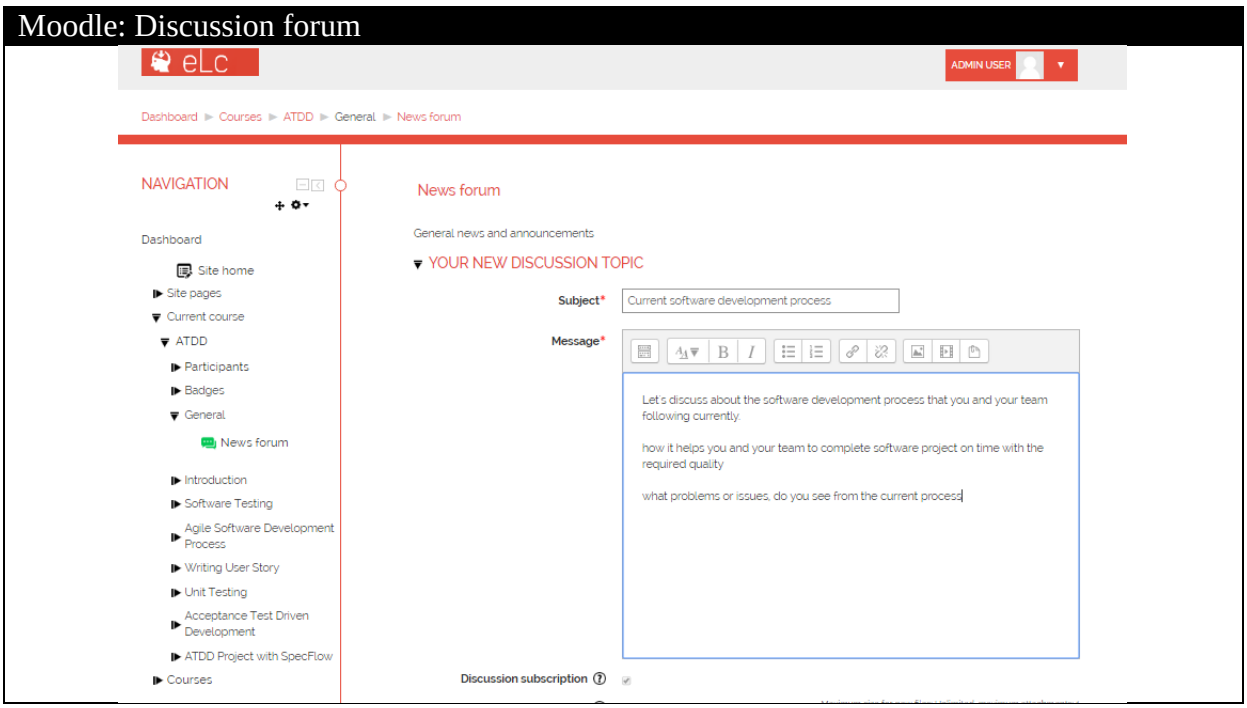


Figure 22: Communication in moodle

Figure 22 represents a screenshot of Moodle where instructor creating a announcement to inform learners about upcoming activity. Learners have been informed via emails about the upcoming learning activities and made them aware and engaged with the learning activities.

CHAPTER 7: EVALUATION

Measuring effectiveness and check for achievements for planned objectives are two most desired topics to get answered through the evaluation. According to Kirkpatrick's four levels of learning evaluation, it's considered (1) learners' reaction to the course, then (2) learnings achieved through the course, and then (3) behavioral changes or things have been changed in day to day work of the learners in the work setting after following the course and finally (4) Results or the Business Impact happened through the learnings and behavior changes by following the course. However at the scope of this project we are interested only the reactions and learnings because usually it takes few years to collect necessary data and understanding of behavior and results of the course and make a decision on return on investment (ROI)

7.1. LEARNERS' REACTIONS AND LEARNINGS

Learners' reaction to the course content and course delivery provide us information about how effective our content design is? How effective our content arrangement is? And how effective our learning environment is?

The evaluation also indicates how much learner has engaged with the course. How frequently leaner has log into the course and interacts with the learning materials, average time leaner has spent on learning activities and how actively participated in group discussions and forums. In order to measure the learner's reaction, two methods were used. (1) Questionnaire to collect data of learner's experience of the course and (2) Data collected through LMS of learner's behavior and interactivity using system logs.

After following the course for about 4 weeks, learners were given a questionnaire containing 32 questions to get data on their experience with the course. The questionnaire is containing four major categories. (1) Learning Content (2) Activities, Assessments and Feedbacks (3) Facilitator Support and (4) Overall experience

Data were also collected from LMS in order to understand the learners' achievements. Analysis was mainly focus on (1) course grades, (2) course completion rate (3) how often LMS was used by the learners and (4) level of collaboration. Under the level of collaboration, we were particularly interested about level of constructiveness when executing the

collaborative learning activities like group discussion. Also have the leaners seek instructor's or subject matter expert's help when they need help.

7.2. DATA COLLECTION

ATDD course was delivered as a pilot project to 24 selected individuals based on (1) volunteering first-come-first-serve basis. (2) Their role in the development team. It was considered to have same team composition ratio 1:3:2 in to the pilot sample. 1:3:2 ratio based on 1 member from product team, 3 members from development and 2 members from testing. Pilot test was carried for 1 month and after 1 month period; they have given the questionnaire to collect data for measure the effectiveness of the course as well as measure how satisfy they are with the course delivery.

Questionnaire containing close-end questions were used to collect data in quantitative format.

Table 7.1 contains questionnaire with closed-end questions. SD – Strongly Disagree, D – Disagree, A- Agree, SA – Strongly Agree

	Question	SD	D	A	SA
Category: Course content					
1	A clear statement of course requirements was provided at the beginning of the course				
2	The objectives for the course were clearly stated				
3	Audio and Video quality of the learning materials are acceptable				
4	The learning materials were clear and understandable				
5	The materials were accurate and current				
6	The materials were sequenced appropriately				
7	The materials were interesting and engaging				
Category: Course activities					
8	The course activities helped me to learn				
9	The course activities were sufficient for me to learn				
10	The course activities helped me to examine issues, to evaluate new ideas, and to apply what I have learned				
11	The course activities encouraged me to communicate and exchange ideas with other learners				
12	The activities were realistic and could be performed with the resources I had available				
13	The workload was just right				
Category: Course assessments					
14	The grading criteria were clear and explicit				
15	The assignments helped me to learn the course material.				
16	The assignments were challenging				

17	Assignments and tests were marked and returned promptly				
18	The assignments were related to what I have learned				
Category: Feedbacks					
19	The tutor clearly articulated the standards of performance				
20	The tutor provided clear constructive feedback				
21	The tutor provided meaningful guidance on my progress				
22	The tutor gave me constructive feedback on assignments				
23	Feedbacks were helped me to learn the course better				
Category: Supportive service					
24	The tutors could be contacted easily				
25	The tutors provided helpful information and explanations				
26	I have never found any disruptions in LMS				
27	The technical support information was given clearly				
28	The technical support was satisfactory				
Category: Overall experience					
29	The quality of the course met my expectations				
30	The course objectives, content, and assessments were consistent				
31	Considering both the limitations and possibilities of the subject matter and the course I am satisfied with the learnings				
32	I would recommend this course to a colleague				

Table 14: Questionnaire for analyze leaner experience

Table 14 represents the questionnaire that has been given to the leaners in order to gather the data on leaner experience after following the pilot course.

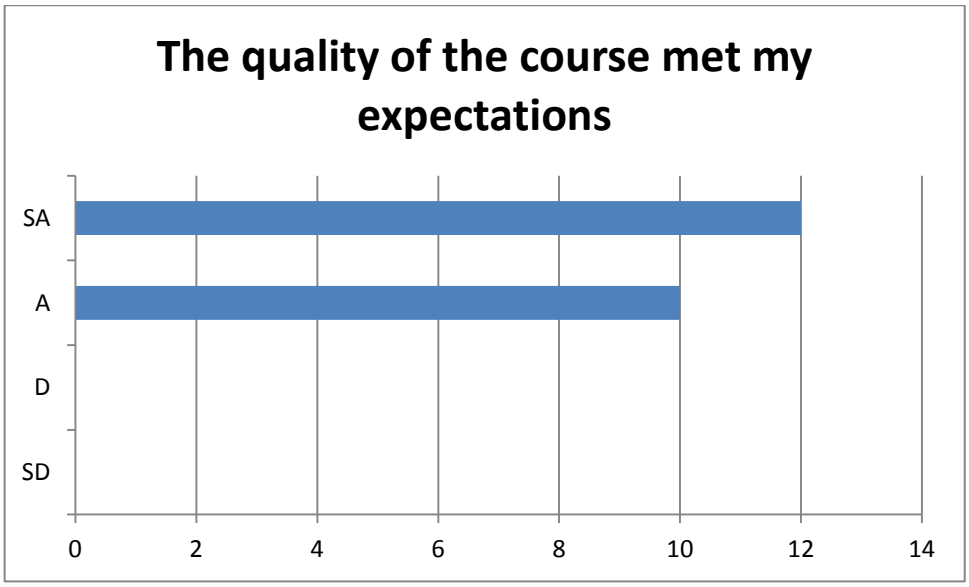
7.3. ANALYSIS

For the pilot course, we were reduced some modules from original The ATDD course where it required 10 weeks to complete the course with 4 hours workload per week. De-scoped ATDD course is expecting that learners will need 4 weeks to complete the course which roughly about 20 hours of workload total and 4-5 hours per week. Test run was carried out during January 9th to February 6th 2017 with 24 individuals. Evaluation forms were given to the participation at the end of the pilot run in electronic medium using Google forms.

Overall, pilot run was successful. 19 participants out of 24 were able to complete the course on time and 2 out of 5 who were behind the schedule, were able to complete course within next week of time. Overall 84% completion rate was achieved with the pilot course run.

7.3.1. PARTICIPANT'S FEEDBACK

Participant's feedback was generally positive. Most of them agreed or strongly agree that course has lived up to their expectation and learnings were useful (Figure 23)



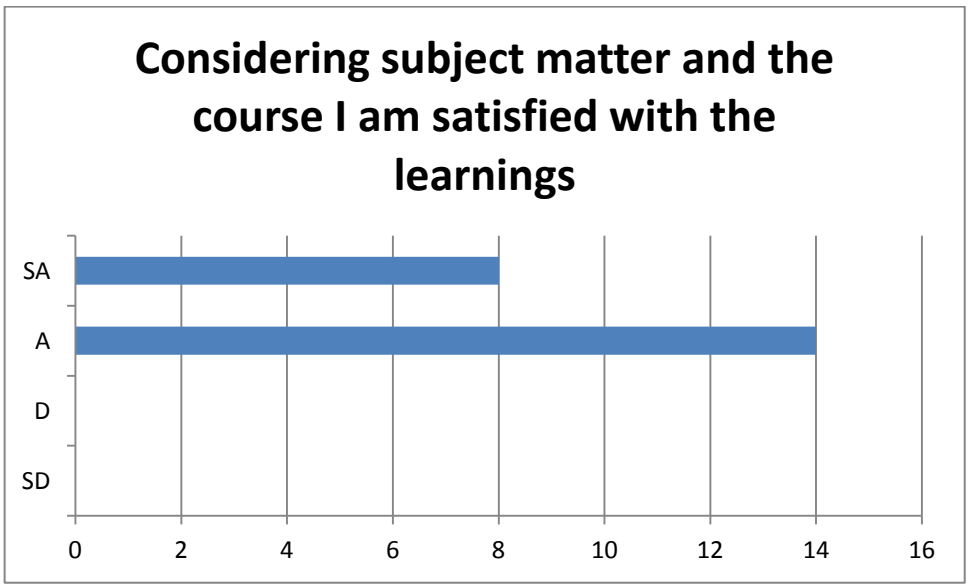


Figure 23: Participant's satisfaction

- **Consistency in the course material**

  Majority of participants agreed that they have found consistency in course materials from one module to other. However some of participants pointed out they found some overlaps in course modules (Figure 24). However we believed this is because we iterate ATDD concept in most of the module in order to highlight the concept and it was shown that it is unnecessary.

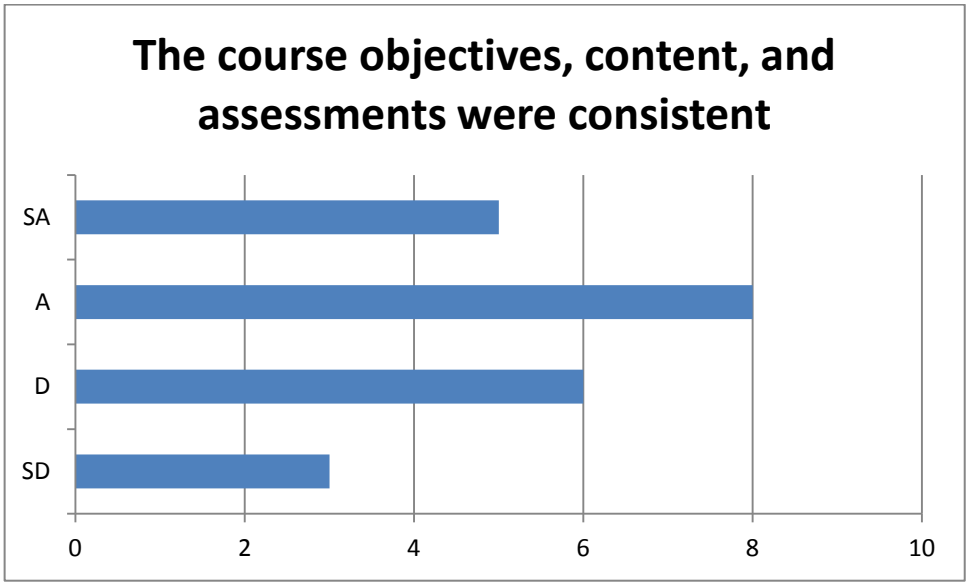


Figure 24: Participant's feedback on consistency of course materials and assessments

- **Audio Quality**

  It is pointed out by the participant's that audio quality is not exceptional as would they like. This is mainly because of the narrative voice is not clear and the pace was bit out of order in some occasions. Actions will be taken to revise the Audio narrations as the corrective action.

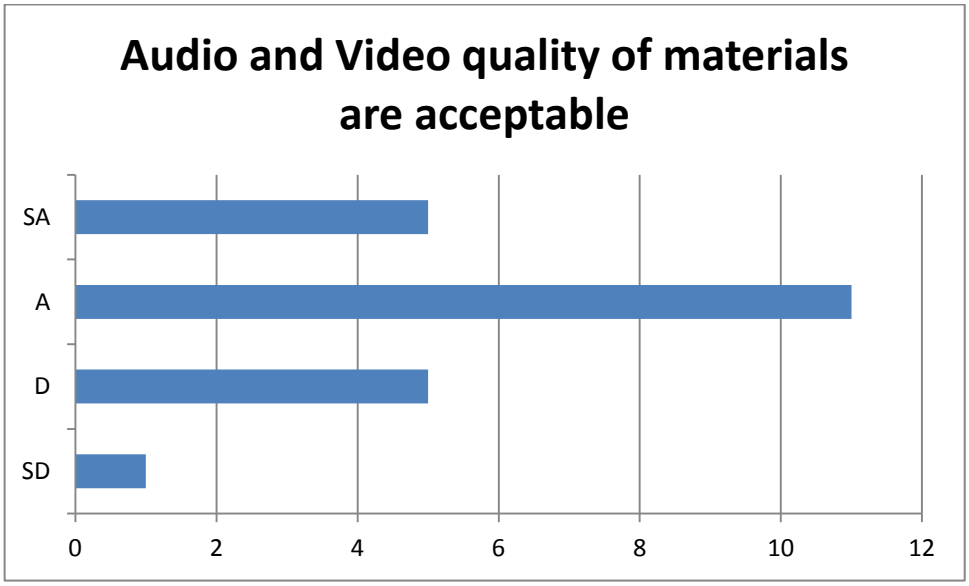


Figure 25: participant's feedback on Audio and video quality

Figure 25 represents the respondent's feedback on audio and video quality of the course content

- **Group discussions**

Figure 26 represents the respondent's feedback on group discussion and It is pointed out by the participants that group discussions were not active as they might expected and this is mainly because participants are not following the modules at same pace. So in order to participate in group discussion, they need to follow the modules in synchronized order. As a result of this finding it is recommended to have a schedule for each module and group discussion should be schedule accordingly.

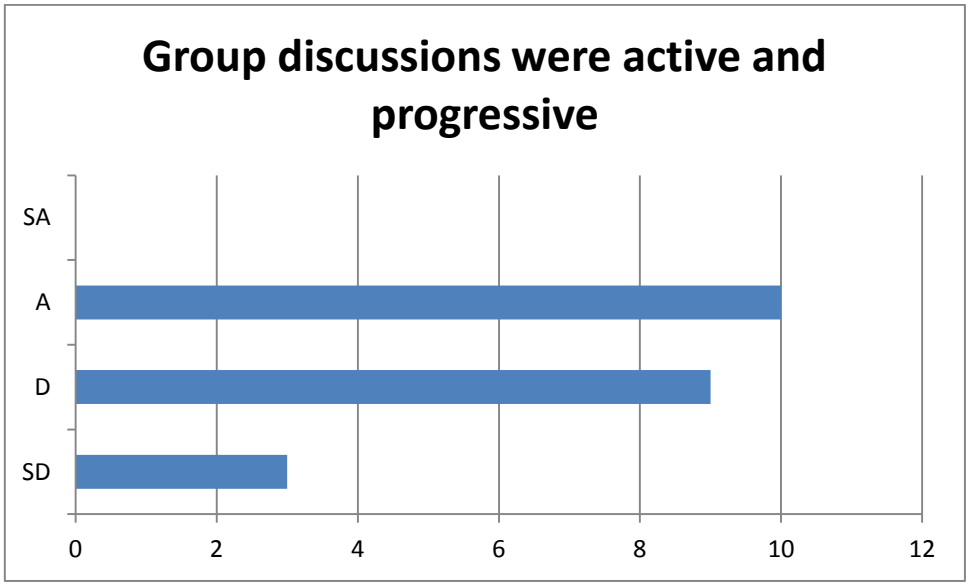


Figure 26: Participant's feedback on group discussion

CHAPTER 8: CONCLUSION AND FUTURE WORK

8.1. INTRODUCTION

With the rapid development of the information technology, E-learning has become effective learning method to deliver on-the-job training for corporate work setting. With the right support from the management of the organization and correct instructional design, implementation and delivery, e-learning deliver major impact compare to traditional learning.

The Acceptance Test Driven Development e-learning course was planned, analyzed, designed, developed, implemented and evaluated by using e-learning frameworks such as Khan's framework and Kirkpatrick model. Developing an e-learning course is neither easy nor cheap at all. It requires deep thinking and consideration about learners and learnings. A successfulness of the e-learning project is heavily is relies on pillars of pedagogical, technological, user interface, evaluation, management, resource support, ethical and institutional as B.H Khan described in 8 dimensional in his framework. Learner's behavior, Context of learning and organization's readiness are key factors for an e-learning project to become successful.

8.2. CONCLUSION

This section presents a conclusion of the project as well as the findings from project evaluation

During the project it is mainly focus on solve the performance problem that company facing in the software development process. With the breakdown of root cause analysis and needs assessment it is found that the major problem is due to lack of clarity in requirements, miscommunication and lack of test automation to cover the implementation. This finding is major step towards applying correct solution to the problem.

With the learner analysis, it is found that organization already has the technological, cultural and infrastructural capabilities to deliver the training through distance learning. Learners are already familiar with the e-learning systems and attitude towards learning the new technology was high (78%)

In the evaluation of course content and delivery, it is indicate that course should be more focus on course content and instructional design. It is also indicate that learner motivation is not necessary relies on learning the new technology but how it is rewarding them in return. Providing clear understanding about the benefits learners get in return is more important to motivate the learners.

Final conclusion of the project is e-learning is a viable solution for technical training. However it has various factors to be considered, ranging from management support, cultural and technological readiness, to the learner attributes instructors experience to technical support.

8.3. CHALLENGES AND LIMITATIONS

We were able to achieved most of the objectives of the project as we planned with the limited budget and time frame. Quality of the content was managed with multimedia principles and also checked against the alignment with learning outcomes. Assessments were also designed to complement the learning content and align with learning outcomes.

Few modules were taken out from the original course design in-order to accommodate the time and budget limitations. However in the future, those modules will be developed and include in the course. It is learnt that evaluating the actual code implementation is time consuming for the instructors. A new automated evaluating technique would be useful for assess the code implementation. However reliability of the automated system and feasibility of such a system needs to be analyzed further.

8.3.1. CULTURAL LIMITATION

In the analysis stage, questionnaire is limited to closed end questions rather than interview or observation because of the company policies and exposing minimal distractions the employees. It has been limited the capability of finding the correct work setting analysis and learner task analysis. Course work was based on the findings gathered through questionnaire and limited observation. It has impact on designing the course work.

8.3.2. INFRASTRUCTURAL LIMITATIONS

In overall, organization had a proper infrastructure for e-learning system with high bandwidth network. By considering this factor, Course has designed with demonstrational videos with high definition format. However, it is found that the high definition format is problematic when students are accessing the system from outside. It is needed to consider different formats when considering the public network capabilities when designing the content, not necessary focusing on limited infrastructure in organizational boundary.

8.3.3. TECHNOLOGICAL LIMITATION

In the course delivery, it could have been better if the learning system can provide simulated environment to practice the learning which were demonstrated through video content. With such a simulated environment, leaners behavior can be remotely monitor. However such a simulated environment for actual code development is not readily available and developing such a system is financially not feasible.

Similarly, Assessments were needed to evaluate by instructors manually since the complexity of the assessment and the nature of the assessment. Limitation of having automated evaluate for such assessments were limited the self-paced learning. However it is also financially not feasible to develop such an evaluation system.

8.4. FUTURE IMPROVEMENTS

As it was described in evaluation, we couldn’t find the behavioral change and business impact or Return on Investment (ROI) of the course due to course is in such an early stage and it would take few years to get visibility from those aspects. However it would be considered to build a matrix to find those behavioral changes using code complexity and test coverage. It is needed to capture the successfulness of learning in the work setting. Project delivery time, defect density and work life balance of the employees can be used as measurement of successfulness. It is needed to capture those inputs in order to decide the actual impact. So it is needed to implement system to capture the key success factors in coming years.

SonarQube is identified as a tool to measure code complexity and test coverage and it will be proposed to use by every development team in the future. By gathering data from SonarQube

tool we can monitor the progress of actual application of what they have learnt during the ATDD course work. In order to capture the code complexity and test coverage, this tool must be integrated with all project works and it is required considerable of effort.

Course it self needs to be improved with evolving nature of the technology and it needs to support the customizable learning paths in order to match with the learner experience. By doing so, learners can skip the course modules that they have already familiar. This is important since, learner's previous experience is varying from years of expertise and technology. It is also recommend a reward system, since it is needed to motivate the learners to follow the course work. Continuous support is needed from the management and technical stuff to deliver the course for larger audience of learners.

REFERENCES

[1] P. D. Wolf, *Best Practices in the Training of Faculty to Teach Online*, Journal of Computing in Higher Education, Faculty and Distance Education Services, University of Maryland University College, 2006.

[2] G.A. Heeger, *President's testimony to the Senate Budget and Taxation Subcommittee on Education, Business, and Administration of the Maryland General Assembly*. 2003 [online] available: http://www.umuc.edu/president/testimony/2003/2003senate.pdf.

[3] C.J. Bonk, *Online training in an online world*. USDLA Journal, 2002 [online] available: http://www.usdla.org/html/journal/MARO2_Issue/article02.html.

[4] B. Muirhead, *Training new online teachers.* USDLA Journal, 2002 [online] available: http://www.usdla.org/html/journal/OCT02_Issue/article06.html.

[5] N.L. Henning, *A case study: experiences in developing online courses at a community college*. Dissertation Abstracts International, 2000.

[6] N. Burke, *Teaching educators to teach in their pajamas: the perils and promises of a faculty based training program*, 1998 [online] available: http://leahi.kcc.hawaii.edu/org/tcon98/paper/burke.html.

[7] M.M. Lynch, *The online educator: A guide to creating the virtual classroom.* New York: Routledge Farmer, 2002.

[8] L.P. Hitch, D. Hirsch, *Model training. Journal of Academic Librarianship*, 2001

[9] B.L. Bower, *Distance education: facing the faculty challenge.* Online Journal of Distance Learning Administration. 2001. [online] available: http://www.westga.edu/-distance/ojdla/summer42/bower42.html.

[10] K. Mantyla, J.R. Gividen, *Distance learning: A step-by-step guide for trainers*. Alexandria, VA: American Society for Training and Development, 1997.

[11] T.E. Cyrs, *Teaching and learning at a distance: what it takes to effectively design, deliver, and evaluate programs*, San Francisco: Jossey-Bass. 1997.

[12] Mielke, *Effective teaching in distance education*. ERIC Digest. 1999

[13] T. McCallie, L. McKinzie, *Teaching online: a professional development model.* Society for Information Technology & Teacher Education International Conference. 1999.

[14] L. Star, *A connected experience: A faculty development model for teaching and learning*. Journal of Instruction Delivery Systems, 2001.

[15] G. Kearsley, *Tips for training online instructors*. Unpublished article. 2003. [online] available: http://home.sprynet.com/-gkearsley/Oltips.htm.

[16] E. Fredericksen, A. Pickett, P. Shea, *Factors influencing faculty satisfaction with asynchronous teaching and learning in the SUNY learning network*. Journal of Asynchronous Learning Networks, 2000.

[17] R.M. Palloff, K. Pratt, *Beyond the looking glass: What faculty and students need to be successful online*. In K.E. Rudestam & J. Schoenholtz-Read (Eds.), In Handbook of online learning: Innovations in higher education and corporate training. Thousand Oaks, CA: Sage Publications, 2002

[18] D. Olcott, *Instructional technologies--part two. Strategies for instructor success. Teaching at a distance: A handbook for instructors*. Mission Viej0, CA: League for Innovation in the Community College. 1999.

[19] D. Wesley, A Critical Analysis on the Evolution of E-Learning, International Journal on E-Learning 1, 2002.

[20] D. R. Tobin, All learning is self-directed. American Society for Instruction and Development (ASTD), 2000.

APPENDICES

APPENDIX A: OVERALL STORYBOARD

MODULE1: SOFTWARE TESTING

Visual	Explanation
Introduction To Software testing Topics of the module Learning objectives Image	This page describes objectives of this module, and topics covered by the module <topic List> What is Software testing and why it's important? Software testing principles Software testing levels Testing processes Traditional software testing Agile software testing <Objective list> describe the purpose of software testing describe different testing levels and processes
What is software Testing <description about software testing> Image	This page describe the purpose of software testing
Software Testing Principles Animation	This page contains a video explaining 7 software testing principles. Testing shows the presence of bugs, Exhaustive testing in impossible, Early testing, Defect clustering, The pesticide paradox, Testing is context dependent, Absence of errors fallacy. Animation should contain 7 principles. User click on each principle should bring up more detail view
Quiz [Multiple choice question]	Multiple choice questions from software testing principles.
Testing Levels	This page contains a video which describes testing levels in software development process. Unit Testing, Integration Testing, System Testing, Acceptance Testing

Animation	progressing level Animation should display Testing Levels. When user clicks on each testing level, it should display details for the selected level
Quiz [Multiple choice questions]	Multiple choice questions from software testing levels
Software Testing processes Traditional Testing practice Agile Testing practice	This page should describe the Characteristics of Traditional testing process and Agile testing process
Test	

MODULE2: AGILE SOFTWARE DEVELOPMENT PROCESS

Visual	Explanation
Agile Software development process Topics of this Module Learning objectives Image	This page describes objectives of this module, and topics covered by the module <topic list> Agile software development principles Agile Testing lifecycle Test Driven Development Test Automation <Objectives> Describe Agile Values Understand Agile Testing Identify Benefits of TDD and Test Automation
What is Agile? <Description> Image	This page describes What Agile Software Development is

Agile Software development principles and values <Description> Image	This page describes the Agile manifesto values
Agile Testing Lifecycle Animation	This page contains an Animation showing stages in Agile Testing Lifecycle. Detail of each stage will brings upon clicking on a lifecycle stage
Test Driven Development <Description> <Characteristics> Image	This page describes What Test Driven Development is and it's Characteristics
TDD Rhythm Animation	This page contains an Animation showing TDD Rhythm: red-green-refactor. Detail of the each item should appear upon clicking the each cycle stage
Test Automation <Description> Image	This page describes the What Test Automation is and it's characteristics

MODULE3: WRITING USER STORIES

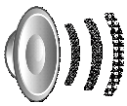
Visual	Explanation
What is User Story? <Description> Image	This page describes the What User story is and it's characteristics Structure:

<User story Structure>	As a … < user who requires this feature> I want … < do something > So That … <business justification>
User Story Process <Description of 3Cs> Image	This page describes the User Story Process. Which contains 3Cs (code, conversation, confirmation)
Steps to Create Good User Story Video	This page contains a Video to list down tips for how to make good user story
Acceptance Criteria <Description> Image	This page describes What Acceptance Criteria is and it's characteristics
Example of Acceptance criteria	This page shows Acceptance criteria taken from a user story

MODULE 5: ACCEPTANCE TEST DRIVEN DEVELOPMENT

Visual	Explanation
Introduction to ATDD Objectives: <Learning objectives of this module> Image Structure of the Module: <Topic List>	This page describes objectives of this module, and topics covered by the module <Topic List> What is ATDD Difference between ATDD and TDD Process of the ATDD Details of ATDD stages Benefits and challenges in ATDD Image: shows ATDD process figure

What is ATDD Definition of ATDD Image	This page describes the definition of ATDD. Image: shows figure of transforming TDD to ATDD TDD -> ATDD
ATDD vs TDD TDD Focus Image ATDD Focus	This page describes TDD focus area and difference between ATDD focus. Image shows ATDD process and TDD in its core
ATDD Process Animation	This page contains an Animation of ATDD process stages of Discuss, Distill, Develop, Demo When user hover over any stage, explanation will be provided for that stage.
Activity (Drag & Drop)	This page contains a Drag & Drop Activity in order to identify learners' knowledge on ATDD process. Collections of Activities given to the user. User must select correct activities related to ATDD process and should match into correct stage Activities: Create User case diagrams (not related) User Story Unit Test Coding Requirement Analysis Performance Testing (not related) Architecture reviews (not related) …

Definition of Done Image 	This page describes What DOD is. And the importance of having clear DoD Image: show questions to ask in order to get clear DoD How will user use the solution? (examples) How we can demonstrate it? How will we test it? Audio: Clip will play DoD and each of above questions to get to the DoD
Benefits of ATDD For Business Team: For Developers: For Testers: Image	
Challenges of ATDD Cultural Challenge: Slicing Requirements: Defining DoD:	
Quiz MCQ	Based on DoD, Benefits of ATDD and Challenges
Introduction to Gherkin Image	
Gherkin Syntax Feature: User Registration Check for home page See of the registration is working Also verify if the register user is displayed Background: Given: Clear already created user before begin	

Scenario: Register user with minimal password combination Given I've opened the website And I'm in the homepage When I click the register link Then I should see the register page And I fill the form with details |user name | password | cPassword| |lahiru | abc@123 | abc@123 |	
Few Examples:	
Test	Test is based on use cases. Learner will be give use cases in real software requirements and asked to provide Acceptance Test scenarios using Gherkin Syntax

MODULE6: SAMPLE ATDD PROJECT

Visual	Explanation
Setting up Project Environment Video	This page contains a video which demonstrates how to setup the .net environment, visual studio with nUnit and SpecFlow using NuGet package manager
Creating a Project Video	This page contains a video which demonstrates how to create sample project using visual studio
Creating Specification and Feature for the Acceptance Test	This page contains a video which demonstrates how to create step definition feature file using Gherkin language with SpecFlow

Video	
Quiz [Multiple choice questions]	
Add a Failing Test Video	This page contains a video which demonstrates how to add failing test into the project for a implementation which is yet to be developed
Implementation to Pass the Test Video	This page contains a video which demonstrates how to implement the code for the feature which results the Test to be passed.
Refactoring the Code Video Discussion Forum discussion on to clarify any issues occurred while	This page contains a video which demonstrates how to refactor the code with better design and retest for validity

APPENDIX B: DETAIL STORYBOARD

MODULE 1: SOFTWARE TESTING

Course Name: **Module 1: Software testing**			Storyboard File no. 01.01.01.00
Course section: 1.1			
Lesson Name: **Introduction To Software testing**			ID's name:
Objective(s):			SME's name:
Page Title: Introduction To Software testing		Page no. 01.01.01.00	CD's name:
Date Designed:	Date SME contributed:		Date verified:
Design			

Introduction To Software testing

Lesson Structure

- What is Software testing and why it's important?
- Software testing principles
- Software testing levels
- Testing processes
- Traditional software testing
- Agile software testing

Objectives

- Describe the purpose of software testing
- describe different testing levels and processes

Special Comment(s):

Page Title: What is Testing?	Page no. 01.01.02.00	SME's name:
Date Designed:		Date verified:

Design

What is Testing?

Software testing is the process used to identify the correctness, completeness and quality of developed computer software.

It is the process of executing a program/application under positive and negative conditions by manual or automated means.

It checks for the

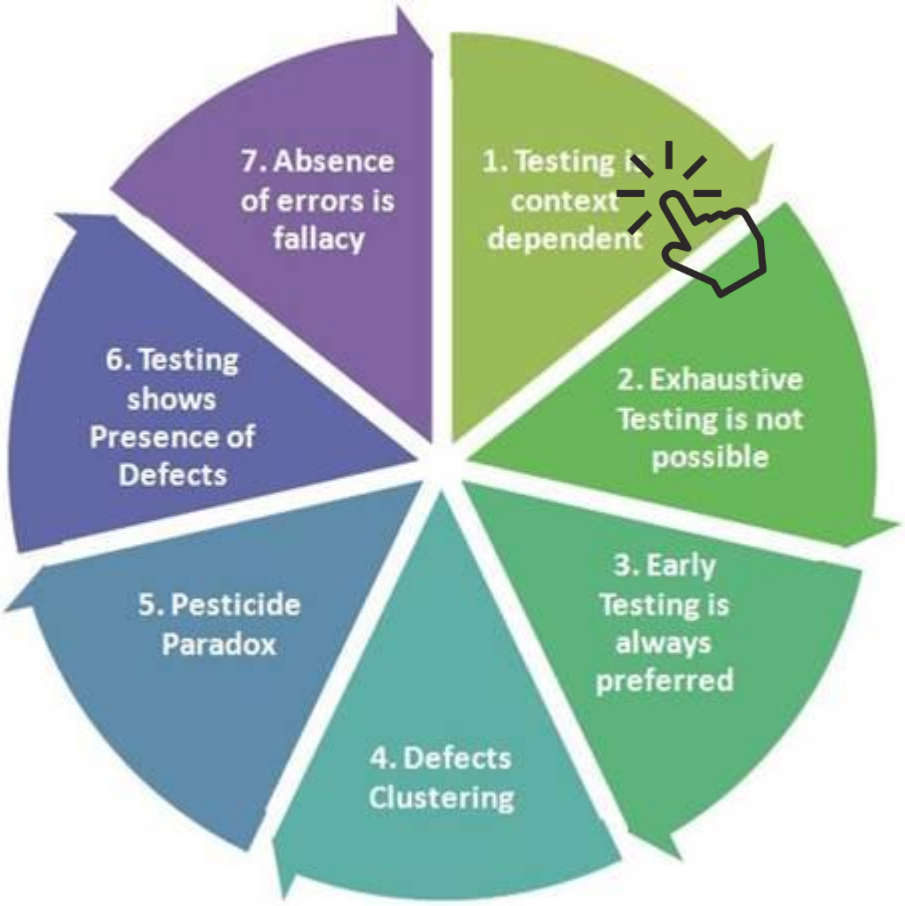
- Specification
- Functionality
- Performance

Special Comment(s):

Page Title: Software Testing Principles	Page no. 01.02.01.00	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles



Special Comment(s):
This is an interactive animation

User should be able to click any item to get more details about the principle. Please refer next screen (01.02.01.01) for detail view

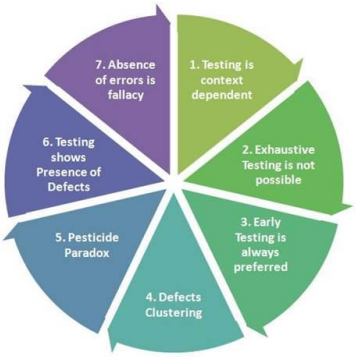
Hand icon should be flashing to indicate "click" here for more action.

Page Title: Software Testing Principles	Page no. 01.02.01.01	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles

Principle 1 – Testing shows the presence of defects.



- ❑ Testing show that defects are present but cannot make sure that there are no defects.
- ❑ Testing show that defects are present but cannot make sure that there are no defects.
- ❑ Design Test cases is very essential step which find defects as many as possible.

< Back

Special Comment(s):

User can be click "< back" or "Color Wheel" to get back to 01.02.01.01 screen

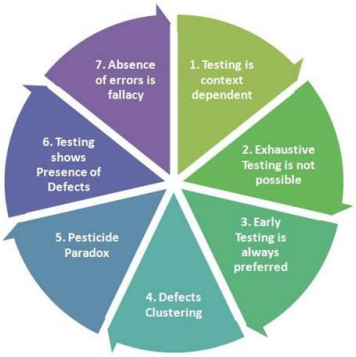
Page Title: Software Testing Principles	Page no. 01.02.01.02	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles

Principle 2 – Exhaustive Testing is impossible

- It is impossible to test all possible all possible combinations of input cases, data and scenarios.
- Tester should focus on the most critical priorities and risks; we could say risk analysis and priorities should be used to focus testing efforts.



< Back

Special Comment(s):

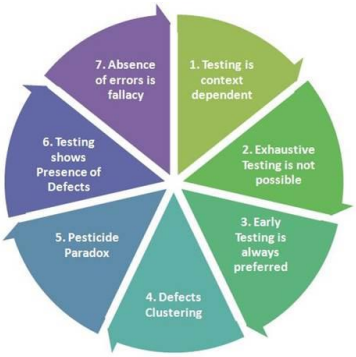
User can be click "< back" or "Color Wheel" to get back to 01.02.01.01 screen

Page Title: Software Testing Principles	Page no. 01.02.01.03	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles

Principle 3- Early testing



- Testing should begin as early as possible in SDLC and focus on pre-defined objectives. Errors identified later in the process leads more expensive to fix in comparison to fix the errors we find in early stage in process.

- Error in a product specification may be much easier to fix. However, if that error is transferred to the coding, then fixing the mistake could be more costly and time-consuming will be disadvantage

< Back

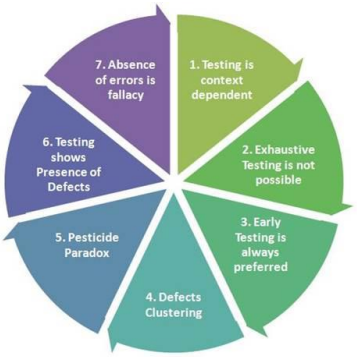
Special Comment(s):

User can be click "< back" or "Color Wheel" to get back to 01.02.01.01 screen

Page Title: Software Testing Principles	Page no. 01.02.01.04	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles



Principle 4 – Defect clustered

- "Pareto principle" states %80 defects will be found in approximately %20 of modules. This means %20 defect causing %80 of problems.
- Small no. of modules contains most of defects during pre-release testing. There are no equal distribution errors between different modules. If one defect on any module, you will likely to find more.

< Back

Special Comment(s):

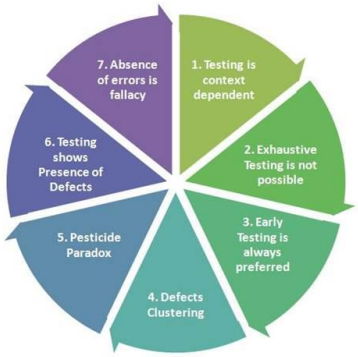
User can be click "< back" or "Color Wheel" to get back to 01.02.01.01 screen

Page Title: Software Testing Principles	Page no. 01.02.01.05	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles

Principle 5- Pesticide Paradox



- If using the same set of test over and over again testers should Revise Existing test cases and Develop newer ones, which will be able to uncover more bugs

- Use a variety of tests and techniques to find a range of defects across different areas of the product. Avoid using the same set of tests over and over on the same product or application, because this will reduce the range of bugs you will find as same test case is not much affected now.

< Back

Special Comment(s):

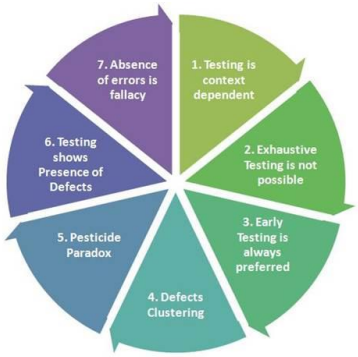
User can be click "< back" or "Color Wheel" to get back to 01.02.01.01 screen

Page Title: Software Testing Principles	Page no. 01.02.01.06	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles

Principle 6- Testing context is dependent



- Software testing varies testing efforts depending on circumstances. Different methods, techniques and types of testing are related to the type and application nature.
- The same tests should not apply across because different software products have different requirements, functions and purposes.

< Back

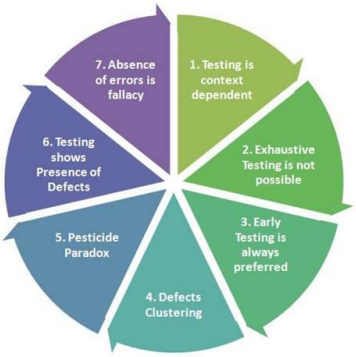
Special Comment(s):

User can be click "< back" or "Color Wheel" to get back to 01.02.01.01 screen

Page Title: Software Testing Principles	Page no. 01.02.01.07	SME's name:
Date Designed:		Date verified:

Design

Software Testing Principles



Principle 7 – Confusing an absence of errors with product fit is a fallacy

- fallacy states that testing might succeed in locating and correcting all possible defects in software, software itself might not to be fit for use by an end- user

- Testing main should be matched with user requirements. Finding and fixing defects does not help if the system built is unusable and does not fulfill the user's need and expectations. If the system built is unusable and does not full fill the user's needs and expectations then finding and fixing defects does not help

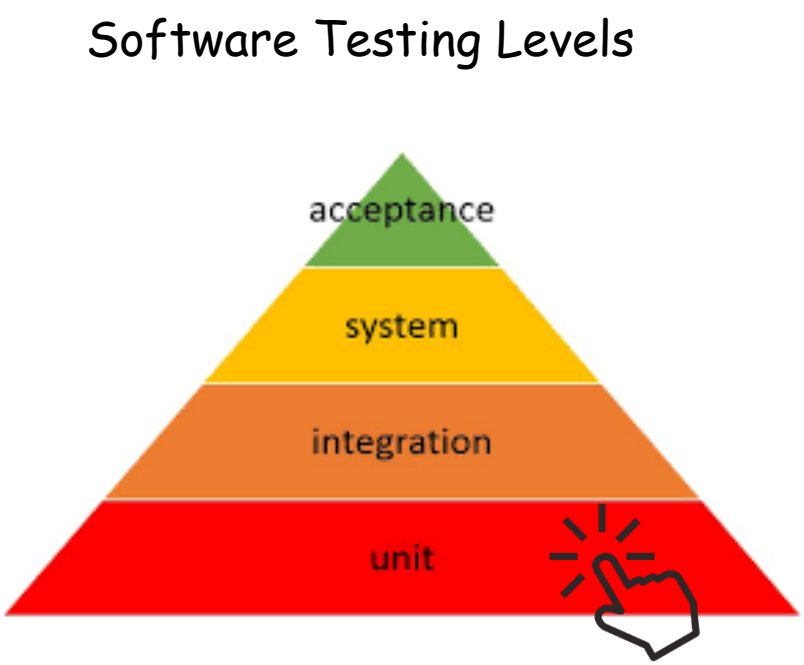
< Back

Special Comment(s):

User can be click "< back" or "Color Wheel" to get back to 01.02.01.01 screen

Page Title: Software Testing Levels	Page no. 01.03.01.00	SME's name:
Date Designed:		Date verified:

Design



Click on any Level to get more details about the Level

Special Comment(s):

This is an interactive animation

User can click any Level to see more details about the selected level

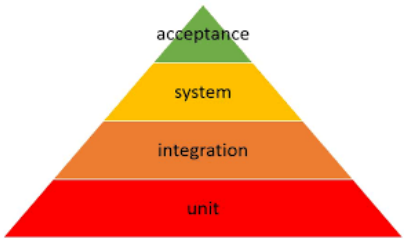
Icon should flash and indicate "click" to get more details. Refer next screen (01.03.01.01) for detailed view

Page Title: Software Testing Levels	Page no. 01.03.01.01	SME's name:
Date Designed:		Date verified:

Design

Software Testing Levels

Unit Testing



This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases.

Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

< Back

Special Comment(s):

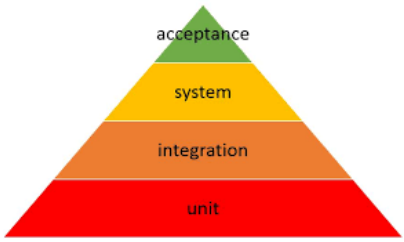
User can click "< Back" or Pyramid to get back to the first screen (01.03.01.00)

Page Title: Software Testing Levels	Page no. 01.03.01.02	SME's name:
Date Designed:		Date verified:

Design

Software Testing Levels

Integration testing



Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

Bottom-up: begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds

Top-down: highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

< Back

Special Comment(s):

User can click "< Back" or Pyramid to get back to the first screen (01.03.01.00)

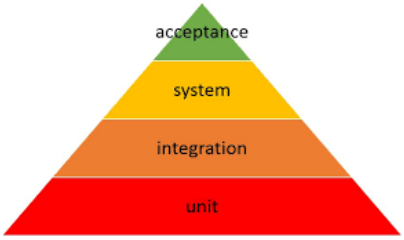
Page Title: Software Testing Levels	Page no. 01.03.01.03	SME's name:
Date Designed:		Date verified:

Design

Software Testing Levels

System Testing

Tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.



Regression Testing

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

< Back

Special Comment(s):

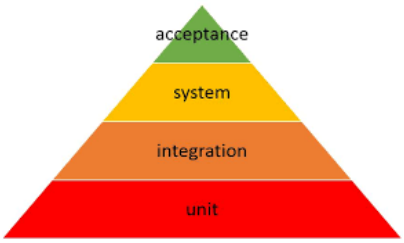
User can click "< Back" or Pyramid to get back to the first screen (01.03.01.00)

Page Title: Software Testing Levels	Page no. 01.03.01.04	SME's name:
Date Designed:		Date verified:

Design

Software Testing Levels

Acceptance Testing



This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application.

By performing acceptance tests on an application, the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system

< Back

Special Comment(s):

User can click "< Back" or Pyramid to get back to the first screen (01.03.01.00)

Page Title: Software Testing Processes	Page no. 01.04.01.00	SME's name:
Date Designed:		Date verified:

Design

Software Testing Processes

Traditional waterfall development model

- Testing is performed by an independent group of testers after the functionality is developed
- often results in the testing phase being used as a project buffer to compensate for project delays, thereby compromising the time devoted to testing

Agile or Extreme development model

- Uses "test-driven software development" model.
- Unit tests are written first. Of cause these test fail initially; as they are expected to.
- Code is written it passes incrementally larger portions of the test suites.
- The test suites are continuously updated as new failure conditions and corner cases are discovered
- They are integrated with any regression tests that are developed.
- Unit tests are maintained along with the rest of the software source code and integrated into the build process.
- The ultimate goal of this test process is to achieve continuous integration where software updates can be published to the public frequently

Special Comment(s):

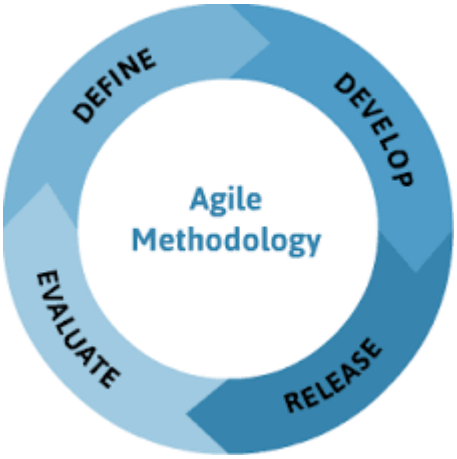
Course Name: **Module 2: Agile Software development process**		Storyboard File no. 02.01.01.00
Course section: 2.1		
Lesson Name: Agile Software Development Process		ID's name:
Objective(s):		SME's name:
Page Title: Agile Software Development Process	Page no. 02.01.01.00	CD's name:
Date Designed:	Date SME contributed:	Date verified:

Design

Agile Software Development Process

Lesson Structure

- Agile Software development principles
- Agile Testing Lifecycle
- Test Driven Development
- Test Automation



Objectives

- Describe Agile Values
- Understand Agile Testing
- Identify Benefits of TDD and Test Automation

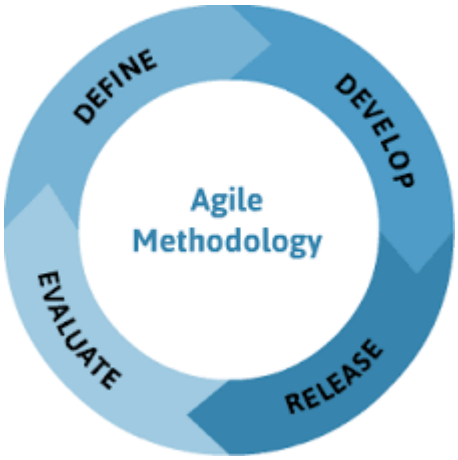
Special Comment(s):

Page Title: What is Agile?	Page no. 02.01.02.00	SME's name:
Date Designed:		Date verified:

Design

What is Agile?

- Agile – A continuous stream of business value
- Agile methods in contrast to traditional ones produce completely developed and tested features at frequent intervals of 2-4 weeks
- Iterative approaches mean we can trade features for time instead of sacrificing quality



Special Comment(s):

Page Title: Agile Software development principles	Page no. 02.01.03.00	SME's name:
Date Designed:		Date verified:

Design

Agile Software development principles

Manifesto for Agile is focus on 4 values

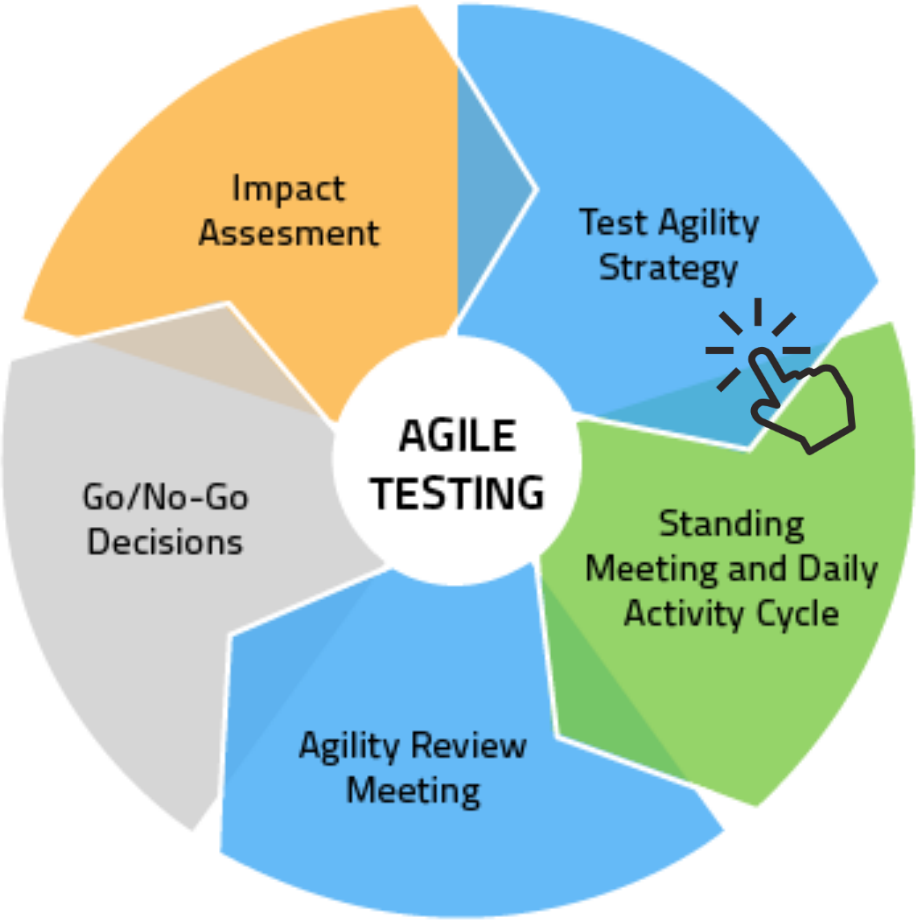
- Individuals and interactions over Processes and tools
- Working software over Comprehensive documentation
- Customer collaboration over Contract negotiation
- Responding to change over following a plan.

Special Comment(s):

Page Title: Agile Testing Lifecycle	Page no. 02.02.01.00	SME's name:
Date Designed:		Date verified:

Design

Agile Testing Lifecycle



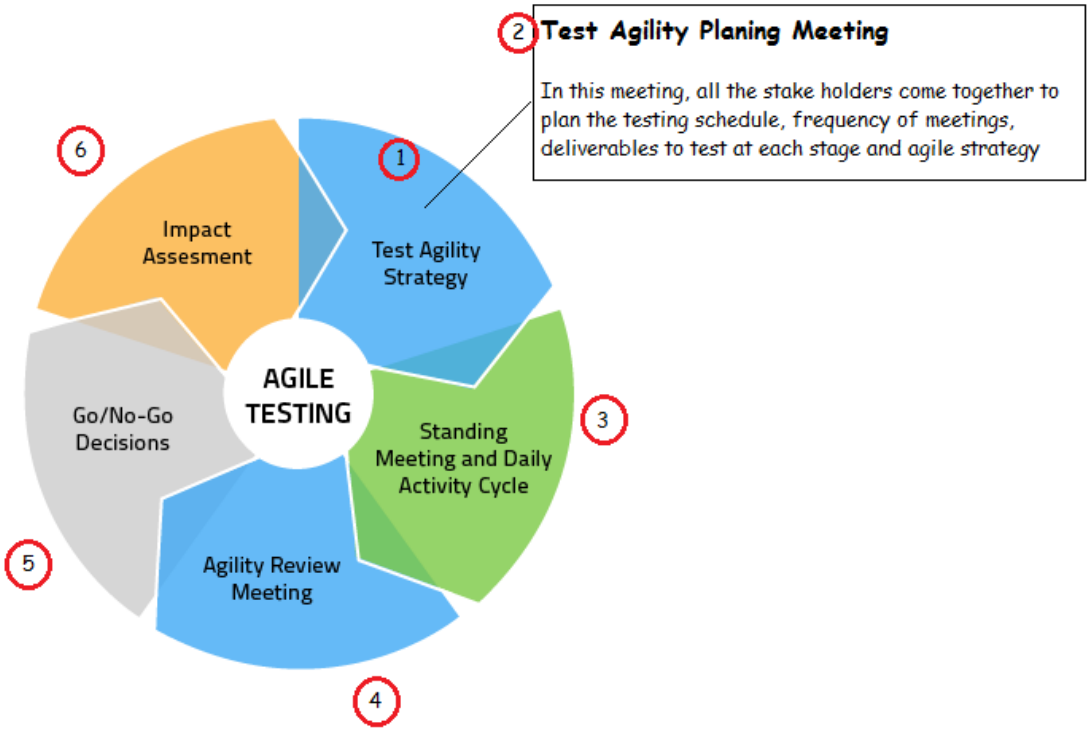
Click on any stage to get more details about the stage

Special Comment(s):
This is an interactive animation. User can click on any stage to get more details

Should indicate this is the starting point and "click" get more details. Refer next screen(02.03.01.01) for more details

Page Title: Agile Testing Lifecycle	Page no. 02.02.01.01	SME's name:
Date Designed:		Date verified:

Design



Special Comment(s):

As shown in the sample, when user clicks on 1 (Test Agility Strategy) detail explanation should appear as shown (2) in a box.

Other areas (3,4,5,6) also should follow the same pattern. Here's the text for those.

3:**Daily Scrums**

These daily standups which typically happens at the beginning of the day help catch up on testing status and set the course for the rest of the day and goals for tomorrow.

4:**Agility Review**

This is a periodic review meeting that is typically performed once a week where larger group of stakeholders
meet and asses the progress against milestones

5:**Release Readiness**

Here we review if our incremental features that have been developed are ready to go live. or else we go back to previous stage in the cycle

6:**Impact Assessment**

We gather inputs from user and the stake holders. This acts as a feedback for the next deployment cycle.

Page Title: Test Driven Development	Page no. 02.03.01.00	SME's name:
Date Designed:		Date verified:

Design

Test Driven Development (TDD)

"Test-driven development" refers to a style of programming in which three activities are tightly interwoven: coding, testing in the form of writing unit tests and design in the form of refactoring

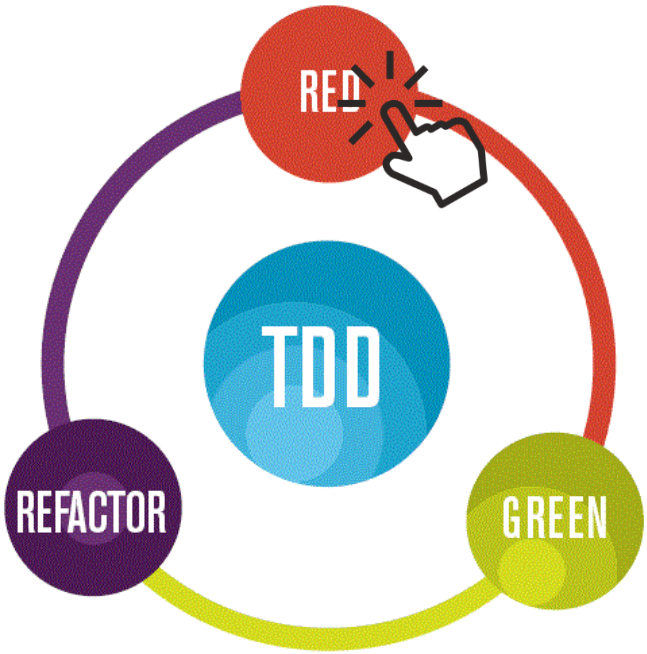
- TDD is a Test-First Approach. Write the Test first and then write the code
- Think about "How to use a component" first then about "How to implement"
- As much about Design Technique as Testing Technique
- As much about Working Document as Testing

Special Comment(s):

Page Title: Red-Green-Refactor	Page no. 02.03.02.00	SME's name:
Date Designed:		Date verified:

Design

Red - Green - Refactor



Click on any stage to get more details about the stage

Special Comment(s):

This is an interactive animation

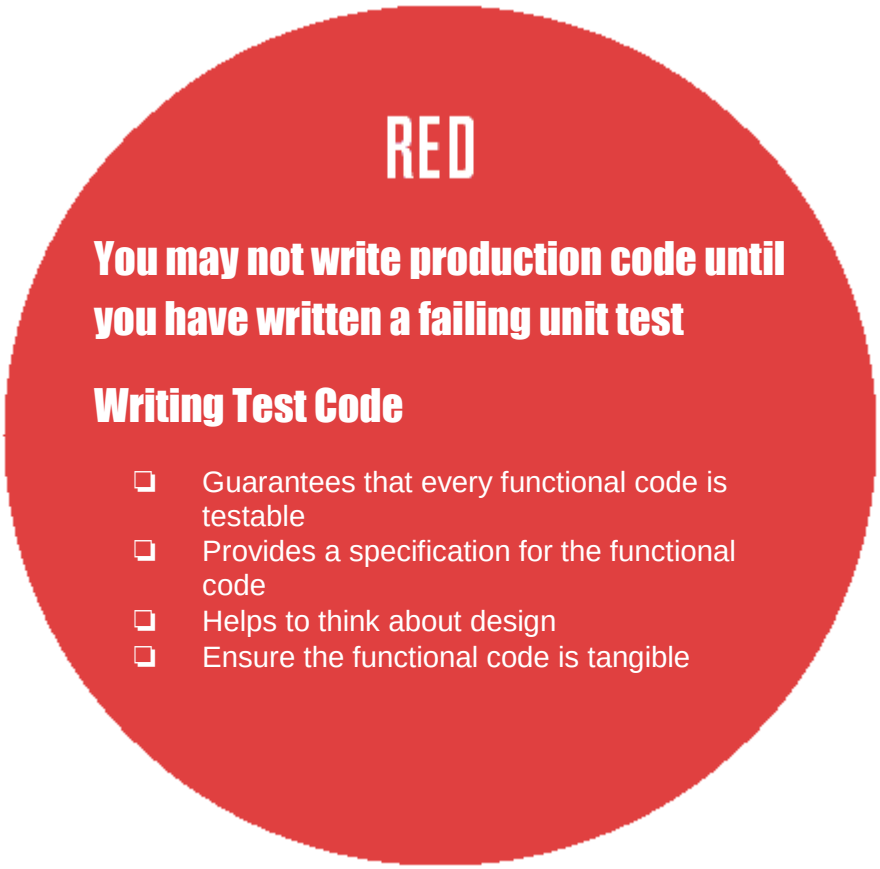
When user clicks "Red" circle, circle should become wider covering all other element and should display detail view of the stage.

Please refer the next screen

Page Title: Red-Green-Refactor	Page no. 02.03.02.01	SME's name:
Date Designed:		Date verified:

Design

Red - Green - Refactor



Click on any stage to get more details about the stage

Special Comment(s):

Background color : # FE4040
Font Heading: Impact (size: 16)
Font Bullet-list: Arial (size: 12)
Font Color: White #FFFFFF

Content:
You may not write production code until you have written a failing unit test

Writing Test Code
- ❑ Guarantees that every functional code is testable
- ❑ Provides a specification for the functional code
- ❑ Helps to think about design
- ❑ Ensure the functional code is tangible

---------- End of Content ----------

Similarly for Green

Background color : # A4AD19
Font Heading: Impact (size: 16)
Font Bullet-list: Arial (size: 12)
Font Color: White #FFFFFF

Content:
write "just enough" code, the simplest possible, to make the test pass

Write Functional Code

- ❏ Fulfill the requirement
- ❏ Write the simplest solution that works
- ❏ Leave improvements for a later step
- ❏ code written is only designed to pass the test

---------- End of Content ----------

Similarly for Refactor

Background color : # 4E1B55
Font Heading: Impact (size: 16)
Font Bullet-list: Arial (size: 12)
Font Color: White #FFFFFF

Content
"refactor" the code until it conforms to the simplicity criteria

Refactor

- ❏ clean-up the code
- ❏ make sure the code expresses intent
- ❏ Re-think the design
- ❏ Delete unnecessary code

---------- End of Content ----------

Page Title: Test Automation	Page no. 02.04.01.00	SME's name:
Date Designed:		Date verified:

Design

What is Test Automation

Automatically executed code that verifies an application in a reliable, resilient, repeatable and fully automated way.

- Reliable. Always the same result.
- Resilient. Refactoring the Application does not break the Tests
- Repeatable. Can execute any number of times.
- Automatically Executed. No human interaction to prepare and start the Test Automation
- It verifies requirements or predefined behaviors. Test Automation does not ensure that the requirements improve the application

Special Comment(s):

Course Name: **Module 3: Writing User stories**		Storyboard File no. 03.01.01.00
Course section: 3.1		
Lesson Name: Writing User stories		ID's name:
Objective(s):		SME's name:
Page Title: Agile Software Development Process	Page no. 03.01.01.00	CD's name:
Date Designed:	Date SME contributed:	Date verified:
Design		

Writing User Stories

Lesson Structure

- ❑ Introduction to User Story
- ❑ Writing the Right User Story
- ❑ Converting User stories into Acceptance Tests

Objectives

- ❑ Describe components of user story
- ❑ Identify What makes a good user story
- ❑ Able to Write good user story

Special Comment(s):

Page Title: User Story	Page no. 03.01.02.00	SME's name:
Date Designed:		Date verified:

Design

User Story

User Story is short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template:

As a <type of user>, I want <some goal> so that <some reason>.

User stories are often written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion. As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written

Special Comment(s):

Page Title: User Story Process	Page no. 03.01.03.00	SME's name:
Date Designed:		Date verified:

Design

User Story Process - 3Cs

Card

- For each feature, write it down in an index card.

Conversation

- Reconcile what the person writing the story and the person building it both understand
- discuss to discover different interpretations
- Clarify and refine the story
- Discuss to explore solution scenarios

Confirmation

- Once it is built, how do you check that it's done?
- Check against a list of things (acceptance criteria)
- Demonstrate functionality

Special Comment(s):

Page Title: Writing Good User Stories	Page no. 03.02.01.00	SME's name:
Date Designed:		Date verified:

Design

Writing Good User Stories

Special Comment(s):
This is a video

"TOP 10 Tips" should be displayed in a "Red" (# FE4040) circle. Until the narration is on this display should be there. Refer next screen and script (05.03.01.01) afterwards.

Transcript:

"User stories are probably the most popular agile technique to capture product functionality.

Working with user stories is easy.
But telling effective stories can be hard.

The following ten tips help you create good stories.

So, let's look at them individually"

Page Title: Writing Good User Stories	Page no. 03.02.01.01	SME's name:
Date Designed:		Date verified:

Design

Special Comment(s):

As narration goes "Number 1", #1 should appear in the middle of the Red Circle. And when narration goes "Users Come First", world should appear from the bottom of the circle and slowly come to the bottom of the #1 as narration goes. When the narration goes "Number 2", all the remaining text should be cleared and "#2" should appear in the middle of the Red circle and repeat the same behavior for the rest of the items

Transcript:

Number 1: Users Come First

As its name suggests, a user story describes how a customer or user employs the product; it is written from the user's perspective. What's more, user stories are particularly helpful to capture a specific functionality, such as, searching for a product or making a booking

If you don't know who the users and customers are and why they would want to use the product, then you should not write any user stories. Carry out the necessary user research first, for

example, by observing and interviewing users. Otherwise, you take the risk of writing speculative stories that are based on beliefs and ideas—but not on data and evidence.

Number 2: Use Personas to Discover the Right Stories

A great technique to capture your insights about the users and customers is working with personas. Personas are fictional characters that are based on first-hand knowledge of the target group. They usually consist of a name and a picture; relevant characteristics, behaviors, and attitudes; and a goal. The goal is the benefit the persona wants to achieve, or the problem the character wants to see solved by using the product.

But there is more to it: The persona goals help you discover the right stories: Ask yourself what functionality the product should provide to meet the goals of the personas.

Number 3: Create Stories Collaboratively

A user story is not a specification, but a communication and collaboration tool. Stories should never be handed off to a development team. Instead, they should be embedded in a conversation: The product owner and the team should discuss the stories together.

You can take this further and write stories collaboratively, for instance, as part of your product backlog grooming process. This leverages the creativity and the knowledge of the team and results in better user stories.

Number 4: Keep your Stories Simple and Short

Write your stories so that they are easy to understand. Keep them simple and short. Avoid confusing and ambiguous terms, and use active voice. Focus on what's important, and leave out the rest.

Number 5: Start with Epics
An epic is a big, vague story. It is typically broken into several user stories over time leveraging the user feedback on early prototypes and product increments. You can think of it as a headline and placeholder for more detailed stories.

Starting with epics allows you to sketch the product functionality without committing to the details. This is particularly helpful for describing new products and features: It allows you to capture the rough scope, and it buys you time to learn more about how to best address the needs of the users. It also reduces the time and effort required to integrate new insights. If you many detailed stories in the product backlog, then it's often tricky and time-consuming to relate feedback to the appropriate stories and you have to be careful not to introduce inconsistencies.

Number 6: Refine the Stories until They are Ready

Break your epics into smaller, detailed stories until they are ready: clear, feasible, and testable. All development team members should have a shared understanding of the story's meaning; the story should not too big and comfortably fir into a sprint, and there has to be an effective way to determine if the story is done.

Number 7: Add Acceptance Criteria

As you break epics into smaller stories, remember to add acceptance criteria. Acceptance criteria

complement the narrative: They allow you to describe the conditions that have to be fulfilled so that the story is done. The criteria improve the story, they make it testable, and they ensure that the story can be demoed or released to the users and other stakeholders. As a rule of thumb, use three to five acceptance criteria for detailed stories.

Number 8: Use Paper Cards

User stories emerged in Extreme Programming, and the early XP literature talks about story cards rather than user stories. There is a simple reason: User stories were captured on paper cards. This approach provides three benefits: First, paper cards are cheap and easy to use. Second, they facilitate collaboration: Everyone can take a card and write down an idea. Third, cards can be easily grouped on the table or wall to check for consistency and completeness and to visualize dependencies. Even if your stories are stored electronically, it is worthwhile to use paper cards when you write new stories.

Number 9: Keep your Stories Visible and Accessible

Stories want to communicate information. Therefore don't hide them on a network drive. Make them visible, for instance, by putting them up on the wall. This collaboration, creates transparency, and makes it obvious when you add too many stories too quickly, as you quickly start running out of wall space.

Number 10: Don't Solely Rely on User Stories

Creating a great user experience requires more than user stories. User stories are helpful to capture product functionality, but they are not well suited to describe the user journeys and the visual design. Therefore complement user stories with other techniques, such as, story maps, workflow diagrams, storyboards, sketches, and mockups.

Additionally, user stories are not good capturing technical requirements. If you need to communicate what an architectural element like a component or service should do, then write technical stories or use a modeling language like UML.

Finally, writing user stories is worthwhile when you develop software that's likely to be reused. But if you want to quickly create a throwaway prototype or mockup to validate an idea, then writing stories may not be necessary. Remember: User stories are not about documenting requirements; they want to enable you to move fast and develop software as quickly as possible and not to impose any overhead.

Page Title: Acceptance Criteria	Page no. 03.03.01.00	SME's name:
Date Designed:		Date verified:

Design

Acceptance Criteria

Acceptance Criteria are conditions which a software application should satisfy to be accepted by a user or customer.

- Set of statements tells the result is passing or fails for both functional and non-functional requirements.
- In Agile, acceptance criteria make sure the user story is completed or not.
- It is also known as test completion criteria and fit criteria.
- The acceptance conditions and non- acceptance conditions should be clearly mentioned in the acceptance criteria.

Special Comment(s):

Page Title: Example of an Acceptance Criteria	Page no. 03.03.02.00	SME's name:
Date Designed:		Date verified:

Design

Example of an Acceptance Criteria

User Story: As a Shopper I want to Create an orders in online shopping cart, so I can purchase times

Criteria:

- User should be able to selects multiple items and add to shopping cart.
- The user should be able to see the items in the shopping cart.
- The user should be able to purchase items using their local currency.
- The user should be able to see an order number when the payment method is made.

Other examples of Acceptance Criteria can include:

- The user would not be able to submit a form if all the mandatory fields are not entered.
- Modes of payments can be selected, like payment by credit card, debit card.
- An automatic email is sent once the payment is made and confirmed.

Special Comment(s):

Course Name: **Module 5: Acceptance Test Driven Development**		Storyboard File no. 05.01.01.00
Course section: 5.1		
Lesson Name: Introduction to ATDD		ID's name:
Objective(s):		SME's name:
Page Title: Introduction to ATDD	Page no. 05.01.01.00	CD's name:
Date Designed:	Date SME contributed:	Date verified:

Design

Introduction to ATDD

Lesson Structure

- What is ATDD
- ATDD Process
- Definition of Done (DoD)
- Benefits of ATDD
- Challenges of Practicing ATDD
- Gherkin Syntax
- Few Examples

Objectives

- Describe how ATDD helps to bridge the communication Gap
- Use Gherkin syntax to create feature files
- Create feature files using Gherkin

Special Comment(s):

Page Title: What is ATDD	Page no. 05.01.02.00	SME's name:
Date Designed:		Date verified:

Design

What is ATDD?

- ❑ Stands for Acceptance Test Driven Development
- ❑ ATDD is a collaborative process where the business customer, product owner and Agile team members defines Acceptance Criteria
- ❑ Focus is on Business Rules
- ❑ Acceptance Test design begins before start coding
- ❑ Specification is provided with examples

Special Comment(s):

Page Title: ATDD vs TDD	Page no. 05.01.03.00	SME's name:
Date Designed:		Date verified:

Design

ATDD vs TDD

TDD

- is about building the software right
- focus on individual functionality (Unit tests)
- implement by the developers

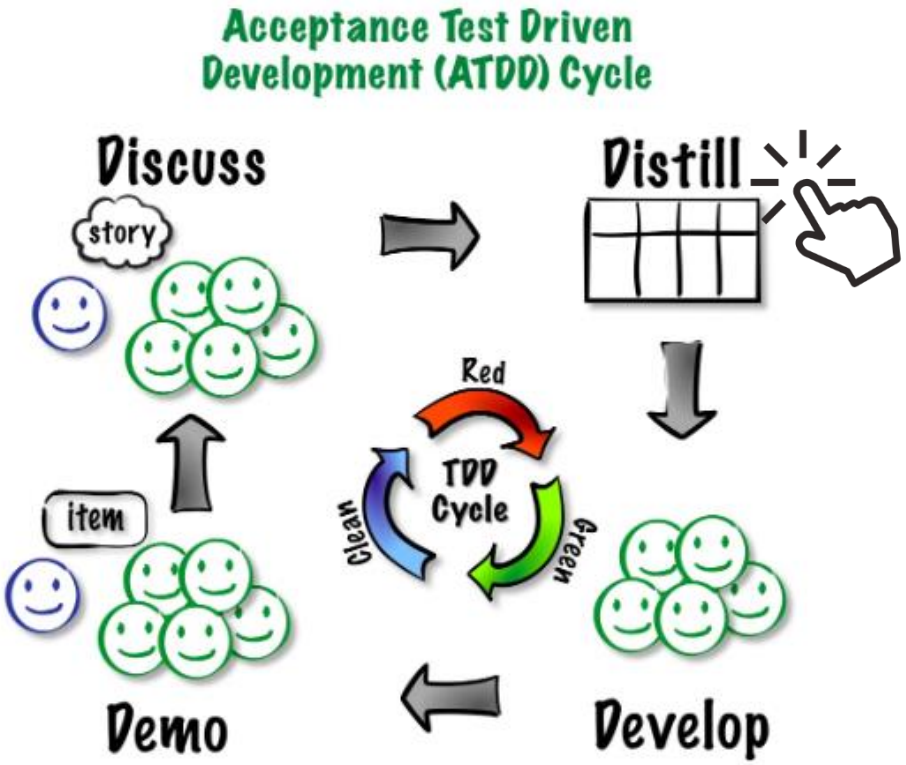
ATDD

- is about building the right software
- focus on the business rule (Acceptance tests)
- implement by the agile team including the developers

Special Comment(s):

Page Title: ATDD Process	Page no. 05.02.01.00	SME's name:
Date Designed:		Date verified:

Design



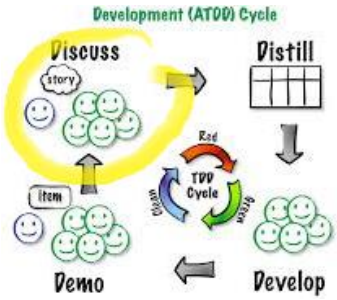
Click on any stage to get more detail

Special Comment(s):


icon should flash and indicate it's clickable and get more details about the stage

Page Title:	Page no. 05.02.01.01	SME's name:
Date Designed:		Date verified:

Design



Discuss:

Required Artifact: User Story – need a business requirement to start from. What is needed is a business value to be delivered.

Format: Meeting with access to a whiteboard

How it works: The Business Analyst has previously developed the user story through his conversations with the Product owner, he will be able to explain the user story's business value. He will also be able to explain the conditions of satisfaction. Shared understanding of goals will guarantee the real goal is attained and not a consequence of somebody's assumption

Outcomes:

#1: Examples – examples cover all the aspects of the user story plus those aspects that were not covered in the user story.

#2: The team have a common understanding of the business value of the user story

#3: The discuss activity might highlight that the user story is too big to be delivered, in this case the activity will produce a list of user stories and the examples for the first one that is taken into development.

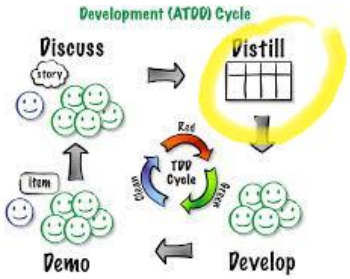
Next >

Special Comment(s):

User can select "Next >" for continue and click "ATDD Cycle" graph to get back to the initial screen (05.02.01.00)

Page Title:	Page no. 05.02.01.02	SME's name:
Date Designed:		Date verified:

Design



Distill

Required Artifact: Examples

Format: Pair programming

How it works: Now that we have the examples written down, we can transform them into tests in a format that works with our test automation framework. There are a variety of test automation frameworks that support defining the tests in advance of the implementation including Jbehave and Cucumber.

Tests will be written using the Given When Then format. Tests will cover all the examples that were identified as result of the Discuss activity. Extra tests could be added based on the improved understanding of the business goal.

Outcomes

#1: Tests – The Tests cover all the aspects of the examples plus those aspects that were not covered in examples that were uncovered while writing the tests.

#2: The tests will be written in English so that every team member is able to understand and give feedback. The Tests represent the blueprint (documentation) for what we will eventually deliver. The tests will be highly visible and easily accessible at any time.

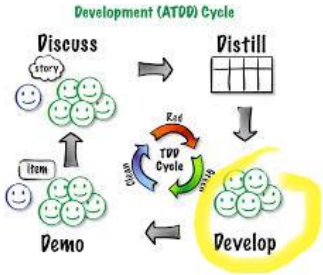
<Back | Next >

Special Comment(s):

User can select "Next >" for continue or "< Back" to previous screen or click "ATDD Cycle" graph to get back to the initial screen (05.02.01.00)

Page Title:	Page no. 05.02.01.03	SME's name:
Date Designed:		Date verified:

Design



Develop

Required Artifact: Tests

Format: Pair programming or Single developer writing code + Code Review

How it works: When implementing the code, the developers are following a test-first approach, they execute the tests and watch them fail. They will write the minimum amount of code required to get the acceptance tests Green. Once the acceptance tests are green he will manually verify that everything hangs together and will call another Developer or a Tester to perform Exploratory Testing. Once exploratory testing is completed and any defects fixed the user story is done and working software is ready to be delivered. While coding the developer might identify scenarios that were not identified earlier and add tests for them. Such tests need to be added to the previous set and shared with the rest of the actors. If the new scenarios identified represent a large amount of work a decision might be made that pushes the new uncovered scenarios to a subsequent user story or we could decide to deliver them.

Outcomes: Working software + more comprehensive tests

<Back | Next >

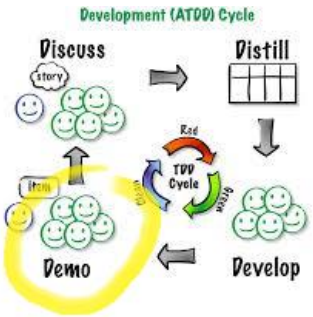
Special Comment(s):

User can select "Next >" for continue or "< Back" to previous screen or click "ATDD Cycle" graph to get back to the initial screen (05.02.01.00)

Page Title:	Page no. 05.02.01.04	SME's name:
Date Designed:		Date verified:

Design

Demo



Required Artifact: Working Software

Format: Meeting with large monitor

How it works: Before organizing a Demo the development team needs to be sure the user story adheres to the definition of done. One very good practice is to create a demo script in which the demo facilitator writes down the steps to follow in order to demonstrate the user story business value to the product owner.
The demo should be an occasion for the development team to be proud of what was delivered.
The product owner will be able to use the Tests to validate all the required functionality has been delivered. At the end of a successful Demo, the product owner will accept the original User Story through the business value demonstrated by running the tests.

Outcomes: Business value

<Back

Special Comment(s):

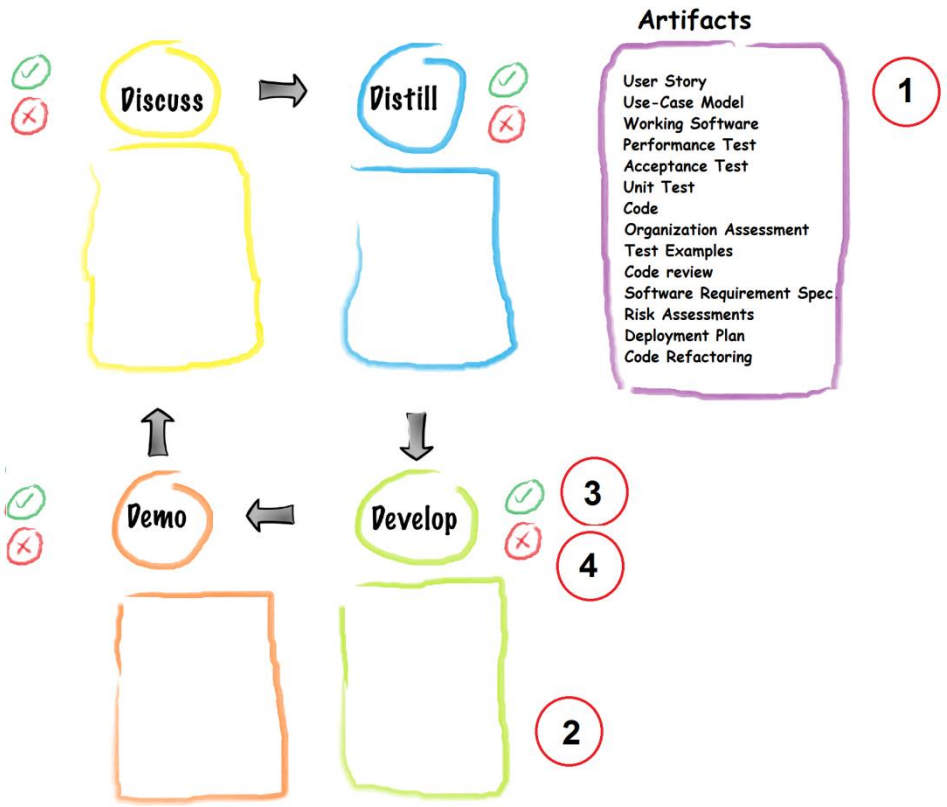
User can select "Next >" for continue or "< Back" to previous screen or click "ATDD Cycle" graph to get back to the initial screen (05.02.01.00)

ACTIVITY:

Page Title:	Page no. 05.03.01.00	SME's name:
Date Designed:		Date verified:
Design		

ATDD Process Cycle

Activity: Map artifacts into correct stage.
You need to drag and drop activities from "Artifacts" Box into correct ATDD process. If an artifact does not related to ATDD process, place them in "Not Related to ATDD" Box.



Special Comment(s):

User can select and drag and drop artifacts for "Artifacts" Box into the correct cage to get marks

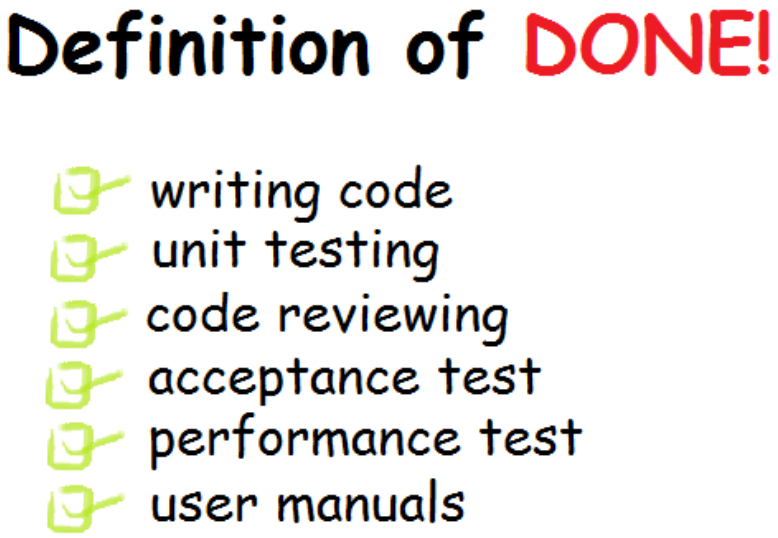
Should be able to select "Artifacts" from 1

Should be able to Drag selected artifacts into chose cage 2

If the Artifact is not belong the Drop cage, it should go back to the "Artifacts Box" and

Indicate "Incorrect Move" and Should increase the Wrong count 4 by 1

If the Artifact is placed in correct cage, Right count 3 should increase by 1		
Page Title:	Page no. 05.04.01.00	SME's name:
Date Designed:		Date verified:
Design		



Special Comment(s):

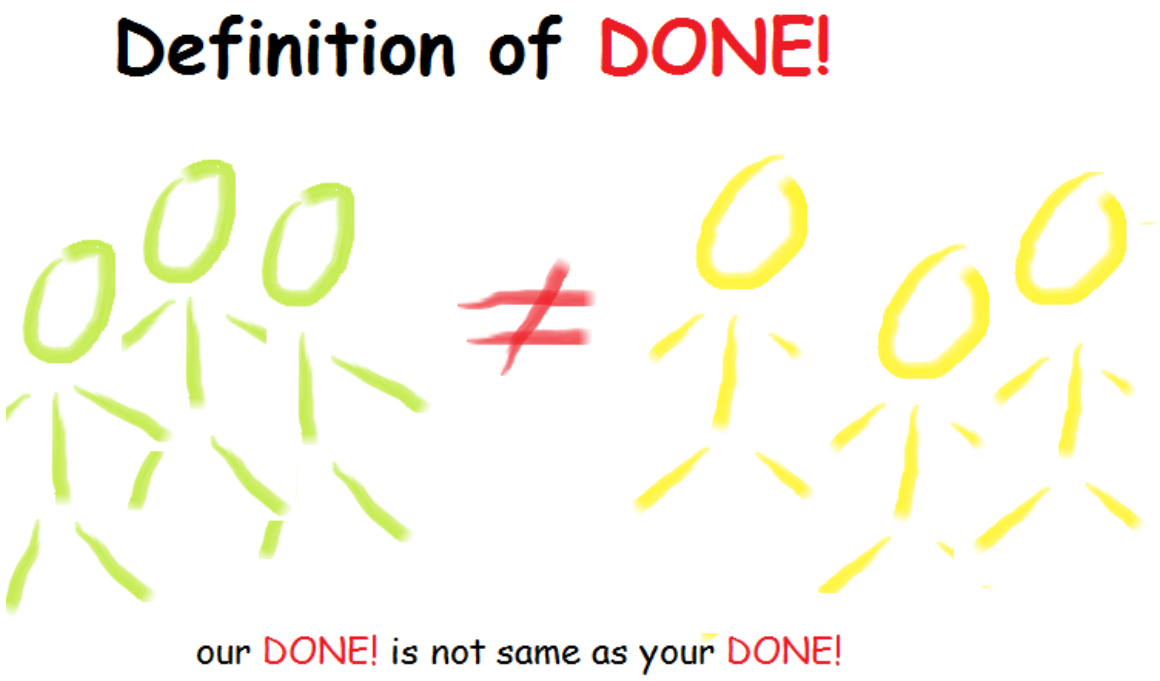
Narration:
"Definition of Done! Let's look at what's DONE means. For Agile teams that done means, nothing more needs to be done for a piece of work to be taken into production."

"Definition of done is, actually a checklist, checklist of activities required to produce complete software. Activities such as writing code, unit testing, code reviewing, acceptance testing, performance testing, user manuals, etc.
This check list allows the team to focus on what must be completed in order to build software. While eliminating wasteful activities that only complicate software development effort. you can think of the definition of done as an extra set of acceptance criteria
That is rubber stamped onto each and every user story."

Notes: each check box should appear sequentially while reading the check list items

Page Title:	Page no. 05.04.01.01	SME's name:
Date Designed:		Date verified:

Design



Special Comment(s):

Narration: "Definition of done is unique to the team. One team's DoD can't be applied to another team. and even team's
Definition of done won't remain the same throughout the lifetime of the project. It will get evolved with the time.
For an example, team might not be able to do so much automate testing when first starting out. but, hopefully they
Would add that to their definition of done over time."

Page Title:	Page no. 05.04.01.02	SME's name:
Date Designed:		Date verified:

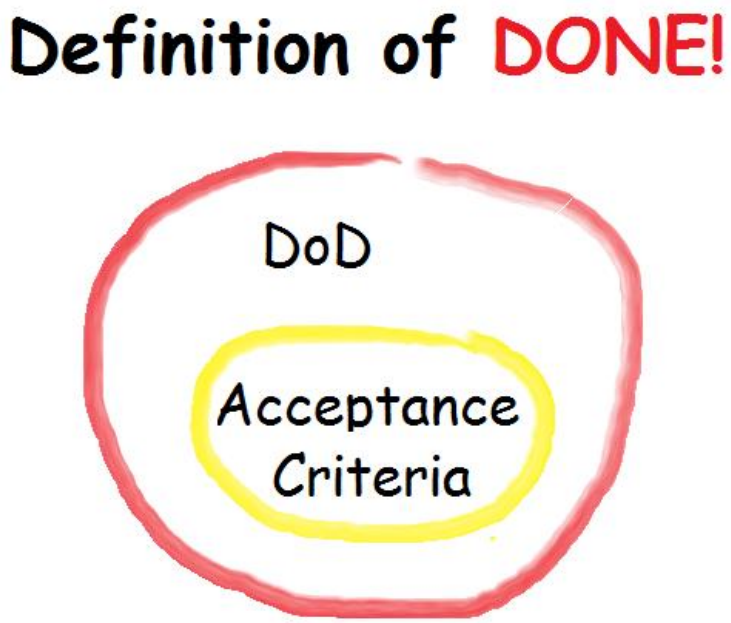
Design

Special Comment(s):

Narration: "However it seems Acceptance criteria and Definition of Done are same, actually they are not!"

Page Title:	Page no. 05.04.01.03	SME's name:
Date Designed:		Date verified:

Design

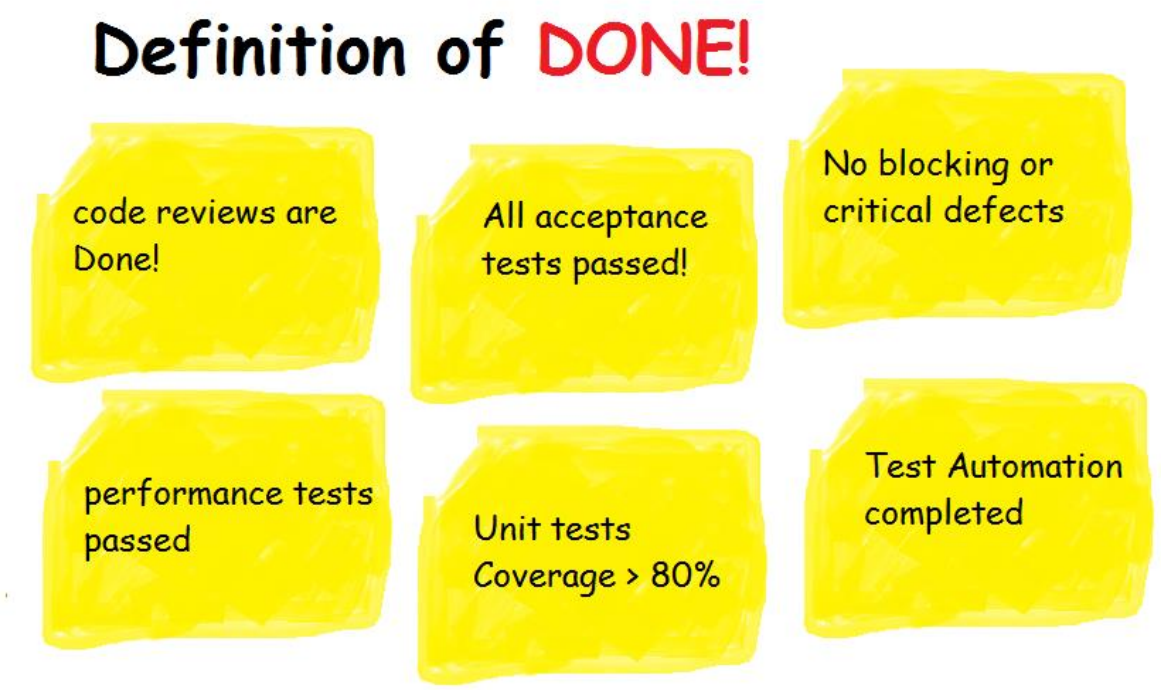


Special Comment(s):

Narration: "Definition of done is generic and applicable to all stories, while acceptance criteria is specific and different
For different user stories. but one item in the definition of done can be something like, "Do the user story meet the acceptance
Criteria?"

Page Title:	Page no. 05.04.01.04	SME's name:
Date Designed:		Date verified:

Design



Special Comment(s):

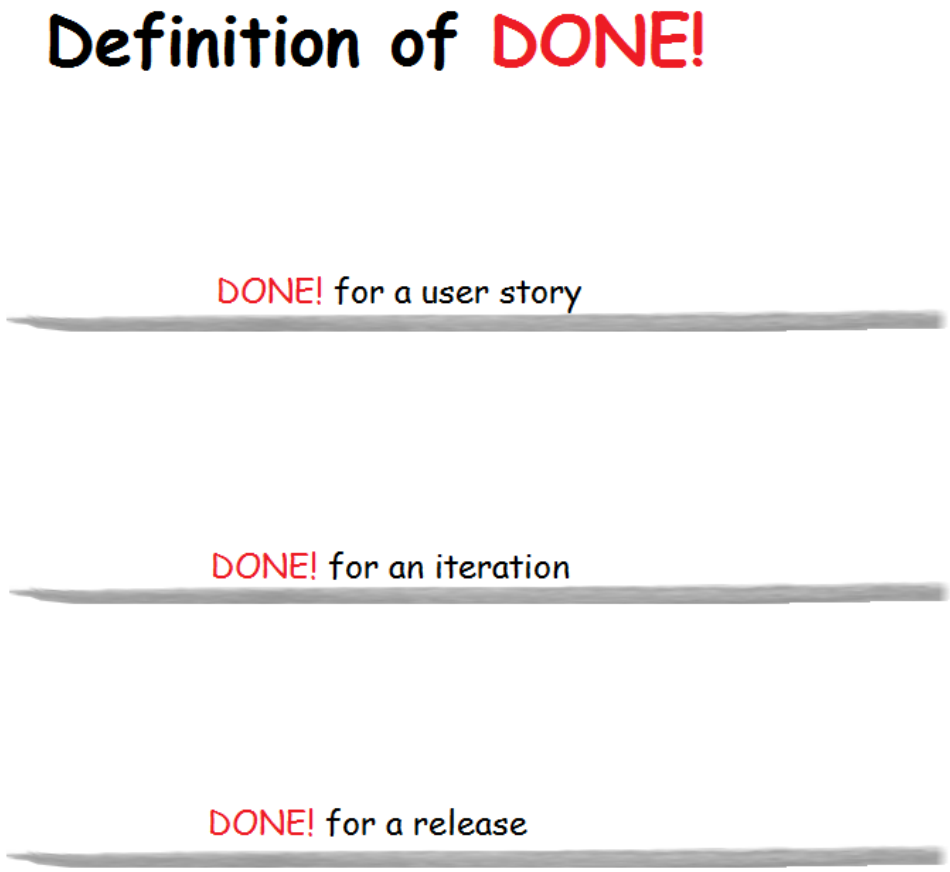
Narration: "so how we create Definition of done

As a team we should gather and discuss what DONE is mean to us. Team should include everyone. product owner, business
Analysts, developers, testers. team should discuss in a workshop meeting, usually time-boxed to 30 minutes or 1 hour, and try to identify and write down
All of the work necessary for a release. Write each item on a separate post-it note. and product owner needs sign of each
Work item, so it can be considered as an agreement. "Code reviews are done"
"

Note: as speak of each work item, post-it should appear in the screen. first sample is given for "Code reviews are done"
"Performance tests passed",
"All acceptance tests passed", "Unit tests code coverage is more than 80 percent",
"No more than 5 open defects", "No blocking or critical defects", "Test Automation is completed" are few examples for work items."

Page Title:	Page no. 05.04.01.05	SME's name:
Date Designed:		Date verified:

Design



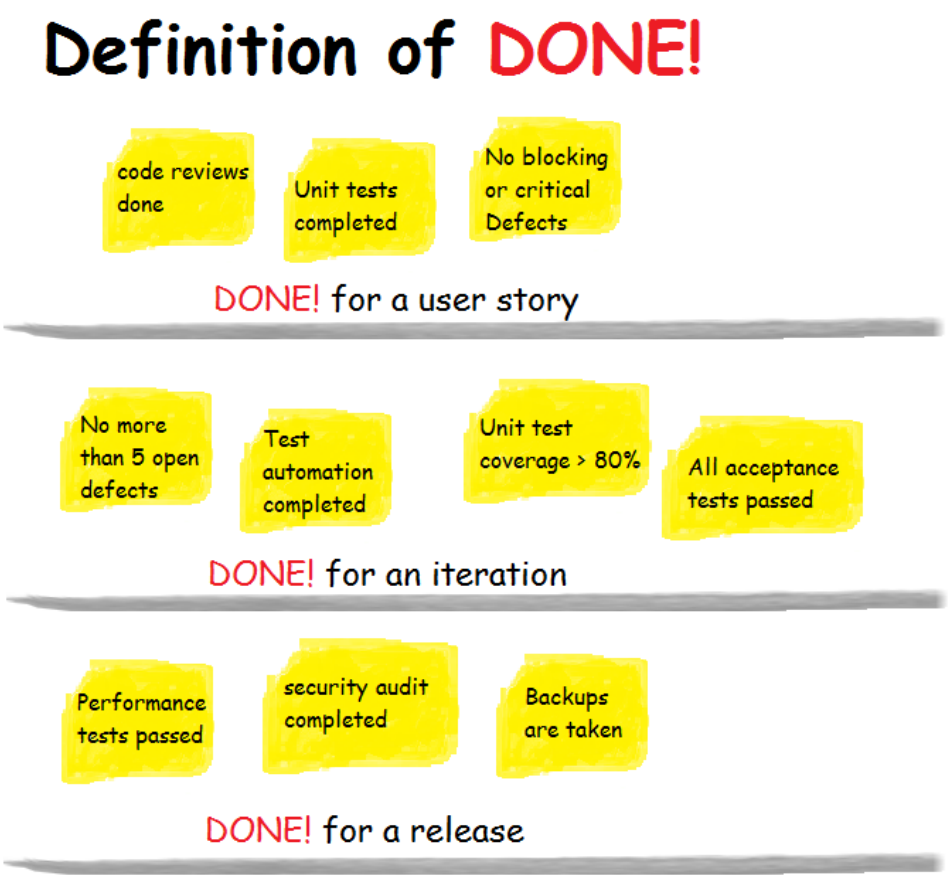
Special Comment(s):

Narration: "Then we need to categories them into 3 groups. One is "Done for a USER STORY", second one is "Done for an ITERATION", and last one is "Done for a RELEASE".

"

Page Title:	Page no. 05.04.01.06	SME's name:
Date Designed:		Date verified:

Design



Special Comment(s):

Narration: “"Code reviews are done", "Unit tests are done", "No blocking or critical defects" can be categorized into "Done for a User story". If these work list items are met for a user story, we can simply say, this user story is done.
"No more than 5 open defects", "Test Automation is completed", "All Acceptance tests passed", "Unit test coverage is more than 80 percent"
Can be categorized into "Done for Iteration". So when those work list items completed we can say we are done with the iteration.
Work items like "Performance tests passed", "security audit completed", "Backups are taken" can be categorized into "Done for a Release". once those activities completed we can say we are DONE with the release”

Page Title:	Page no. 05.05.01.00	SME's name:
Date Designed:		Date verified:

Design

Benefits of ATDD

For the Business and Team

- ❑ Improves Communication and Collaboration
- ❑ Better definition of done!
- ❑ Fast customer feedback
- ❑ Significantly less bugs
- ❑ Reliable automation
- ❑ Accurate documentation
- ❑ Changes are easy
- ❑ Happy customers!

Special Comment(s):

Page Title:	Page no. 05.05.02.00	SME's name:
Date Designed:		Date verified:
Design		

Benefits of ATDD

For developers

- ❏ No more unclear requirements
- ❏ Clear focus: make the test pass!
- ❏ Easy to keep the code clean
- ❏ Less bugs!
- ❏ Less debugging

Special Comment(s):

Page Title:	Page no. 05.05.03.00	SME's name:
Date Designed:		Date verified:
Design		

Benefits of ATDD

For testers

- Big impact on quality!
- No tedious test cycles
- More time for exploratory testing
- No more "it's work on my machine!" scenarios

Special Comment(s):

Page Title:	Page no. 05.06.01.00	SME's name:
Date Designed:		Date verified:

Design

Challenges in ATDD

- Works best in agile environment
- Writing good scenarios takes practice
- Poorly written tests can lead to higher test maintenance cost
- Treat test automation code like production code
- Requires high business engagement and collaboration

Special Comment(s):

Page Title:	Page no. 05.07.01.00	SME's name:
Date Designed:		Date verified:

Design

Cucumber & Gherkin Language

Cucumber is a software tool for testing other software. It's one of the least technical tools, so everyone can use it without much trouble.

Gherkin is the format for cucumber specifications. Technically speaking it is line-based language with a well-defined syntax, but at the same time it's so simple, that you don't have to know programming in order to use it

Cucumber

Special Comment(s):

Page Title:	Page no. 05.07.02.00	SME's name:
Date Designed:		Date verified:

Design

Gherkin Syntax

Gherkin documents are stored in regular text file with .feature file extension. here's a sample feature file

```
Feature: User Registration
Check for home page
See of the registration is working
Also verify if the register user is displayed

Background:
Given: Clear already created user before begin

Scenario: Register user with minimal password combination
Given I've opened the website
And I'm in the homepage
When I click the register link
Then I should see the register page
And I fill the form with details
|user name | password | cPassword|
|lahiru | abc@123 | abc@123 |
```

Special Comment(s):

Page Title:	Page no. 05.07.03.00	SME's name:
Date Designed:		Date verified:

Design

Gherkin Keywords

- Feature
- Background
- Scenario
- Given
- When
- Then
- And
- But
- *
- Scenario Outline
- Examples

helper keywords

- """ (doc string)
- | (data tables)
- @ (tags)
- # (comments)

Special Comment(s):

Page Title:	Page no. 05.07.04.00	SME's name:
Date Designed:		Date verified:
Design		

Gherkin Keywords

`Feature` - Each Gherkin file begins with the Feature keyword. This keyword doesn't really affect the behavior of your Cucumber tests at all. It just gives you a convenient place to put some summary documentation about the group of tests that follow. In valid Gherkin, a Feature must be followed by one of the following: • Scenario • Background • Scenario Outline

`Scenario` - To actually express the behavior we want, each feature contains several scenarios. Each scenario is a single concrete example of how the system should behave in a particular situation. If you add together the behavior defined by all of the scenarios, that's the expected behavior of the feature itself.

Special Comment(s):

Page Title:	Page no. 05.07.05.00	SME's name:
Date Designed:		Date verified:

Design

Gherkin Keywords

`Given, When, Then` – we use `Given` to set up the context where the scenario happens, `When` to interact with the system somehow, and `Then` to check that the outcome of that interaction was what we expected

```
Scenario: Successful withdrawal from an account in credit
Given I have $100 in my account # the context
When I request $20 # the event(s)
Then $20 should be dispensed # the outcome(s)
```

Special Comment(s):

Page Title:	Page no. 05.07.06.00	SME's name:
Date Designed:		Date verified:

Design

Gherkin Keywords

`And` , `But` – Each of the lines in a scenario is known as a step. We can add more steps to each `Given`, `When`, or `Then` section of the scenario using the keywords `And` `But`

```
Scenario: Attempt withdrawal using stolen card
Given I have $100 in my account
But my card is invalid
When I request $50
Then my card should not be returned
And I should be told to contact the bank
```

Cucumber doesn't actually care which of these keywords you use. the choice is simply there to help you create the most readable scenario

Special Comment(s):

Page Title:	Page no. 05.07.07.00	SME's name:
Date Designed:		Date verified:
Design		

Few Examples - #1
(Purchasing from online store)

```
Feature: Feedback when entering invalid credit card details
 In user testing we've seen a lot of people who made mistakes
 entering their credit card. We need to be as helpful as
possible here to avoid losing users at this crucial stage
Of the transaction.

 Background:
 Given I have chosen some items to buy
 And I am about to enter my credit card details

 Scenario: Credit card number too short
 When I enter a card number that's only 15 digits long
 And all the other details are correct
 And I submit the form
 Then the form should be redisplayed
 And I should see a message advising me of the correct
number of digits

 Scenario: Expiry date invalid
 When I enter a card expiry date that's in the past
 And all the other details are correct
 And I submit the form
 Then the form should be redisplayed
 And I should see a message telling me the expiry date
must be wrong
```

Special Comment(s):

Page Title:	Page no. 05.07.08.00	SME's name:
Date Designed:		Date verified:

Design

Few Examples - #2
(Withdrawal from ATM)

```
Scenario: Successful withdrawal from an account in credit
Given I have $100 in my account # the context
When I request $20 # the event(s)
Then $20 should be dispensed # the outcome(s)

Scenario: Attempt withdrawal using stolen card
Given I have $100 in my account
But my card is invalid
When I request $50
Then my card should not be returned
And I should be told to contact the bank
```

Special Comment(s):

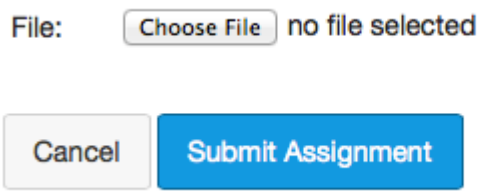
ACTIVITY

Page Title:	Page no. 05.08.01.00	SME's name:
Date Designed:		Date verified:
Design		

Assignment: (10 Marks)
Due date: 2016-08-10
No submission after: 2016-08-17
Late penalty: 40% will be reduced from the marks for late submissions

Look at the given user story and convert it into .feature file using Gherkin syntax. Upload the .feature file using "Choose file" button and submit the assignment.

As a shop visitor I want to collect books in my shopping basket
So that I can purchase multiple books at once.
Books can be added to the shopping basket
Books can be removed from the shopping basket
Shopping basket is initially empty
The same book can be added multiple times to the shopping basket



Special Comment(s):