**1.1 Introduction:**

Software engineers have introduce a lot of developing model with different process, workflow and roles in order to produce software with high quality. Generally quality here compose to two type: project’s quality which refer to issues that affect the project as whole such as (time management and cost management); product quality which refer to issues that is related with software product itself such as (high performance, reliable software, correctness and so). In balancing between project’s quality and product’s quality companies agree that improving the software processes is the solution for software against delivery delays, cost overriding, and quality flaws in the end products (R. A. Khan et al, 2000). These problems make software engineer starting to develop more flexible model with more flexible process which called Agile method, after the spread of software companies and the competition between its, each company wish to reach the market first so the demand on Agile methodologies is grown .The growth of using Agile placing on QA teams for faster delivery cycles is forcing many to replace the older monolithic development models with a more streamlined process (kristen Aebersold). However, pervasive method such as manual testing are keeping the teams for adopting flexible practices like continuous integration and continuous delivery and reaching true agility(kristen Aebersold) .Continuous integration is software developing practice in which small piece of code are verified using automated building and automated testing every time a team member make change in code repository .Continuous delivery is software approach in which software product’s is ready for deploying to the market at any times. In order to reach true agility we need to develop continuous integration continuous delivery pipeline to enable developing team to release constant flow of software updates into production to quicken release cycles, lower cost and reduce the risk associated with development (kristen Aebersold).

1.1.1 What is continuous integration?

“is a software development practice where software is integrated continuously during development”( M.Fowler, 2006) .In constant , some project have integrate the work of developers after amount of time in fact when the integration is late it possible to have a lot of line conflict in the code which increase development time in fixing this conflicts (Eero Laukkanen et al.,2015) .It’s better to integrate the code several time in a day by using share repository .In additional each time a code is integrated to the repository it trigger an auto building and testing to ensure that the code is still do what is intended from it while it’s changed and as soon as test fail it reported and fixed.

1.1.2 Continuous Delivery (CD)?

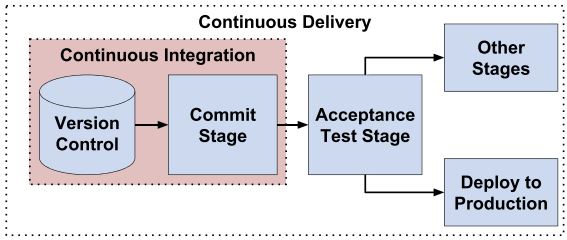
“is a software engineering approach in which teams keep producing valuable software in short cycles and ensure that the software can be reliably released at any time”( Chen , 2015a). Companies that have adopted CD have reported huge benefits , such as significant improvements in time to deployment, customer satisfaction, product quality, release reliability, productivity, efficiency and the ability to build the right product through rapid experiments (Chen , 2015a ; Leppanen et al , 2015). These benefits have motivated many companies to adopt CD (Lianping Chen, 2016). However, implementing CD can be quite challenging (Chen, 2015a; Leppanen et al., 2015; Claps et al., 2015). The following figure show the different between the CI and CD, which indicate that CI is a part of CD, but CD has much process. When a commit into CI happen it triggers an auto build and test to the code in the repository if the test is pass deploy release into production otherwise the developer recommended fixing it.

Figure 1: different between CI and CD

* 1. **Search problem:**

1.2.1 Current system example scenario:

In development phase a group of developer configure their development environment and setup repository for sharing their codes, after that they starting develop, each developer has its own task which may has some dependency with other developer task. Each developer start developing and committing code into the repository so that other developer can see and use it, after the development group finish their module they send it for testing and waiting for the result of testing. A tester run all the test and report bugs back to the developers, developers start debagging and fixing bugs and send module back to testing, after the tester report no more bugs developer start developing anther module.

1.2.1.1 Scenario problem:

1. Time to find and fix bugs are maximized: testing is performing per module so the scope of searching for each bugs is all the module. Early generate bugs stay without fixing until the module is finished and tested so that the cost and the effect of fixing it is increase. Also other developer may dependent on this ill piece of code.

2. Deploy software has not full tested: in some cases adding small change after testing process make the cost-benefits of run all tests again decreased resulting in deploy some piece of code has not been tested as stated in Continuous Delivery: Overcoming adoption challenges paper(Lianping Chen ,2016).

3. Testing environment: developers spend 20% of their time setting up or fixing their testing environment. In addition because the testing environment is shared among teams, a team may wait for one or two weeks for a testing environment to be released by other teams as stated in Continuous Delivery: Overcoming adoption challenges paper (Lianping Chen, 2016).

**1.3 Objective**

* Automate the process from development to delivery by Build an automated pipeline starting development small piece of code and commit to share repository which pipe auto building and auto testing processes.
* Early bug detecting by running the test per small piece of code’s change and send feedback to developer.
* Grantee that each piece of code has been verified by its targeted test.
* Performing the testing process in virtual environment that is setup to act as same as production environment.
* Increase productivity.
* Building valuable software.

**1.4 Expected outcome**

* Reliable software release of software products: by verifying each change has been tested.
* Increase number of release.
* There are no problem with software in production environment.

**1.5 Scope**

* Building automation.
* Testing automation.
* Integrated with GitHub as version control.
* Supports JavaSE as programming language and desktop applications only.
* Supports JUnit as java testing library.
* Generate JUnit core report.
* Integrated with Docker as container.