Continuous Delivery is a popular strategy adopted by most modern software development industries. It is, as described by Jez Humble and Dave Farley (2010), the process of get changes into production or into the end users in a reliable way. Changes by developers include adding new features, change configuration, bug fixing, and they have to be broken down into smaller parts that can be delivered respectively [1]. Getting changes into production requires the code to be sustainable, which means that the code should always in a deployable state. To achieve this, it requires automated software building, testing, and deploying, but we only deliver what is potentially profitable and beneficial to the business. Adopting Continuous Delivery allows developers to get feedback from market in time, which improves software quality and releases better products.

In our case study of Expedia, we had a deep conversation with an ex-developer from Expedia advertising group, to see practically how Continuous Delivery helps development work in big IT companies. Expedia develop team mainly work under principle of Scrum, which is a development methodology for agile software development, requires a PO, a Scrum Master, and development team. It aims at responding to emerging requirements, and to adapt to evolving technologies and changes in market conditions [2].

As for project management part, some teams in Expedia rely completely on Jira, an issue and project tracking software, while others make a combination of Jira and Mingo, which is also a project management software. He introduced how development teams make use of Jira as an example. Since teams are facing various requirements every day, both from project details or software developers. Requirements contain a general problem description, an expectation for the problem, or a specification of outcoming product. Different requirements can be categorized into epics, stories, or tasks, based on size and complexity of requirements in descent order. Each of epics, stories, and tasks is assigned to different Scrums, through the “tasking meeting”, which allow developers understand the task size, what is requested and what is expected.

A fixed bi-weekly Sprint is also a component in Scrum in Expedia. Previously, because of the fact that Expedia website updates have a weekly release strategy, any bug fix and small changes have to be conduct at the end of that two weeks, and developers usually address everything discussed on Sprint within a small period just before the deployment, which is not efficient for product development. Specifically, He told that the deployment day for Expedia is Wednesday weekly, and there is a “cut-off time” on Tuesdays that every change made by that time will be deployed to the latest version of their website. This strategy has an obvious drawback that if some new features have to be added, or some emerging bug fix need to be conducted, which is so important that cannot wait until next deployment day, then it becomes a catastrophe. That bug needs to experience a complicated process of fixing, integration, approved by manager and project director, and finally build the implementation from head to tail before deployment. Normally small sub-deployment won’t be approved by director, instead the team will perform some security tricks such as shut down that particular service for a week by switching flags.

After using continuous delivery, such bug fixing problem becomes a lot easier. Take their mobile web page for instance, developer only needs to push a change to the git repository and waits until that change to be deployed. The build pipeline can automatically monitor every commit that developer pushes and begin building. Afterwards, it will conduct automated tests that the team set in advanced, such as unit test, integration test, and squash test (simulating user clicks). Finally, it will perform automated deployment, and the change will be push to cloud and allow user to see the latest Expedia mobile page immediately. It is apparent that using changing into such software development strategy is potentially risky, as if someone’s push causes small problems, it will directly deliver to user’s hand and more severely affect functionality from other parts. Therefore, it requires every developer to bear a high responsibility that every single commit they push represents the entire Expedia company. Specifically, it requires the code in their hands should be production ready at any time and take ownership of what they are doing. There are also push review team to test code in an extremely careful manner before pushing to examine potential possibility about performance decay. Moreover, there are various monitoring parameters recorded on pipeline, such as webpage loading speed. If such parameters decrease dramatically after a particular push, there will be a caveat prompted to reminded that developer to revert if necessary. If every developer can be carefully enough and make sufficient test before push and every team works stably, then it is a highly efficient and robust developing system.