Essay 3- Software Engineering (CEN5035)

Ken Johnston's MVQ Testing Model and EaaSy Framework

Name: Vikas Chaubey, **UFID:** 3511 5826, **Email:** vikas.chaubey@ufl.edu

In the lecture and the blogpost written by Mr. Ken Johnston's he has talked about importance of employing new software methodologies like "Everything as a service (EaaSy) framework" and "Minimum viable quality (MVQ) model". According to the author with the advent of cloud technology, the business model for software has also changed, most of the software systems are shifting towards service models, and instead of standalone software products , consumers are frequently buying these software services such as storage, operating systems, communication etc. since the software market is evolving , user expectations from the software providers are also changing , now days users expects new innovative features and software updates more frequently at rapid pace from software providers, this in turn is forcing the software teams developing the software products to ship the code more frequently. Another important factor to be considered is that most of the current devices whether it be hardware or software, now have capability to connect to internet hence it is very easy to update the software associated with these devices. According to the Mr. Ken Johnston to keep up with this rapidly changing pace in software industry, we need to incorporate new frameworks and models in the software development and testing process because as the software market is evolving older methodologies might not prove to be as effective as per requirement and hence we need to either update them or incorporate new methodologies into development process. That's where "Everything as a service (EaaSy) framework" and "Minimum viable quality (MVQ) model" come into picture.

EaaSy framework tries to leverage benefits associated with service model architecture such as loose coupling, speedy delivery, frequent updates, bigdata benefits and connected ness of devices in order to achieve satisfactory quality and faster shipping of the software updates. Everything as a service (EaaSy) is a Framework which incorporates Minimum Viable Quality (MVQ) model which states that the software could be shipped to the user with minimum required quality, with enough testing initially at that point of time and then bug fixes could be shipped in form of software updates at rapid frequent rate. The frequent rate of bug fix updates keeps the user satisfied with the software product but also helps in faster shipping of software without having extra overhead of detailed testing initially. The Software development teams which provide support for the software for long tenures have to rely heavily on traditional means of testing as well as automation testing, In various cases the automation testing code base is bigger and larger than actual functional modules hence it's a management overhead, Mr. ken Johnston further explained that in 15 years of his real life software practice in industry he has often experienced that despite maintaining a large testing code base to identify bugs, many times software teams still ship the software product even if bugs are identified , this is done to meet the shipping timelines. Hence a thorough and detailed testing, identification and resolution of bugs serves little purpose in initial stage and is no more an efficient approach if the software updates have to be shipped at faster rates. According to the lecture, as the software market is evolving, the perception of software quality is also evolving, as software systems are moving towards service architecture, in future the quality will also shift towards being a data driven service itself. These are some important factors which EaaSy framework incorporates. It is a model which leans towards lean data driven testing and reduces the reliance on heavy automation testing to facilitate fast shipping. To do that EaaSy makes use of Minimum viable Quality (MVQ) model. In conclusion with EaaSy framework a software product can leverage benefits of service architecture model and could be managed as a service.

The use of EaaSy framework directly translates into the software system having five key capabilities which are actually driven from the service architecture model. The EaaSy model is based on these five fundamental pillars which are componentization, flighting for quality and experiments, continuous delivery, rich fast data for quality of service(qos) and quality of experience (qoe) and user segmentation. According to the lecture in order to implement EaaSy framework, a software system should exhibit these five capabilities. Once it incorporates these capabilities the software system behaves like a service. Along with these capabilities author in the blog post also stresses that the system should be able to be updated on regular basis. with that approach software teams can update and ship the updates for the software system in much faster fashion without any hassle.

The most important capability parameter in EaaSy framework is “componentization” of the software system. In order to understand why this is so important, we need to look at the service architecture model. The main feature of service architecture that makes it so efficient is its modularity, that means each functionality in the system exists and works as an independent module , The key advantages of using this approach is that these modules could be deployed faster, a module could be updated without effecting the existing functionality of other modules another very important advantage is localization of bugs, a bug occurring at any specific location does not impact the functionality of other modules , with separate modules it becomes easy to locate ,identify and resolve the bugs without effecting the other parts in the system. Componentization in EaaSy framework follows the same concept, and it states that a software system should implement separate modules for different functionalities. The EaaSy framework suggests that for componentization to work effectively the system has to be both forward and backward compatible.

The next important factor is "continuous delivery". The componentization feature can only work effectively and efficiently if the software system is associated with continuous delivery. The EaaSy framework puts emphasis on frequent release of software updates for new incorporated features and bug fixes as soon as they are available. This is done because it is always a better approach to update a software in small increments more often than to update the software with large updates with long time interval gaps in between. Small incremental updates can easily be tested faster for bugs, even if the bugs create any system wise problem the update could be easily retracted, and the software system could be put back to its previous state without much effort. This could be achieved using continuous delivery (CD), The goal of continuous delivery is to make the modified software production ready with thorough testing once it clears continuous integration automation testing. If the new modifications are bug free then a deployment pipeline is triggered, and the new build is deployed to testing or production environments. These deployment pipelines could trigger automatically on the basis of artifact availability, or on scheduled basis or manually whenever required.

According to the lecture the "User Segmentation" parameter in the EaaSy framework is the most important one. User segmentation is the process of dividing the software users into groups or segments according to some characteristics. This is done in order to determine different risk pools associated with these segments. If some kind of user segmentation is not implemented, then what it means is that when the code is released for the first time it has to be up to the mark in terms of quality because it will then be used by users for which we have not defined any fault tolerance boundaries , hence without a user segmentation method in place we have to assume that our code has to be release ready and could be used by end user without facing any bugs. This is a very strict guideline and it enforces heavy testing of the software system to ensure it does not have bugs in the initial release phase itself. This goes against the principles of EaaSy framework which says that first release should be viably qualitative initially. By defining user segments, we can define fault tolerance boundaries for each segment and when software crosses the next segment boundary it has to maintain a certain acceptable quality level, this way software could be released initially with acceptable quality and as it crosses different user segments its quality increases incrementally. Author in the blog post has given an example of the rings of risk model to implement user segmentation which is followed in the Microsoft corporation, where the development team is the core of the system and has highest tolerance for bugs , the next further layers boundaries define levels where software is used at company level and then by beta users and finally by end users. At each passing boundary the fault tolerance for the software system decreases and adds to systematic development of bug free software.

Another important feature of EaaSy framework is "flighting", flighting is a concept or capability used to route and mainly to turn code paths on and off via configuration settings in the cloud. This feature helps when a new feature has to be experimented to check user's response, in that case a new code changes could be activated and deactivated by simple configuration changes. For example, in A/B testing, flighting is used to check the response of users for two different user interfaces, one which appeals more to the users is kept. Flighting could be used to manage new code access and exposure to the users which improves roll back. It can also be used for user segmentation.

Lastly the EaaSy framework states that a software system should incorporate "Rich fast data" for quality of service(qos) and quality of experience (qoe). for any software system the user response and engagement are the key indicators of software usability and performance, if any new code update or shipment decreases the user interaction with the system then it's a clear sign that either code update has more bugs or usability of the product has decreased with new upgrade. Hence, we can see how data is very important in order to maintain the quality of the system or even improve it near future. RFD could be used to mitigate the risks and respond to the defects occurring in the system fast. This results into continuous improvement of the software system.

The main reason behind adopting DevOps in development process is to make deployment and operation of a software application efficient, we can clearly see the EaaSy framework closely follow the principles of DevOps in order to make the applications more efficient and frequently deliverable. The most significant and visible example is use of "continuous deployment.", just like DevOps operations EaaSy methodology focusses on frequent release of software updates for new incorporated features and bug fixes as soon as they are available. The use of continuous deployment is done so that "componentization" made in the system could be fully leveraged. Without CD componentization cannot be effective. "flighting" is a technique which is used in DevOps very frequently to run different code profiles in different environments for example activating the "test", "production" profiles while testing or deploying the software. Similarly, "user segmentation" is could be implemented by using DevOps methodologies so that each user base gets the respective version of the software system for example, beta users can get beta version while end users get the most stable version of the software application. We can clearly see that EaaSy framework closely incorporates the DevOps mythologies in order to achieve maximum efficiency while developing and shipping software applications.

References:

1) https://www.linkedin.com/pulse/future-quality-easy-eaasy-mvq-ken-johnston
2) https://www.dropbox.com/s/b8z7ekqusb3cjml/CEN6070_30-hh.mp4?dl=0
3) https://docs.microsoft.com/en-us/windows insider/flighting
4) https://www.synopsys.com/glossary/what-is-continuous-delivery.html
5) https://simplicable.com/new/service-architecture