Berk Yıldız

21502040

Section 2

CS453 – Application Lifecycle Management – Homework 1

1. **Explain point to point and model-based integration approaches. Which one is easier to scale and why?**

   Point to point integration meant that a project or repository is linked to all other projects or repositories in an IT system. Simply each project pair is mapped individually in point to point approach. It can be used in small infrastructures however it is very difficult to maintain point to point integration in large infrastructures because of the huge complexity. In model-based integration there is a project grouping which consists of projects and all the project groups have connection with a common defect model instead of having connection with other projects. Defect model provides the communication and translation between project groups.

   Model-based integration is easier to scale because it does not include complexity and tight coupling of point to point integration. Maintanence of model-based integration is easier and it is always in ready position for further changes however cost of a little change in point to point integration is seriously higher. So model-based integration presents a more regulated and steady environment for repositories and projects which directly have affects on scalability.

2. **How does model-based integration reduce the integration overhead? Support your argument with an example.**

   Number of mappings in point to point integration is much more than model-based integration. The reason for high overhead in point to point integration is the number of mappings and complex dependencies between repositories or projects because each mapping is a cost of work and tight coupling causes problems for further changes. Model-based integration minimizes the number of mappings and prevents the tight coupling problems of point to point integration. For example let say there are 3 repositories which consists of 5 projects each. When there is a need for change in 4 projects from different repositories, cost of maintanence overhead would be very high for point to point integration because of the tight coupling. However changes can be

done very easily in model-based integration without thinking of dependencies because each project group already connected to common defect model.

3. **Assume that there exists a software project where the developer decided to use a model-based integration approach. Knowing that this solution was not beneficial for them considering the cost of service from a model-based integration tool. What can you assume about this project?**
   It is a small-sized software project with small number of dependencies. So setting model-based integration can be expensive and not efficient when we compare with the point to point integration.

4. **What does traceability mean in software engineering? What are the limiting factors on traceability for a modern software project?**
   Traceability is related with product validation which includes three important business management processes: quality management, change management and risk management. Traceability is important for delivery of the software because it provides ability to keep track of processes and outcomes of development lifecycle. A basic Requirement Traceability Matrix or spreadsheet can limit the traceability of software project becuase it would not cover all factors which is related with traceability. Also having large number of dependencies would help to limit traceability.

5. **What are the components of a software value stream? How does linking these components with automation generate business value in a large scale software project?**

   There are two components of software value stream: downstream which consists of creation, build and deploy and upstream which consists of product request, planning and design. Linking these components with automation would avoid manual work which can reduce human error and restrain waste of time due to duplicate entries. Moreover, getting rid of manual operations would provide efficiency and enables better management of the delivery process becuase large scale software projects would include complexity in all manners.