Why should developers use configuration management tools to manage their software programs? What can go wrong?

Answer: Configuration management applied to an application, provides us with the application's transparent understanding and control of its functional and performance components.

Using configuration management developers can easily maintain the flow of making changes to the build, releasing it and deploying it so that the customer can use the product without any issues.

It makes sure that the documentation that includes, requirement analysis, planning, designing, and testing, is accurate and consistent with the actual working of the application. Thus, it can be verified that the application is working as intended, and can be later easily identified in complete detail to support future changes in the application build.

Configuration Management provides the application development, following benefits: (1) Cost reduction (2) Reliability (3) Organization

But, sometimes the incorrect use or implementation of configuration management may in turn cause unwanted problems, such as failure of application. Following are a few examples:

- Changes not documented or part substitution in final product can result in life cycle management issues [1].
- When root cause analysis is not performed, it may lead to recursive scrap and repair [1].
- Lack of baseline control results in vaguely defined requirements, analysis, and CM implementation [1].
- Untrained developers performing CM functions, results in a lack of CM control [1].

Explain the difference between continuous integration, continuous delivery, continuous deployment, in your own words.

Answer: **Continuous Integration:** It is a practice that helps to make preparing a release easier. For continuous integration, one must write automated tests, for any new feature, bug fixing or improvement. Whenever new changes or commits are made to the application, CI makes sure to automate the testing process to avoid application failure. Developers keep committing their changes to master branch frequently, which is tested and validated to make sure there is no issues during application release.

Some of the benefits of continuous integration are: (1) Obvious and simple bug can be avoided before shipping the product to production (2) Making the build release process easier.

Continuous Delivery: It extends Continuous integration. The delivery needs to be automated such that once started it does not require any manual interference at any point of time. So that, one can release new changes to the clients quickly. Continuous delivery is basically automating this release process. It gives you the freedom to decide how frequently to release the build.

Some benefits of Continuous delivery are: It removes the complexity of deploying software. One can release the product more often.

Continuous Deployment: Using continuous deployment one can make sure that there are no issues at the time of release. since releases become less risky and easier to fix. Also, on the client side, continuous stream of product improvement can be seen. It accelerates the feedback loop with the clients. In short, the changes to the build are automatically tested and deployed to production environment.

References:

[1] Configuration Management Theory, Practice and Application- Kim L. Robertson et al., Ch. 11