**Assignment 15.2**

**Problem Statement:**

Explain the working and the differences between Maven, Gradle and SBT in detail.

**Maven:**

Maven was released in 2004. Its goal was to improve upon some of the problems developers were facing when using Ant.

Maven continues using XML as the format to write build specification. However, structure is diametrically different. While Ant requires developers to write all the commands that lead to the successful execution of some task, Maven relies on conventions and provides the available targets (goals) that can be invoked. As the additional, and probably most important addition, Maven introduced the ability to download dependencies over the network (later on adopted by Ant through Ivy). That in itself revolutionized the way we deliver software.

However, Maven has its own problems. Dependencies management does not handle well conflicts between different versions of the same library (something Ivy is much better at). XML as the build configuration format is strictly structured and highly standardized. Customization of targets (goals) is hard. Since Maven is focused mostly on dependency management, complex, customized build scripts are actually harder to write in Maven than in Ant.

Maven configuration written in XML continuous being big and cumbersome. On bigger projects it can have hundreds of lines of code without actually doing anything “extraordinary”.

Main benefit from Maven is its life-cycle. As long as the project is based on certain standards, with Maven one can pass through the whole life cycle with relative ease. This comes at a cost of flexibility.

In the meantime the interest for DSLs (Domain Specific Languages) continued increasing. The idea is to have languages designed to solve problems belonging to a specific domain. In case of builds, one of the results of applying DSL is Gradle.

**Gradle:**

Gradle is the youngest build tool of the three and its developers tried to combine Ant’s power and flexibility with Maven’s dependency management and conventions to create nice and shiny build tool. After several years of developers, Gradle v1.0 was released in 2012, and has quickly started to gain popularity. It’s developing really fast and already adopted by some big enterprises–Gradle, for example, was selected to be the build tool for Google’s Android OS.

One interesting point is that with Gradle, XML is not used anymore–instead, developers have a Domain Specific Language (DSL) based on the JVM language Groovy, which was invented so that developers could ditch the verbosity of XML and write more simple and clear statements. This has sparked a debate among users as to whether the standard, easily-understandable (but long-winded) style of XML is better or worse than DSL.

Gradle combines good parts of both tools and builds on top of them with DSL and other improvements. It has Ant’s power and flexibility with Maven’s life-cycle and ease of use. The end result is a tool that was released in 2012 and gained a lot of attention in a short period of time. For example, Google adopted Gradle as the default build tool for the Android OS.

Gradle does not use XML. As a result, Gradle build scripts tend to be much shorter and clearer than those written for Ant or Maven. The amount of boilerplate code is much smaller with Gradle since its DSL is designed to solve a specific problem: move software through its life cycle, from compilation through static analysis and testing until packaging and deployment.

Gradle effort can be summed as “convention is good and so is flexibility”.

**SBT:**

SBT stands for Simple Build Tool.

It is a general purpose build tool written in Scala for JVM developers. It borrows good ideas from other successful build tools like Ant, Maven, and Gradle.

1. Default project layouts
2. Built-in tasks
3. Plugin architecture
4. Declarative Dependency management
5. Code over Configuration: A DSL for build tool

Apart from the feature set mentioned above SBT also provides the following additional features:

1. Interactive nature: It isn't just a build tool, it also provides an interactive environment to work in.
2. Scala REPL integration

You can use sbt in two modes -- command-line mode and interactive mode. In the command-line mode, you run sbt task from your machine terminal. Once the task successfully finishes then sbt exits. For example, when you ran sbt about task, it printed sbt and build information on the console and then sbt exited and you were back to your terminal. In the interactive mode, you run sbt command and it launches a sbt shell. Inside the sbt shell session, you run sbt tasks.

**Differences between Maven, Gradle and SBT:**

SBT is very simple and it is focused on Scala it relies on Ivy for dependency management.

Maven it's a great build tool and it enables to control the entire software lifecycle with XML files. Using the Project Object Model you can intercept all points of the software lifecycle from compile to test, packaging and deploy.

Maven has its own dependency manager. In my humble opinion the bad issue in Maven is the XML syntax, writing a POM can be annoying and too much expensive. I think we cannot compare Gradle with Maven (or SBT). Gradle is built on top of Maven, Ant and Ivy. It uses Maven repositories.

Gradle doesn't use XML, it's a polyglot build tool. It combines the Ant API with the Groovy language to enable developers to write a build script with an intuitive DSL. With a few lines of code you can write a Gradle build script that can do the same things that Maven can do. With Gradle you can define your own task with the Groovy language and intercept programmatically your build execution. This functional approach is not for all developers, in fact Maven it's good if you don't want this behavior in your build environment. Both Maven and Gradle have plugins to integrate your build with technologies used in your projects.