**Plan driven versus agile development elaborated through concrete methods**

Plan-driven development is development where every process activity is planned at the beginning of the project and the progress is measured according to the plan. In the agile development the processes are planed incrementally and continues all the time until the software is done. As the agile development is much more flexible it is harder to measure the progress as the requirements may change very often however it is cheaper and easier to make changes in the software as it is being developed.

**Plan-driven development**

The best example of plan-driven development is the waterfall model. The waterfall model divides process activities in five phases:

* Requirements definition
* System and software design
* Implementation and unit testing
* Integration and system testing
* Operation and maintenance

Requirements definition is the phase where all the necessary requirements and details are discussed with the customer or system users to get a good idea about what has to be built and they serve as a system specification.

System and software design is the phase where all the documentation of systems architecture has been made. Everything is well documented and diagrams are made so the plan for software would be clear.

Implementation and unit testing is the phase where software implementation and unit testing is done. Implementation is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

Integration and system testing is the phase where all the units of program are integrated and tested as a complete system to ensure that all system requirements are met. After this phase the software is delivered to the customer.

Operation and maintenance is the phase with the longest life cycle. The system is installed and put in to use. All the errors are corrected that was not detected in development, unit improvement and enhancing the system’s services as new requirements are discovered.

Waterfall model consists of a lot of analysis and documentation which covers all the small details of how the software can be implemented. Each phase has to be done and “signed off” before next one starts. If a new requirements or information becomes available the previous stages should be modified to reflect the required system changes and must be approved by the customer and may delay the overall development. Customer and developers can make an agreement of freezing the requirements so no changes can be made. However this may cause problems which are left for later resolution which are ignored or programmed around and may lead to a badly structured system. It is also expensive to change some requirements when the development passes first phase.

This model is good for:

* embedded systems where the software has to interface with hardware systems
* critical systems where there is need for extensive safety and security analysis of the software specification and design
* large software systems that are part of broader engineering systems developed by several partner companies

This model is bad for:

* situations where informal team communication is possible and software requirements change quickly.

**Agile development**

The best examples for agile development are XP, the SCRUM and KANMBAN.