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

Both of these types of development are good for the specific types of projects. Plan-driven development mostly aims for perfection, planning everything ahead, lots of documentation, it is usually split in phases and every phase has to be done and “singed off” before the next one begins. In agile development the processes are planned incrementally starting from the most important ones and new ones added as the development continues. The priority is to do the current increment and to release working functionality of software quickly so it could be used in practice but still keep working on the rest of the software. Agile manifesto principles clearly describe the differences between plan-driven and agile development [agile manifesto]:

* Individuals and interactions over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan

This means that agile methodologies are focused on keeping customer close to project by involving him as much as possible instead of making a lot of agreements, planning and contract negotiations, trying to deliver working software quickly instead of making a lots of documentations and welcomes changes in requirements without a lot of modifications in projects overall instead of modifying all the previous work.

The progress of the plan-driven development is measured by the plan which was made from beginning, however in agile development progress is not that visible to be measured as the software grows by adding new functionality and the requirements change very often. To address this issue Schwaber, Beedle and Rubin proposes to use the Scrum agile method to provide a framework for organizing agile projects and, to some extent at least, provide external visibility of what is going on [Schwaber and Beedle 2001; Rubin 2013]. The Scrum follows principles from agile manifesto.The other way to address this issue is to use Kanban method [kanban]. Kanban also provides with a framework for organizing projects but works on slightly different principles:

* Start with existing process
* Agree to pursue incremental, evolutionary change
* Respect the current process, roles, responsibilities and titles
* Leadership at all levels

The Kanban method starts with existing roles and processes and stimulates continuous, incremental and evolutionary changes to the system however Scrum does not have a specific roles except scrum master, which may change, and product owner which is the customer or other stakeholder representative. In Scrum every member of the developer team is usually involved in everything. All the principles mostly aimed to the roles which are not that important for Scrum. However Kanban has fewer rules than Scrum, higher degree of freedom, needs more experienced developers [Kanban slides]. Both of these are also referred as agile development methods. They also might be combined with other methods.

Projects are not always perfect and requirements may change all the time as the time goes. When changes has to be made in plan-driven development it is expensive as the previous phases and documentation has to be modified. In agile development the changes are done all the time as the customer is closely involved in development by using informal communications rather than formal meetings with written documents.

Waterfall model is typical plan-driven development which is split in five phases [waterfall model]:

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

As mentioned before every next phase can begin after previous is done.

[Incremental approach](http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/) is typical agile development where the processes of specification, design and implementation are interleaved [Incremental development]. According to agile development techniques Extreme Programming (XP) is one of the most significant methods in agile development [XP]. In contrast to waterfall model XP practises different approaches of development:

* Planning Game
* Small releases
* Metaphor
* Simple design
* Define test first
* Refactoring
* Pair programming
* Collective ownership
* Continuous integration
* 37 hour week
* On-site customer
* Coding standards

It doesn’t have any phases and everything works incrementally and done in many iterations. The principles of XP are also based on the agile manifesto.

When it comes to maintenance the plan-driven development outperforms agile development. As a result of detailed documentation of the software the maintenance can be done much easier. That’s where agile development is weak as a documentation is really small. To address this issue you must keep customer involved in process.

The plan-driven development 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

Agile development is good for:

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

[agile manifesto] Software Engineering, 10th edition, Ian Sommerville, Page 76., http://agilemanifesto.org/

[Schwaber and Beedle 2001; Rubin 2013] Software Engineering, 10th edition, Ian Sommerville, Page 85.

[waterfall model] Software Engineering, 10th edition, Ian Sommerville, Page 47.

[Incremental development] Software Engineering, 10th edition, Ian Sommerville, Page 49.

[XP] Software Engineering, 10th edition, Ian Sommerville, Page 77; System development slides, eCampus, session 3

[kanban] <https://en.wikipedia.org/wiki/Kanban_(development)>, 10.16.2015

[Kanban slides] system development slides, eCampus, session 4