*Plan driven:*

Plan-driven software development is a more formal specific approach to creating an application. Plan-driven methodologies all incorporate: repeatability and predictability, a defined incremental process, extensive documentation, up-front system architecture, detailed plans, process monitoring, controlling and education, risk management, verification and validation. Customer is only part of the beginning of the process to provide the requirements.

Plans are typically generated by the following:

* Project broken down into stages/tasks
* Each task broken into its composite activities
* Each individual task estimated (perhaps using metrics)
* Estimates aggregated to give total effort
* Task list analysed for concurrencies and dependencies to provide schedule
* Schedule and delivery date help provide team size and team composition
* Team assembled and project commences
* Monitoring and control throughout development
* Delivery!

Agile:

Most agile development methods break product development work into small increments that minimize the amount of up-front planning and design. Iterations, or sprints, are short time frames (time boxes) that typically last from one to four weeks. Each iteration involves a [cross-functional team](https://en.wikipedia.org/wiki/Cross-functional_team) working in all functions: [planning](https://en.wikipedia.org/wiki/Project_planning), [analysis](https://en.wikipedia.org/wiki/Requirements_analysis), [design](https://en.wikipedia.org/wiki/Software_design), [coding](https://en.wikipedia.org/wiki/Computer_programming), [unit testing](https://en.wikipedia.org/wiki/Unit_testing), and [acceptance testing](https://en.wikipedia.org/wiki/Acceptance_testing). At the end of the iteration a working product is demonstrated to stakeholders. This minimizes overall risk and allows the product to adapt to changes quickly. An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal [bugs](https://en.wikipedia.org/wiki/Software_bug)) at the end of each iteration. Multiple iterations might be required to release a product or new features.

Working software is the primary measure of progress.

Differences:

Plan driven: Development is phase-based and sequential. Try to eliminate process and product [variability](http://www.innolution.com/resources/glossary/variability). Make each decision in its proper phase. Assumes we have all of the correct information up front to create the requirements and plans. Exploit what is currently known and predict what isn’t known. Change is disruptive to plans and expensive, so it should be avoided. The process is highly [predictive](http://www.innolution.com/resources/glossary/predictive-process). The process is tolerant of long-lived [assumptions](http://www.innolution.com/resources/glossary/assumption). Critical learning occurs on one major analyze-design-code-test loop. The process is tolerant of late learning. Batches are large, frequently 100% to all before any. Economies of scale should apply. [Inventory](http://www.innolution.com/resources/glossary/inventory) isn’t part of the belief system so is not a focus. Allocate people to achieve high levels of utilization. [Cost of delay](http://www.innolution.com/resources/glossary/cost-of-delay) is rarely considered. Conformance is considered a primary means of achieving a good result. Demonstrate progress by progressing through stages or phases. Process centric. Follow the process; do things right the first time and go fast. Quality comes at the end, after an extensive test and fix phase. Formality (well-defined procedures and checkpoints) is important to effective execution.

Agile: Development should be [iterative and incremental](http://www.innolution.com/resources/glossary/iterative-and-incremental-process). Leverage variability through [inspection](http://www.innolution.com/resources/glossary/inspection), [adaptation](http://www.innolution.com/resources/glossary/adaptation), and [transparency](http://www.innolution.com/resources/glossary/transparency). Keep options open. We can’t get it right up front. Favor an adaptive, [exploratory](http://www.innolution.com/resources/glossary/exploration) approach. Embrace change in an economically sensible way. Balance predictive up front work with adaptive just in time work. Validate important assumptions fast. Leverage multiple concurrent [learning loops](http://www.innolution.com/resources/glossary/learning-loop). Organize workflow for [fast feedback](http://www.innolution.com/resources/glossary/fast-feedback). Use smaller, economically sensible [batch sizes](http://www.innolution.com/resources/glossary/batch-size). Recognize inventory and manage it to achieve good [flow](http://www.innolution.com/resources/glossary/flow). Focus on [idle work](http://www.innolution.com/resources/glossary/idle-work), not [idle workers](http://www.innolution.com/resources/glossary/idle-workers). Always consider cost of delay. Adapt and replan rather than conform to a plan. Measure progress by validating working assets. Value centric deliver the value. Go fast but never hurry. Build quality in from the beginning. Employ minimally sufficient [ceremony](http://www.innolution.com/resources/glossary/ceremony).

Why Agile?

We wanted to work with scrum. Documentation is minimal. A strong team that is best suited for an agile process. It is adaptive and not restrictive. As you can see on the chart our project is best suited for an agile process.

Boehm’s 5 forces:

