Agile vs. Traditional Development Techniques

Agile methods are based on adaptive software development methods, while traditional SDLC (Software Development Life Cycle) models (waterfall model, for example) are based on a predictive approach. In traditional SDLC models, teams work with a detailed plan and have a full list of characteristics and tasks that must be completed in the next few months or the entire life cycle of the product. Predictive methods completely depend on the requirement analysis and careful planning at the beginning of the cycle. Any change that is to be included will go through a strict change control management and prioritization.

The agile model uses an adaptive approach where there is no detailed planning and only clear future tasks are those related to the characteristics that must be developed. The team adapts to dynamic changes in the product requirements. The product is frequently tested, minimizing the risk of major faults in the future. Interaction with the clients is the strong point of agile methodology and open communication and minimal documentation are typical characteristics of the agile development environment. Teams collaborate closely and often are located in the same geographical space.

While agile SDLC is better suited for small and medium projects, on large scale traditional SDLC is still the better choice. Therefore it is important that the development team selects a SDLC that is best suited for project at hand. There are criteria that can be used by the development team to identify the dimension of the desired SDLC. They include team size, geographical location, size and complexity of the software, project type, business strategy, engineering capabilities etc. Also, it is very important for the team to study the differences, advantages and drawback of each SDLC before making a decision.

Furthermore, the team must study the context of the business, industry requirements and business strategy before evaluating the candidate SDLCs. It is important to have a SDLC evaluation and selection process because it maximizes the chances to create successful software. Therefore, selection and adoption of an appropriate SDLC is a management decision with long term implications.

Although agile methodologies triumph over traditional ones in several aspects, there are many difficulties in making them work. One of them is the significant reduction of documentation and the claim that the source code itself should be the documentation. Thus, developers used to agile methods tend to insert more comments in source code in order to clarify and explain. It is difficult for beginner developers or new members of the team to complete their tasks when they cannot fully understand the project. They ask lots of questions to the experienced developers and this may delay completion of the iteration, which can lead to increased development costs.

On the other hand, traditional methods emphasize documentation in orientation and clarification of the project for the development team, so there is no concern about not knowing the project details or not having a knowledgeable developer. Agile methodologies are well known for the importance given to communication and client implication.

For each version delivered, the development team and the clients will organize a meeting where the team will present the work done in current iteration and the clients will provide feed-back on the delivered software (improvements on current features or addition of new ones).

Most times, developers will find the periodic meeting (usually weekly) boring and tiring because they have to present the modules repeatedly, to new members and clients and, on each iteration, changes may happen as requested by clients.

The time frame for each iteration is short (usually weeks). Developers will find the schedule too tight for each module, even more so for modules that require complex processing algorithms. This leads to delays in each iteration and hardships in establishing an efficient communication between team members and the clients.

On the other hand, traditional methodologies have a well-defined requirements model before the implementation and coding process starts and this acts as a reference for the development team during the coding process. Clients do not participate in this stage of the development life cycle. The development team will perform the coding according to the documentation provided by the business analysts until the system is complete and only then it will be presented to the clients as final product. Developers are not concerned about frequent meetings and have more time to finish the system. This allows them to provide a better product.

The fact that agile development allows changes in requirements in an incremental way lead to two dependency problems in design: rigidity and mobility. Rigidity means a change in the system leads to a cascade of changes in other modules, while mobility means the inability of the system to include reusable components because they involve too much effort or risk. When such problems are present throughout the system, there must be a high level restructuring in order to eliminate unwanted dependencies.

	Traditional development	**Agile development**
Fundamental principles	Systems are fully specifiable, predictable and are developed through extended and detailed planning	High quality adaptive software is developed by small teams that use the principle of continuous improvement of design and testing based on fast feed-back and change
Management style	Command and control	Leadership and collaboration
Quality control	Difficult planning and strict control. Difficult and late testing	Permanent control or requirements, design and solutions. Permanent testing
User requirements	Detailed and defined before coding/implementation	Interactive input
Requirements	Stable	Rapid change
Cost of restart	High	Low
Remodeling	Expensive	Not expensive
Development direction	Fixed	Easily changeable
Client involvement	Low	High
Testing	After coding is completed	Every time