Chapter-5

LIMITATIONS OF AGILE METHODS

An agile approach for software development is good for so many reasons. But agile development also requires certain things that can be disadvantageous.

Having fully detailed documented requirements that will not change is unrealistic in the software development process because changes will always occur. During the software development process, if errors occur in the requirements engineering stage and the developers continue with the project, then the customer will not be satisfied with the final product. Agile is a more recent software development methodology introduced to help and address some of these system development challenges. However, as in all agile methodologies, there is a resistance to the development of traditional documents. Many developers perceived problems with agile development processes. Difficulties [118] faced during implementation of agile methods are:

- **Developer’s fear of skill-deficiency exposure**

  In many reviews it has been found that developers feared that the agile process could bring their deficiencies to light. So, always there is a pressure on the person. To address this challenge, developers need an environment where they feel safe to expose their weaknesses, like they could document any fears, issues or
concerns due to which they didn’t feel comfortable in an open forum, or there may be separate meetings for junior or new staff.

- **Broader skill sets for developers**

  Generally in all the companies, agile environments seem to blur the boundaries among developer’s roles and require competence in a broad range of skills, as opposed to specialization in one. To address this challenge, organizations must strike a balance between team members becoming “masters of all” or “masters of none.” It is to be advisable that developers must have broad knowledge on all aspects of software development but should also specialize and hone their skills in certain areas.

- **Increased social interaction**

  Agile practices such as pair programming, collocation, meetings, and retrospectives increase social interaction, communication, and our presentation skills. In most of the cases there are people who were technically very talented but had inherently weak communication and presentation skills, while all managers saw the benefits of constant face-to-face communication, the degree of communication in agile environment. Social-skills training are an obvious solution to this challenge. It can help new people to get into the company values easily.

- **Understanding agile principles**

  Some projects can or some can’t implement agile values and principles. Some implemented agile methods “on paper,” but they didn’t achieve agility’s
ultimate goals. It may be due to many reasons like intangible combination of staff 
personality, management style and other factors.

In fact, it is not recommended to adopt agile development solely to increase 
productivity. Its benefits like ability to release software more frequently which has 
come from working differently, not from working faster. Our team will need time to 
learn agile development. When they learn, it will take a quarter or two; they’ll go 
slower, not faster. In addition, emphasizing productivity might encourage our team 
to take shortcuts and to be less rigorous in their work, which could actually harm 
productivity. So limitations [119] [120] [121] [122] are:

- Architecturally very risky for new projects when potential solutions are not 
  very well understood.
- No time for careful design during iterations.
- No considerations for alternative, so potentially better design choices can be missed.
- No focus on quality attributes except some implicit focus on performance issues.
- Need high-quality collaboration between customers and agile development team.
- Need a high-level of customer involvement.
- Lack of long-term detailed plans.
- Producing a lower level documentation.
• Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for new programmers, unless combined with experienced resources.

• Active user involvement and close collaboration are required throughout the development cycle. This is very engaging, rewarding and ensures delivery of the right product. It’s the fundamental principle in agile that ensures expectations are well managed. And since the definition of failure is not meeting expectations, these are critical success factors for any project. However these principles are very demanding on the user representative’s time and require a big commitment for the duration of the project.

• Requirements emerge and evolve throughout the development. This creates the very meaning of agile flexibility. Flexibility to change course as needed and to ensure delivery of the right product. There are two drawbacks to this principle though. One is the potential for scope creep, which we all know can create the risk of ever-lasting projects. The other is that there is much less predictability, at the start of the project and during, about what the project is actually going to deliver. This can make it harder to define a business case for the project, and harder to negotiate fixed price projects. Without the maturity of a strong and clear vision, and the discipline of fixing timescales and trading scope, this is potentially very dangerous.


- Agile requirements are barely sufficient. This eliminates wasted effort on deliverables that don’t last, which saves time and therefore money. Requirements are clarified just in time for development and can be documented in much less detail due to the timeliness of conversations. However this can mean less information available to new starters in the team about features and how they should work. It can also create potential misunderstandings if the teamwork and communication aren’t at their best and difficulties for team members (especially testers) that are used to, everything being defined up front.

- Testing is integrated throughout the lifecycle. This helps to ensure quality throughout the project without the need for a lengthy and unpredictable test phase at the end of the project. However it does imply that testers are needed throughout the project and this effectively increases the cost of resources on the project. This does have the effect of reducing some very significant risks that have proven through research to cause many projects to fail. The cost of a long and unpredictable test phase can cause huge unexpected costs when a project over-runs. However there is an additional cost to the project to adopt continuous testing throughout.

- Frequent delivery of product and the need to sign off each feature as done before moving on to the next makes UAT (user acceptance testing) continuous. The users or product owner needs to be ready and available for
prompt testing of the features as they are delivered and throughout the entire duration of the project. This can be quite time-consuming but helps drastically to ensure a quality product that meets user expectations.

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development lifecycle.

- The project can easily get taken off track if the customer representative is not clear about what final outcome they want.

- For small sized projects agile can be adopted but for large size projects traditional methods should be adopted as they have the clear outline of what has to be developed.