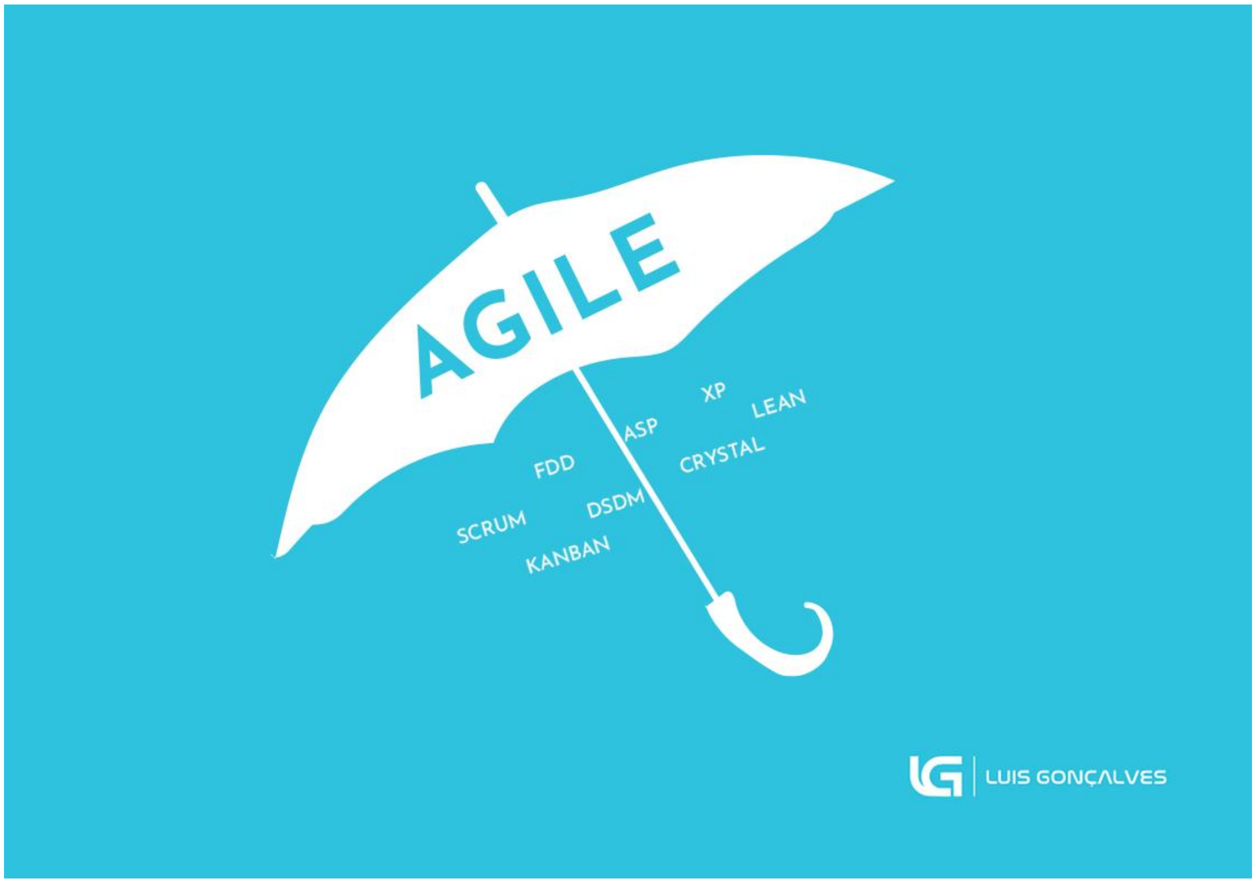


What is Agile Methodology?

I am often asked the question: **What is agile methodology?** Quite simply, agile is a hype word that the IT industry uses to describe an alternative method of project management.

Agile is a process that helps teams provide quick and unpredictable responses to the feedback they receive on their project. It creates opportunities to assess a project's direction during the development cycle. Teams assess the project in regular meetings called sprints or iterations.

An agile is a very **empowering process** that helps companies design and build the right product. The management process is very beneficial for software companies because it helps them analyze and improve their product throughout its development. This enables companies to produce a highly valuable product so they stay competitive in the market.

Where did Agile come from?

In 2001, a small group of people, tired of the traditional approach to managing software development projects, designed the agile manifesto. It is a more improved method for managing the progress of software projects.

The agile manifesto has four important values:

- Focus should be more on individuals and interactions instead of processes and tools
- Working software is more important that comprehensive documentation
- Customer collaboration is more vital than contract negotiation
- The process should respond to change rather than follow a plan

There are **12 principles of agile software development**:

- Deliver customer satisfaction by delivering valuable software continuously
- Always accept change of requirements matter how early or late in the project
- Deliver software that works within a shorter timescale
- Both developers and business professionals must work closely together daily throughout the duration of the project

- Information is best transferred between parties in face-to-face conversations
- Motivate people to build a project by creating an environment of appreciation, trust, and empowerment
- Working software is the key measure of progress
- The agile process promotes sustainable development
- Continuous attention to excellence and quality in technical development and design boosts the agility
- 10 Simplicity is a vital part of effective agile management
- Self-organized teams produce the best architecture, requirements, and design
- Teams should reflect through inspection and adaption to be more effective

There are different methods of agile that promote the values and principles of the manifesto. Scrum and XP are two well-known examples.

The Benefits of Agile Software Development

Top software developers developed agile meetings. After repeatedly experiencing challenges and limitations from traditional waterfall development in real life projects, they wanted to create a more efficient process for analyzing project development. The approach they used addresses the issues regarding the philosophies and processes of traditional methods directly.

There are many **benefits to the agile software development.** They include:

Stakeholder Engagement and Satisfaction

The agile process creates many opportunities throughout each sprint meeting for genuine engagement between the team and the stakeholders. Because the client is actively involved in the entire project, there is a continuous level of collaboration between all parties. This gives the team a chance to fully understand the client’s vision. By delivering high quality, working software frequently, the stakeholders quickly develop a trusting and authentic relationship with the team. This also further promotes engagement between the client and the team.

Transparency

The agile approach actively involves the client throughout the entire project including the iteration planning, review sessions, and new feature builds in the software. Clients, however, must understand that during the transparency of the project, they are seeing a work in progress and not the final product.

Early and Predictable Delivery

Sprints are held on a fixed schedule of 1 to 4 weeks duration. By using this time-boxed method, predictability is high as new features can be delivered to the stakeholders quickly and frequently. It also allows the team to beta test or release the software sooner if it has sufficient business value.

Predictable Costs and Schedule

Because the Sprints are on a fixed schedule, the costs are limited and predictable, and based on the amount of work done. By combining the estimated costs before each Sprint, the client will better understand the approximate costs of each feature. This offers more improved decision-making opportunities when prioritizing the features or adding iterations.

Flexible Prioritization

Scrum methodologies allow more flexibility by prioritizing the customer-driven features. The team has more control in managing the shippable units of work with each sprint boundary; making continued progress towards the final product milestone. To get a prompt RIO from the engineering, the work needs to be shipped early to the customers so they will realize the value from the features.

Allows for Change

While the focus should be to deliver the agreed subset of the products features, Agile processes create an opportunity to continually reprioritize and refine the product backlog. These changes can be added to the next iteration so the new changes can be introduced within a few weeks.

Focuses on Business Value

The team has a better understanding of what is most important for the client's business and can deliver features that give the most value to the business.

Focuses on Users

The user's stories are commonly used to define the product features as they relate to business-focused acceptance criteria. By focusing on the user's needs, each feature delivers real value and not just an IT component. It provides a better opportunity to gain valuable feedback through beta testing the software after each Sprint. This provides vital feedback earlier in the project so that changes can be made as needed.

Improves Quality

The projects are broken down into manageable units, making it easier for the team or focus on high-quality development, testing, and collaboration. By creating builds and conducts tests or reviews throughout the iteration, defects and mismatches can be found and fixed early, improving over-all quality.

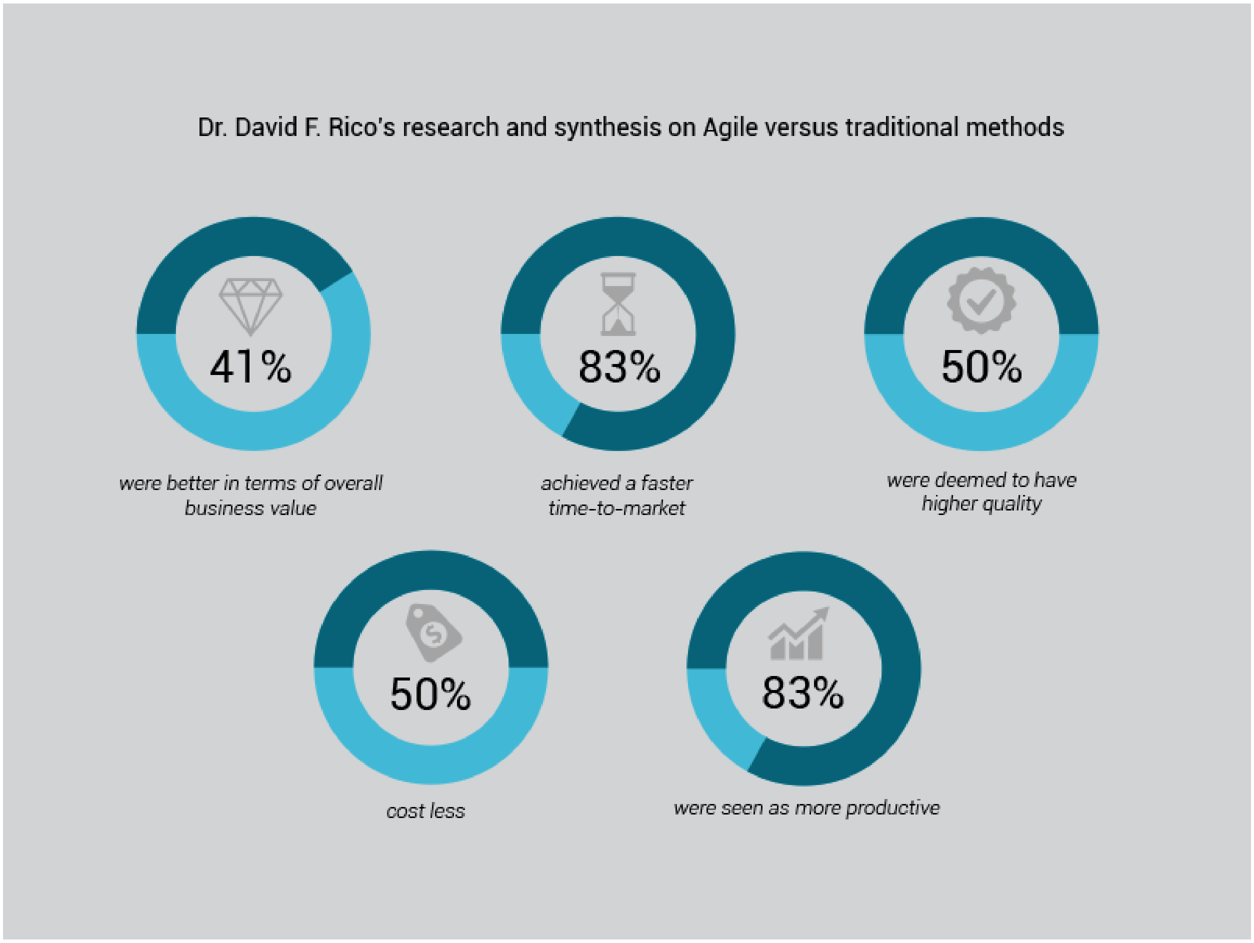
It gives your team Purpose

Most agile processes focus on creating a shared sense of ownership and goals for all team members. This to gives your team purpose rather than creating a false sense of urgency. Purposeful teams are more productive and challenge themselves to be faster and more efficient.

The Business Benefits of Agile

Agile management reduces the common risks that are associated with the delivery, scope, and budget of the project.

It encourages **collaboration** between the customer and the team; offering mutual benefits in the **mitigation of the high risks** during the development of the software.



In 2009, Dr. David F Rico compared Agile with traditional methods of software project management. During his research and synthesis, he analyzed 23 Agile processes, comparing them with 7,500 traditional projects.

He found **20 benefits** to Agile projects:

- *41% were better in terms of overall business value*
- *83% showed quicker time-to-market speeds*
- *50% were higher in quality*
- *50% were less costly*
- *83% were more productive*

Agile Methodologies

There are several agile methodologies; all share similar philosophies, characteristics, and practices. However, from the point of implementation each agile has its own practices, terminology, and tactics. Some of the main agile software development methodology components include:

Scrum

Scrum is a management framework with far reaching abilities to control and manage the iterations and increments in all project types. They are lightweight and can be used with other agile methodologies for various engineering practices. Scrums have grown in popularity within the agile software development community because they are simple and have a proven productivity rate.

Lean and Kanban

1.Lean Software Development

Lean software development is an iteration methodology originally developed by Mary and Tom Poppendieck. Many of the principles and practices in Lean Software Development came from the lean enterprise movement and were first used by big companies like Toyota. This value based method focuses on giving the customer an efficient "Value Stream" mechanism that delivers the value to the project.

The main principles of this methodology are:

- *Eliminate waste*
- *Amplify learning*
- *Make decisions as late as possible*

- *Deliver results as quickly as possible*
- *Empower the team*
- *Build integrity*
- *Envision the whole project*

By choosing only the features that have real value for the system, prioritizing and delivering them in small batches eliminates waste. Instead, the lean methodology emphasizes on speed and efficiency; relying on rapid, reliable feedback between the customers and programmers. It focuses on the idea that customer requests "pull" the product. The focus is more about the faster and more efficient decision-making abilities of individuals or small teams instead of a hierarchy controlled method. This methodology concentrates on the efficiencies of its team's resources, ensuring everyone is as productive as possible always.

2.The Kanban Method

Organizations use the Kanban method to manage the project's creation while emphasizing on the continued delivery and not overburdening the development team. Like Scrum, Kanban processes are designed to help teams work more efficiently together.

There are **three principles**:

- *Visualize what you do: see all the items within context of each other – more informative*
- *Limit the amount of work in progress (WIP): balance the flow-based approach so teams are not committed to doing too much work at once*
- *Enhance the flow: as soon as one task is finished, start on the next highest job from the backlog*

The Kanban method promotes continued collaboration by the client and team. It encourages ongoing learning and improvements to provide the best possible workflow for the team.

Extreme Programming (XP)

Extreme Programming (XP) was originally described by Kent Beck. It is one of the most popular and controversial agile methodologies. XP is a highly disciplined method of continually delivering high-quality software faster. The customer is actively involved with the close-knit team to perform continued planning, testing and rapid feedback to deliver working software frequently. The software should be delivered in intervals everyone to three weeks.

The original XP method is based on **four simple values:**

- *Simplicity*
- *Communication*
- *Feedback*
- *Courage*

It has **12 supporting practices**:

- *Planning game*
- *Small releases*
- *Customer acceptance tests*
- *Simple design*
- *Pair programming*
- *Test-driven development*
- *Refactoring*
- *Continuous integration*
- *Collective code ownership*
- *Coding standards*

- *Metaphor*
- *Sustainable pace*

Crystal

Crystal methodology is one of the most lightweight and adaptable approaches in developing software. It is made up of several agile processes including Clear, Crystal Yellow, Crystal Orange, and other uniquely characterized methods. There are several factors that drive these processes including: size of the team, criticality of the system, and the priorities of the project.

The Crystal family focuses on the realization that each project has unique characteristics, therefore, the policies and practices must be custom tailored to accommodate these features.

The Crystal method has **several essential tenets** including:

- *Teamwork*
- *Communication*
- *Simplicity*
- *Reflection*
- *Frequent adjustments*
- *Improve processes*

This agile process, like other methodologies, promotes early and frequent working software delivery. It encourages high user involvement, adaptability, and eliminations of distractions and bureaucracy.

Dynamic Systems Development Method (DSDM)

The Dynamic Systems Development Method (DSDM) originated in 1994 to provide an industry standard framework for project delivery for what was then known as Rapid Application Development (RAD). Although it was very popular in the 1990's, the RAD approach developed in an unstructured fashion.

Since its beginning, the DSDM has evolved and matured; it provides a comprehensive foundation in planning, management, execution, and scaling of the agile process and iteration projects.

DSDM has **six key principles** revolving around the business needs:

- *Value*
- *Active user involvement*
- *Empowered teams*
- *Frequent delivery*
- *Integrated testing*
- *Stakeholder collaboration*

DSDM uses a "fitness for business purpose" approach for delivery and acceptance criteria. It focuses on the formula: 80% system deployment in 20% time.

Feature-Driven Development (FDD)

Jeff De Luca, along with contributors A.m. Rajashima, Lim Bak Wee, Paul Szego, Jon Kern, and Stehen Palmer developed Feature-Driven Development (FDD). It is a model-driven, short iteration process that begins by first establishing the shape of the agile model. Iterations on "design by feature, build by feature" are held biweekly. The features appeal to clients because they are small and useful.

The FDD design and development is delivered using these **eight practices**:

- *Domain object modeling*

- *Development of features*
- *Component and class ownership*
- *Feature teams*
- *Inspections*
- *Configuration management*
- *Regular builds*
- *Visibility of progress and results*

Roles in Agile

In the agile development process, the Scrum most clearly defines what agile is in project management.

There are **three roles in the Scrum project**: **Product Owner, Scrum Master, and team members.**

The Product Owner oversees all the business conditions of the project to ensure the right product is built and in the right order. A good product owner balances the competing priorities, is available for the team, and makes decisions about the project.

The Scrum Master is the team's coach; they help the team work together effectively. Scrum Masters service the team by removing barriers that impair the progress, facilitating meetings and discussion groups, tracking progress, problem solving, and performing other project management duties.

The team works together to determine the best approach to achieve the product goals that are outlined by the product owner. The team decides which members will manage specific tasks, and outline the technical practices required to achieve the desired goals.

Where does Scrum Master role fit in Agile Software Development?

In agile project management practices, the Scrum Master is the 21st century's version of the project manager. However, unlike traditional project managers, the Scrum Master does not take the credit or blame for the success or failure of the project.

Their authority only extends to the process. The Scrum Master's expertise in the process motivates and guides the team to perform at its highest level. Other traditional management roles such as project scope, cost, personnel, and risk management are still the project manager's responsibility.

Surprise: Agile Organizations Are Hierarchical!

A common misconception about Agile organizations is that they are **non-hierarchical** or **flat**. Nothing could be further from the truth. Top managers still set the direction and tone for the rest of the organization and people are still fired for not doing their job. The push for a higher performance is even more relentless than in traditional, bureaucratic organizations. In a bureaucracy, poorly performing staff can still hide within the nooks and crannies of the system. But in an agile organization, there is a **radical transparency** that holds all peers accountable for their actions.

The hierarchy in an agile organization is very different than in a bureaucratic agency. The **agile hierarchy** is based on competence, not authority. The performance is not based on pleasing the boss but instead, adding value to the customer. The organization uses a dynamic horizontal and vertical communication approach that is very interactive. Ideas can come from any person in any position, including the customer. It is a network that is continually growing, learning, and adapting to the constant flux; it adds new value to the customers by exploiting the opportunities presented. If done right, the continued delivery of more value to the customers through less work results in more generous returns to the organization.

The agile clearly distinguishes the differences between exploitation and exploration. In an agile organization, all members are constantly exploring ways to add more value to the customer.

During the early years of Agile Management, critics believed the small teams could never handle large, complex issues. But they were wrong. All teams are part of an interwoven network that is driven by ongoing conversations between all parties that focus on common goals. The common and consistent rhythms of all parties, provides a solid platform that enables the smaller teams to manage more complex problems. This is by far, a much better system than the bureaucratic method.

Agile Process

The agile process breaks a larger software project into several smaller parts that can be developed in increments and iterations. Studies have proven that there is a negative correlation between project size and success (i.e.: the shorter the project, the higher the success rate).

The agile approach reduces the size of the project by creating several smaller projects. This iteration approach distinguishes Agile management from other management methods.

Unlike other methods, Agile management uses iterations during the planning and development phases. **Each iteration is usually a week**

long. During these sessions, the project team and customer team collaborate to prioritize what needs to be added to the iteration. **The final result** is a working **software program delivered quickly to the customer** in a production-like environment. Customers can then test their program and make changes if needed. Many releases are made throughout the process as changes to the program are made. This iteration process is repeated until the project is completed.

General management and Agile

Software programming is a critical component for almost every business today. This means that **Agile is an essential process for every type of organization** and form of work.

Agile methodologies that incorporate new values, practices, principles, and benefits are a **better alternative to the traditional command-and-control style of project management.** Agile process is even spreading throughout different industries and functions, including C-suites.

However, while many companies are adopting some Agile processes, they are still operating with a bureaucratic top-down method. In today's digital dominant economy, it is vital for companies to develop agile management practices. But many companies still **struggle with this transition** and operate in a command style culture. This is reflected in the mindset and skills of the senior management and is the greatest obstacle companies face today.

Agile Practices

There are **many different agile practices**; many are not used by agile practitioners. Those who want to convert to using the agile process, should understand the different practices to help under how they can apply the practice to their environment. The following examples help illustrate how Agile practices can be applied.

Daily Standup (Stand-Up Meetings)

Daily stand-up meetings are also called Daily Scrum meetings. The Scrum sessions are held daily by the team so they can share pertinent information with each other. These meetings are designed to keep all team members equally informed and updated on the status of the project. The key to each meeting is brevity.

During the daily scrum meetings, **each member should answer these three questions**:

- *What did I do yesterday?*
- *What will I do today?*
- *What problems are hindering my progress?*

User Stories

A user story is a **brief description of the function wanted by the end user**. There are **three elements to the user story**. They are:

- *A written description, planning the story (usually written on a card)*
- *A conversation about the story to gain better understanding*
- *A series of tests confirming the story.*

The stories are written from the end user's perspective and uses language that they understand. Stories act as currency between developers and customers; both parties clearly understand them. You can read more about 4 reasons why User Stories are not getting "done".

Automated Testing

Implementing formal and thorough automated testing is a vital part of the agile process. The tests find and eliminate defects at their source to ensure that a working software package is delivered to the customer. Developers can create the test code under a safety net using a variety of available frameworks while simultaneously developing the software code. **This method protects other features while making changes to the software.** It is also a **faster, more efficient way to find bugs** in the program.

Automated Builds

A key principle for Agile methodologies is to have running softwareat all times. In practice, the only way to do this is by ensuring that all software development is regularly and automatically compiled, built, deployed and tested. This is usually done many times a day and at least once every time a developer “checks in” code as a main part of the development branch.

Agile Planning: Release, Iteration and Task

There are **three levels of Agile development planning: release, iteration, and task.** In the beginning stages, project developers and customers meet to discuss the primary user stories that are needed for the software. The meeting’s initial focus is on the must-have features to estimate and prioritize what needs to be done.

Release planning is a customer-driven planning session. Both customers and developers choose a date for the first product release series. They collaboratively decide what stories to incorporate during each release. The developers focus on the estimate efforts of the story while the customers focus on the story selections. There are a variety of forms of estimate efforts that are determined by the needs and wants of the customer and development teams.

Iteration planning requires collaborative efforts between customers and developers to initiate part of the release plan. During iterations, the customer defines and prioritizes the user stories while the developers estimate how much effort is required to develop each user story. The timeline is much shorter for iterations, taking only weeks instead of months.

Task planning occurs after the iteration planning. Stories are broken down into a series of doable steps by the development team. Task lists are developed and posted in the project room so they are clearly visible to all party members. Common tools used during this planning session include post-it notes and whiteboards. Each developer volunteers to perform a task and assign an estimate to it.

Pair Programming

In pair programming, two developers work as a team on one programming task. One person is the Driver, the person who enters the code, while the second person is the navigator, the one who plans the next steps while fitting the code into the whole picture.

One common complaint to pair programming is the **waste of human resources to do the task**. It should not take two people to do a job that can be done by one person. However, while the programming uses more man power, the final output justifies the expense. A recent study has found that **pair programming uses 15% more effort but produces 15% less defects.** While results may vary from case-to-case, developers often find that the reduction in errors is worth the extra resources used.

Another **benefit is that pairing is not required full time**. Teams can establish their own rules and schedules when deciding if it’s better to pair.

Continuous Integration

During continuous integration, development teams input their code into the system several times throughout the day. A series of tests are run before the code is added to make sure it won't damage other pre-existing tests or functions in the system. The developer must run all the test for the system first and fix any problems before integrating the other codes. The more often a code is integrated into the software, the quicker and easier it is to merge and spot errors.

Retrospectives

Retrospectives are meeting held at or near the end of a project. They give all parties involved a chance to look back and r**eflect on the work done during the process.** The entire team looks at what went well, what did not, where **improvements** can be made and, most importantly, how they can take the lessons that they have learned and turn them into actionable change.

Conclusion

Agile management is an exciting and fascinating approach to software development. By integrating the product developers and customers in the planning and implementing processes, the result is a more rewarding experience for everyone involved.

When Agile programming is don properly, organizations can continually find ways to increase the value to their customers. It gives more meaning to those who are actively working on the project and creates a more positive experience for the customer, producing more generous end results for the company.