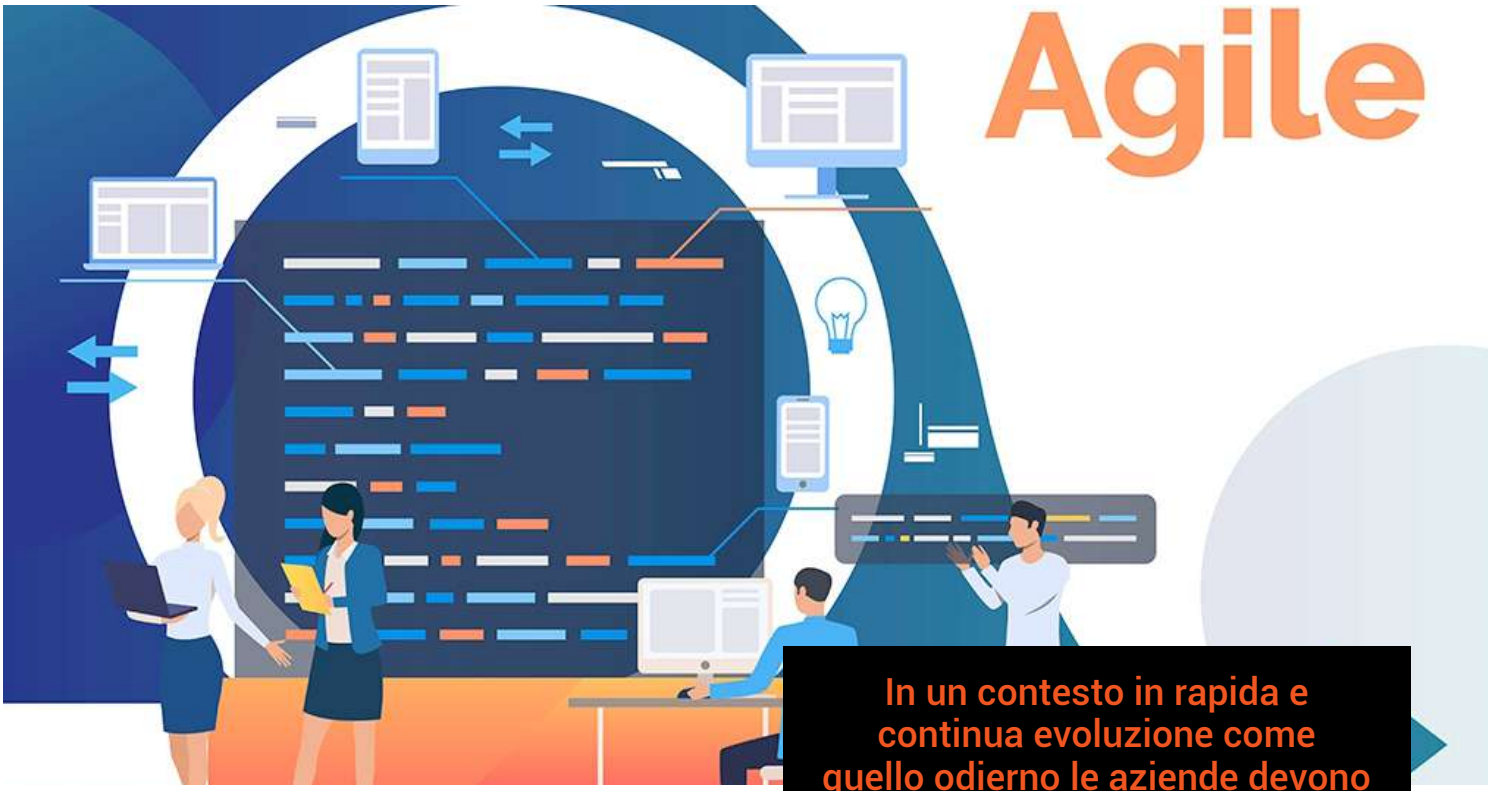
L'APPROCCIO AGILE NEL SOFTWARE DEVELOPMENT



A cura di Massimo Nannini (*)

Sommario

In un contesto in rapida e continua evoluzione come quello odierno le aziende devono affrontare lo sviluppo dei nuovi progetti in condizioni sempre più incerte, volatili e complesse cercando di mantenere la competitività e cogliere i cambiamenti di mercato. Soprattutto negli ambiti di ricerca e sviluppo nuovo prodotto, l'incertezza condiziona fortemente i metodi di lavoro dei teams che si trovano incapaci di prevedere i risultati e gli eventi futuri a causa della differenza tra la quantità di informazioni richieste e quelle effettivamente disponibili.

I classici approcci di sviluppo plan-driven, concepiti secondo un criterio di linearità, risultano eccessivamente vincolati e poco reattivi, in quanto non consentono di accogliere rapidamente le modifiche dello sviluppo software in corso d'opera. Per contrastare questa mancanza di reattività dei metodi più tradizionali, molte aziende hanno iniziato a ripensare al modo in cui i processi di sviluppo sono gestiti e a comprendere il vero valore modello Agile.

Origini ed evoluzione del modello Agile

L'approccio Agile nacque nel febbraio del 2001, quando 17 sviluppatori software professionisti si riunirono a Snowbird, nello Utah, per cercare una valida alternativa alle tecniche di sviluppo tradizionali. Tutti i partecipanti ritennero che il modello Waterfall non rappresentasse la soluzione più efficace, soprattutto in un'ambiente in cui è importante rispondere ai continui cambiamenti e quindi possedere una certa "agilità" per affrontare uno sviluppo dinamico.
L'obiettivo era quello di definire una nuova metodologia in grado di aumentare la qualità, migliorare la flessibilità e accelerare il time to market limitando ritardi e sprechi e soprattutto l'insoddisfazione dei clienti dovuta al rilascio di prodotti non conformi alle specifiche richieste.
Il risultato fu l'elaborazione di un "Manifesto Agile", un documento che delineava i principi fondamentali su cui si basa l'approccio Agile. Questi principi pongono l'accento sulla risposta al cambiamento, sulla consegna frequente di software funzionante, sulla collaborazione tra individui e sulla capacità di accogliere i requisiti anche a uno stadio avanzato dello sviluppo.
L'Agile, un framework di sviluppo che prevede di energizzare, responsabilizzare e consentire ai teams di progetto di generare valore in modo rapido e affidabile, coinvolgendo i clienti, apprendendo e adattandosi continuamente alle mutevoli esigenze dell'ambiente circostante. Mediante questo approccio non si pianifica né si progetta un prodotto in anticipo, ma il processo di sviluppo evolve per cicli iterativi. Queste brevi sessioni, chiamate sprint, solitamente hanno una durata di 2 o 3 settimane, in cui ogni membro del team deve completare una serie di compiti assegnati (comunemente chiamati task). Al termine di ogni sprint si analizzano i risultati ottenuti per misurare lo stato di avanzamento dei lavori e la velocità con cui il team di sviluppo brucia le tappe. A questo punto il processo ricomincia da capo. I processi di gestione dei progetti che incorporano questi principi consentono al cliente di ricevere man mano aggiornamenti e segnalare priorità o introdurre cambiamenti quando necessario.
Dunque, l'Agile consente di rispondere al cambiamento in un business turbolento facendo ricorso a:

- una forte interazione con i clienti e la loro integrazione durante le fasi di sviluppo;
- una riprioritizzazione rapida delle risorse;
- una consegna evolutiva, incrementale e iterativa dei prodotti;
- una metodologia just in time, che generi valore massimizzando il lavoro non svolto.

Rispetto alle metodologie tradizionali che puntano alla massimizzazione del profitto (shareholders based), l'Agile ha l'obiettivo di massimizzare la soddisfazione del cliente, cioè il valore generato (stakeholders based). Inoltre, il processo passa da essere prescrittivo (rule-based) ad adattativo (principle-based).

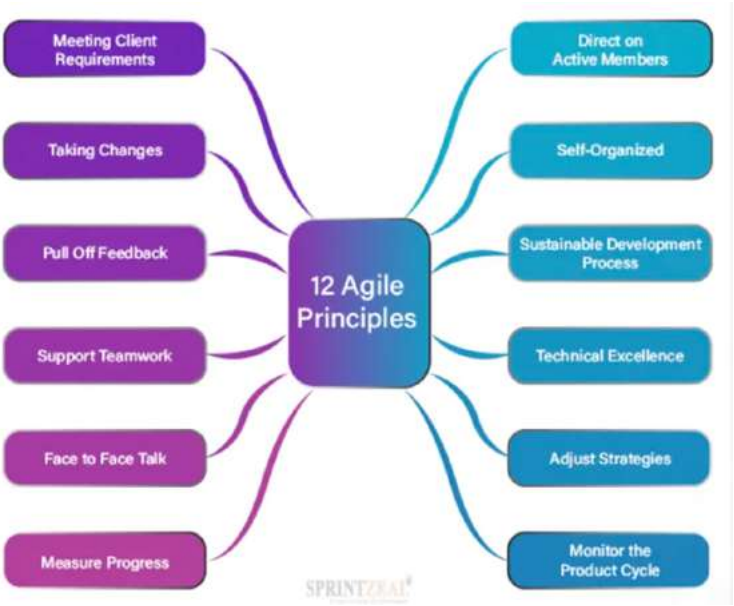
Il Manifesto Agile

Per comprendere appieno le potenzialità dell'Agile, è essenziale esaminare i suoi principi e pratiche fondamentali che sono stati appunto riportati all'interno del "Manifesto Agile" suddivisi in 12 principi e 4 valori che esprimono le idee e le riflessioni dei fondatori:

I principi

1. La nostra massima priorità è soddisfare il cliente rilasciando software di valore, fin da subito e in maniera continua.
2. Accogliamo i cambiamenti nei requisiti, anche a stadi avanzati dello sviluppo. I processi agili sfruttano il cambiamento a favore del vantaggio competitivo del cliente.
3. Consegniamo frequentemente software funzionante, con cadenza variabile da un paio di settimane a un paio di mesi, preferendo i periodi brevi
4. Committenti e sviluppatori devono lavorare insieme quotidianamente per tutta la durata del progetto.
5. Fondiamo i progetti su individui motivati. Diamo loro l'ambiente e il supporto di cui hanno bisogno e confidiamo nella loro capacità di portare il lavoro a termine.
6. Una conversazione faccia a faccia è il modo più efficiente e più efficace per comunicare

con il team e all'interno del team.
7. Il software funzionante è il principale metro di misura di progresso.
8. I processi agili promuovono uno sviluppo sostenibile. Gli sponsor, gli sviluppatori e gli utenti dovrebbero essere in grado di mantenere un ritmo costante.
9. La continua attenzione all'eccellenza tecnica e alla buona progettazione esalta l'agilità.
10. La semplicità – l'arte di massimizzare la quantità di lavoro non svolto - è essenziale.
11. Le architetture, i requisiti e la progettazione migliori emergono da team che si auto-organizzano.
12. A intervalli regolari il team riflette su come diventare più efficace, dopodiché regola e adatta il proprio comportamento di conseguenza.



I principi del Manifesto Agile

I valori

I. Gli individui e le interazioni più che i processi e gli strumenti.
Lo scopo di questo valore è proprio quello di sottolineare l'importanza di avere un bilanciamento tra gli individui e i processi. Oggi molti progetti sono caratterizzati da un'elevata incertezza, con una forte spinta all'innovazione e alla creatività. In contesti come questo, seguire un processo predefinito e standardizzato non è mai la soluzione ottimale per portare a termine con successo un progetto. Di fronte all'incertezza e all'esplorazione di nuove soluzioni, i membri del team dovrebbero avere la libertà di sperimentare e di essere sempre aperti all'apprendimento; ognuno dovrebbe avere la possibilità di creare i suoi processi, personalizzandoli in base alle proprie esigenze e agli obiettivi da perseguire. Tuttavia, i team di sviluppo sono soliti ricevere processi e strumenti predefiniti da parte dell'azienda, rimanendo così bloccati dalle abitudini all'interno di una cultura predittiva e questo chiude l'apertura verso l'innovazione, la creatività e l'esplorazione di nuove soluzioni.

II. Il software funzionante più che la documentazione esaustiva.
Con "software funzionante" si intende un software completo, testato, integrato, che risponda ai requisiti di qualità richiesti e che generi valore per il cliente. La qualità è soggettiva e può cambiare nel corso degli anni, per questo motivo occorre sempre specificare le richieste in maniera chiara e misurabile, indicando qual è il requisito qualitativo da rispettare. Per quanto riguarda la documentazione esaustiva invece, si intende tutto quel lavoro che non è consegnato al cliente perché privo di valore.

III. La collaborazione con il cliente più che la negoziazione dei contratti.
Questo valore si basa sulla necessità di mantenere sempre una relazione di collaborazione con i principali stakeholder del progetto, fortemente fondata sul concetto di trasparenza e fiducia. In tal senso, è necessario formulare dei contratti adeguati alla situazione, senza puntare a una continua negoziazione, bensì a una collaborazione costante.

IV. Rispondere al cambiamento più che seguire un piano.
La metodologia Agile non prevede l'utilizzo di un Gantt, ma ciò non significa che la pianificazione sia completamente assente. Piuttosto, si procede passo per passo con una pianificazione di tipo adattativo, che accetta cambiamenti al progetto ed è estremamente flessibile. Questo tipo di pianificazione non è supportata da un percorso predefinito così da orientare il team ad abbracciare il cambiamento.

Le pratiche

Oltre a queste chiare e specifiche indicazioni, all'interno del Manifesto, troviamo anche una serie di "pratiche" basate su cicli iterativi e di sviluppo incrementale, in cui i requisiti

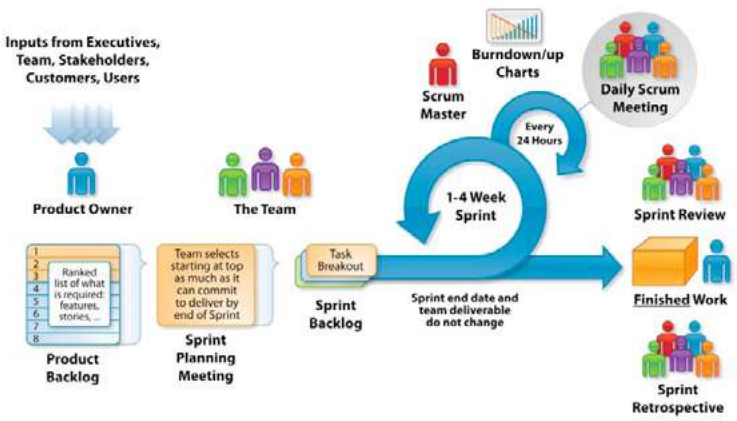
e le soluzioni evolvono dando la priorità ai clienti, grazie al lavoro di team collaborativi, interfunzionali e auto-organizzati. In totale si possono identificare 6 pratiche di gestione:

- L'uso del concetto di "product vision", che prevede l'impiego di strumenti visivi come lavagne, figure o disegni per fornire una semplice descrizione del progetto generale. Aiuta a identificare quali aspetti sono maggiormente apprezzati dai clienti o dal mercato e non a capire come sviluppare il prodotto. Utilizza metafore, figure e prototipi, differendo quindi dal modo tradizionale di presentare il design del prodotto (solitamente è sotto forma di WBS o di distinta base, con una descrizione prevalentemente testuale e dettagliata dell'attività da svolgere e dei parametri delle prestazioni del prodotto).
- L'uso di una comunicazione stretta, utilizzando semplici strumenti e processi di comunicazione del piano di progetto. Si fa riferimento alla comunicazione interpersonale, fra tutti gli stakeholder di progetto (cliente compreso) e serve ad avere una buona analisi dei requisiti e una proficua collaborazione tra gli sviluppatori.
- L'uso di una pianificazione iterativa: non è sviluppato un unico piano, ma l'intero progetto è spaccato su brevi iterazioni, all'interno delle quali c'è una pianificazione di dettaglio. Questa pratica consiste nella consegna rapida e continua di porzioni di prodotto, ottenendo costantemente un feedback da parte del cliente in modo tale da rispondere ai cambiamenti nel modo più rapido possibile.
- L'uso di team autogestiti e auto-diretti per lo sviluppo delle attività, che consente di garantire il pieno impegno e coinvolgimento di ogni membro del team.
- L'impiego di team autogestiti e auto-diretti nelle attività di monitoraggio e aggiornamento del piano di progetto, che aiuta a monitorare costantemente i progressi. Il coinvolgimento attivo del team, sia nella pianificazione che nel controllo del progetto, contribuisce a rendere più efficace l'interazione e la comunicazione tra gli individui, nonché lo sviluppo di figure professionali in grado di imparare, operare e adattarsi di fronte ad ambienti complessi.
- L'aggiornamento frequente delle attività del piano di progetto, soprattutto alla fine di ogni iterazione, è essenziale, a differenza della teoria tradizionale che prevede che il progetto debba essere rivisto dopo un importante milestone o in concomitanza della conclusione di una fase.

Implementazione pratica della filosofia Agile

La filosofia Agile può essere implementata tramite diversi framework di gestione che riflettono i suoi principi. Tra i framework più popolari possiamo ricordare Scrum, Kanban e Extreme Programming (XP), ognuno dei quali offre un approccio unico allo sviluppo Agile.

- Scrum è forse il framework Agile più diffuso ed è caratterizzato da sprint, riunioni quotidiane stand-up e una serie di ruoli chiave come il Product Owner e lo Scrum Master. Gli sprint, di solito della durata di una o due settimane, consentono ai teams di concentrarsi su obiettivi specifici e di fornire valore in modo iterativo.
- Kanban si basa sulla visualizzazione del flusso di lavoro attraverso una board Kanban, divisa in colonne che rappresentano le fasi del processo. Questo permette ai teams di monitorare il lavoro in corso, identificare eventuali blocchi e ottimizzare il flusso di lavoro per massimizzare l'efficienza.
- Extreme Programming (XP)mette l'accento sulla qualità del software attraverso pratiche come il pair programming, il test-driven development (TDD) e la refactoring continuo del codice. Questo approccio mira a ridurre i difetti e a migliorare la manutenibilità del software nel lungo termine.



Approccio Scrum

I vantaggi

Ma quali sono i vantaggi concreti dell'Agile nello sviluppo del codice?

1. Riduzione del Rischio: grazie alla consegna frequente di software funzionante e al feedback tempestivo, i teams Agile sono in grado di identificare e affrontare i problemi in modo rapido ed efficiente, riducendo così il rischio di fallimento del progetto.
2. Maggiore Adattabilità: l'approccio Agile consente ai teams di adattarsi rapidamente ai cambiamenti nei requisiti del progetto o nelle condizioni di mercato, assicurando che il prodotto finale sia sempre allineato alle esigenze del cliente.
3. Migliore Coinvolgimento del cliente: la consegna frequente di valore consente ai clienti di vedere il progresso del progetto in tempo reale e di fornire feedback tempestivo. Questo promuove un coinvolgimento più attivo del cliente e assicura che il prodotto soddisfi pienamente le sue aspettative.
4. Efficienza Operativa: le pratiche di automazione e testing continuo riducono i tempi di sviluppo e migliorano la qualità del codice, consentendo ai teams di concentrarsi su compiti di maggiore valore aggiunto e di massimizzare l'efficienza operativa.
5. Cultura del miglioramento continuo: l'Agile promuove una cultura di apprendimento e miglioramento continuo, in cui i teams cercano costantemente di ottimizzare i loro processi e di superare i propri limiti. Questo favorisce un ambiente di lavoro dinamico e stimolante in cui l'innovazione trova terreno fertile.

Le sfide

Quali sfide devono affrontare le organizzazioni che intendono adottare queste metodologie?
Nonostante i numerosi vantaggi, l'adozione dell'Agile può presentare alcune sfide per le organizzazioni, in particolare quelle abituate a pratiche più tradizionali.

- Cambiamento Culturale: L'Agile richiede un cambiamento culturale significativo, che coinvolge non solo i processi e le pratiche di sviluppo, ma anche le mentalità e le abitudini dei membri del team. Questo può richiedere tempo e sforzo per essere pienamente assimilato.
- Integrazione con Processi Esistenti: Per le organizzazioni con processi consolidati, integrare l'Agile può essere una sfida. È importante trovare un equilibrio tra l'adozione di pratiche Agile e il mantenimento di processi esistenti che funzionano bene.
- Comunicazione e Collaborazione: Anche se l'Agile promuove la comunicazione e la collaborazione, può essere difficile per i teams superare le barriere comunicative e lavorare in modo effettivo insieme, specialmente se lavorano in ambienti distribuiti o remoti.
- Gestione del Cambiamento: L'Agile richiede una maggiore flessibilità e adattabilità da parte di tutti i membri del team. È importante che i leader siano in grado di gestire il cambiamento in modo efficace e di fornire il supporto necessario per il successo dell'adozione Agile.

Conclusione

In conclusione, l'Agile rappresenta un approccio trasformativo allo sviluppo del software, che mette al centro la collaborazione, l'adattabilità e il valore per il cliente. Abbracciare i principi e le pratiche dell'Agile può portare a una maggiore efficienza operativa, una migliore soddisfazione del cliente e una maggiore competitività sul mercato. Tuttavia, è importante riconoscere che l'adozione dell'Agile è un processo continuo che richiede impegno, pazienza e un costante desiderio di miglioramento. Con il giusto approccio e il sostegno adeguato, le organizzazioni possono trarre enormi benefici dall'Agile e trasformare radicalmente il modo in cui sviluppano il codice e consegnano valore ai propri clienti.

https://agilemanifesto.org/

Keywords: Agile, Software Engineering, Software Development, Waterfall, Scrum, Kanban, Extreme Programming, Manifesto Agile, Just-In-Time, Team, Sprint, Framework

(*) Massimo Nannini
IT engineer and business consultant,
info@gemaxconsulting.it

THE AGILE APPROACH IN SOFTWARE DEVELOPMENT

In today's rapidly and continuously changing environment, companies must deal with new project development under increasingly uncertain, volatile, and complex conditions while trying to maintain competitiveness and catch market changes. Especially in the areas of research and new product development, uncertainty strongly affects the working methods of teams that find themselves unable to predict future results and events due to the difference between the amount of information required and that which is actually available.

Summary
Origin and evolution of the Agile model
The Agile Manifesto
Practical implementation of the Agile philosophy
The benefits
The challenges
Conclusion

Classical plan-driven development approaches, designed on a linearity basis, are overly constrained and unresponsive, as they do not allow for rapid accommodation of software development changes in progress. To counter this lack of responsiveness of more traditional methods, many companies have begun to rethink the way development processes are managed and understand the true value Agile model.

Origins and evolution of the Agile model

The Agile approach was born in February 2001, when 17 professional software developers gathered in Snowbird, Utah, to seek a viable alternative to traditional development techniques. All of the participants felt that the Waterfall model was not the most effective solution, especially in an environment where it is important to respond to constant change and thus possess some "agility" to deal with dynamic development.
The goal was to define a new methodology that could increase quality, improve flexibility, and accelerate time to market by limiting delays and waste and especially customer dissatisfaction due to the release of products that did not meet the required specifications.
The result was the development of an "Agile Manifesto," a document outlining the fundamental principles on which the Agile approach is based. These principles emphasize responsiveness to change, frequent delivery of working software, collaboration among individuals, and the ability to accommodate requirements even at an advanced stage of development.
Agile, a development framework for energizing, empowering, and enabling project teams to generate value quickly and reliably by engaging customers, learning, and continually adapting to the changing needs of their environment. Through this approach, a product is not planned or designed in advance, but the development process evolves by iterative cycles. These short sessions, called sprints, usually last 2 to 3 weeks, in which each team member must complete a series of assigned tasks (commonly called tasks). At the end of each sprint, the results obtained are analyzed to measure the progress of the work and the speed with which the development team is burning through the milestones. At this point, the process begins anew. Project management processes that incorporate these principles allow the customer to receive updates as they go along and signal priorities or introduce changes when needed.
So, Agile allows you to respond to change in a turbulent business by resorting to:

- strong customer interaction and integration during development phases;
- rapid reprioritization of resources;
- an evolutionary, incremental and iterative delivery of products;
- a just-in-time methodology that generates value by maximizing work not done.

Compared to traditional methodologies that aim to maximize profit (shareholders-based), Agile aims to maximize customer satisfaction, that is, the value generated (stakeholders-based). In addition, the process shifts from being prescriptive (rule-based) to adaptive (principle-based).

The Agile Manifesto

To fully understand the potential of Agile, it is essential to examine its core principles and practices that have been precisely stated within the "Agile Manifesto" divided into 12 principles and 4 values that express the ideas and thoughts of the founders:

Principles

1. Our top priority is to satisfy the customer by releasing valuable software, right away and continuously.
2. We embrace changes in requirements, even at advanced stages of development. Agile processes leverage change for the customer's competitive advantage.
3. We frequently deliver working software, ranging in cadence from a couple of weeks to a couple of months, preferring short periods
4. Clients and developers must work together on a daily basis for the duration of the project.
5. We base projects on motivated individuals. We give them the environment and support they need and trust in their ability to get the job done.
6. A face-to-face conversation is the most efficient and effective way to communicate with and within the team.
7. Working software is the main measure of progress.
8. Agile processes promote sustainable development. Sponsors, developers and users should be able to maintain a steady pace.
9. Continued focus on technical excellence and good design enhances agility.
10. Simplicity-the art of maximizing the amount of work not done-is essential.
11. The best architectures, requirements and design emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, after which it adjusts and adapts its behavior accordingly.

Values

I. Individuals and interactions rather than processes and tools.
The purpose of this value is precisely to emphasize the importance of having a balance between individuals and processes. Many projects today are characterized by high uncertainty, with a strong drive for innovation and creativity. In contexts like this, following a predefined and standardized process is never the optimal way to successfully complete a project. In the face of uncertainty and the exploration of new solutions, team members should have the freedom to experiment and always be open to learning; everyone should have the opportunity to create his or her own processes, customizing them to suit his or her own needs and goals. However, development teams are used to receiving predefined processes and tools from the company, thus remaining stuck in habits within a predictive culture, and this closes off openness to innovation, creativity and exploration of new solutions.

II. Functioning software rather than exhaustive documentation.
"Functioning software" means complete, tested, integrated software that meets the quality requirements and generates value for the customer. Quality is subjective and can change over the years, which is why it is always necessary to specify the requirements in a clear and measurable way, indicating what the quality requirement to be met is. As for exhaustive documentation, on the other hand, this means all that work that is not delivered to the client because it is worthless.

III. Collaboration with the client rather than contract negotiation.
This value is based on the need to always maintain a collaborative relationship with key project stakeholders, strongly based on the concept of transparency and trust. In this sense, it is necessary to formulate contracts appropriate to the situation, without aiming for continuous negotiation, but rather constant collaboration.

IV. Responding to change rather than following a plan.
Agile methodology does not involve the use of a Gantt, but this does not mean that planning is completely absent. Rather, it goes step by step with an adaptive type of planning that accepts changes to the project and is extremely flexible. This type of planning is not supported by a predefined path so as to orient the team to embrace change.

Practices

In addition to these clear and specific directions, within the Manifesto, we also find a set of "practices" based on iterative and incremental development cycles, in which requirements and solutions evolve by prioritizing customers through the work of collaborative, cross-functional and self-organizing teams. A total of 6 management practices can be identified:

- The use of the concept of "product vision," which involves the use of visual tools such as whiteboards, figures or drawings to provide

a simple description of the overall project. It helps to identify which aspects are most valued by customers or the market, not how to develop the product. It uses metaphors, figures, and prototypes, thus differing from the traditional way of presenting product design (usually it is in the form of a WBS or BOM, with a predominantly textual and detailed description of the task at hand and the product performance parameters).

- The use of tight communication, using simple project plan communication tools and processes. This refers to interpersonal communication, among all project stakeholders (including the customer) and serves to have good requirements analysis and fruitful collaboration among developers.

- The use of iterative planning: not a single plan is developed, but the entire project is split over short iterations, within which there is detailed planning. This practice involves the rapid and continuous delivery of portions of the product, constantly obtaining feedback from the client so that changes are responded to as quickly as possible.

- The use of self-managed and self-directed teams for task development, which allows for the full engagement and involvement of every team member.

- The use of self-managed and self-directed teams in project plan monitoring and updating activities, which helps to constantly monitor progress. The team's active involvement in both project planning and monitoring helps to make interaction and communication between individuals more effective, as well as the development of professionals who can learn, operate and adapt in the face of complex environments.

- Frequent updating of project plan activities, especially at the end of each iteration, is essential, unlike the traditional theory that the project must be revised after a major milestone or at the conclusion of a phase.

Practical implementation of the Agile philosophy

The Agile philosophy can be implemented through various management frameworks that reflect its principles. Popular frameworks include Scrum, Kanban, and Extreme Programming (XP), each of which offers a unique approach to Agile development.

- Scrum is perhaps the most popular Agile framework and is characterized by sprints, daily stand-up meetings, and a number of key roles such as Product Owner and Scrum Master. Sprints, usually lasting one to two weeks, allow teams to focus on specific goals and deliver value iteratively.

- Kanban is based on visualizing the workflow through a Kanban board, divided into columns representing process steps. This allows teams to monitor work in progress, identify any blockages, and optimize the workflow to maximize efficiency.

- Extreme Programming (XP)emphasizes software quality through practices such as pair programming, test-driven development (TDD), and continuous code refactoring. This approach aims to reduce defects and improve software maintainability over the long term.

The benefits

But what are the concrete advantages of Agile in code development?

1. Reduced Risk: Through frequent delivery of working software and timely feedback, Agile teams are able to identify and address problems quickly and efficiently, thus reducing the risk of project failure.
2. Increased Adaptability: the Agile approach enables teams to adapt quickly to changes in project requirements or market conditions, ensuring that the final product is always aligned with customer needs.
3. Better Customer Engagement: frequent delivery of value allows customers to see project progress in real time and provide timely feedback. This promotes more active customer involvement and ensures that the product fully meets customer expectations.
4. Operational Efficiency: continuous automation and testing practices reduce development time and improve code quality, allowing teams to focus on higher value-added tasks and maximize operational efficiency.
5. Culture of continuous improvement: Agile fosters a culture of continuous learning and improvement, in which teams constantly seek to optimize their processes and push their own limits. This fosters a dynamic and stimulating

work environment in which innovation finds fertile ground.

Challenges

What challenges do organizations considering adopting these methodologies face?

Despite its many benefits, the adoption of Agile can present some challenges for organizations, particularly those accustomed to more traditional practices.

- Cultural Change: Agile requires significant cultural change, involving not only development processes and practices, but also the mindsets and habits of team members. This can take time and effort to be fully assimilated.
- Integration with Existing Processes: For organizations with established processes, integrating Agile can be a challenge. It is important to strike a balance between adopting Agile practices and maintaining existing processes that are working well.
- Communication and Collaboration: Although Agile promotes communication and collaboration, it can be difficult for teams to overcome communication barriers and work effectively together, especially if they work in distributed or remote environments.
- Managing Change: Agile requires greater flexibility and adaptability from all team members. It is important that leaders are able to manage change effectively and provide the support necessary for successful Agile adoption.

Conclusion

In conclusion, Agile represents a transformative approach to software development that focuses on collaboration, adaptability, and customer value. Embracing Agile principles and practices can lead to greater operational efficiency, improved customer satisfaction, and increased competitiveness in the marketplace. However, it is important to recognize that adopting Agile is an ongoing process that requires commitment, patience, and a constant desire for improvement. With the right approach and the right support, organizations can benefit tremendously from Agile and radically transform the way they develop code and deliver value to their customers.

https://agilemanifesto.org/

Keywords: Agile, Software Engineering, Software Development, Waterfall, Scrum, Kanban, Extreme Programming, Agile Manifesto, Just-In-Time, Team, Sprint, Framework

(*) Massimo Nannini, IT engineer and business consultant, info@gemaxconsulting.it