SCRUM method: an AGILE method part 1

Par Christian DESTREMAU • Publié le 04/11/2017 à 10:54:40 • Noter cet article: ☆☆☆☆☆ (0 votes)
Avis favorable du comité de lecture

Christian DESTREMAU

Introduction :

Before talking about the SCRUM method, we must explain what the Agile method from which it comes.
The term "method" is a little reductive against it. There is also talk of Agile paradigm, Agile state of mind, philosophy or Agile approach, Agile movement. You will understand it better by reading the part describing this method, and especially the paragraph concerning the Agile Manifesto of software development.
The term "Agile" defines an IT project management approach that stands in contrast to traditional V- or waterfall-type predictive methods. The very notion of "project management" is called into question in favor of "product management". So as to reason more "product" than "project". Is not the purpose of a project to give birth to a product?
Agile methods assume that specifying and planning in full the details of a product (predictive approach) before developing it is counterproductive.
It's as if we were planning in the details a Paris-Béziers trip by car passing by the small roads. Doing this, we would specify each city and each village crossed, the associated crossing times, the streets taken in each agglomeration ... The unexpected as for them, will not fail to arrive: traffic jams, deviations, works, direction of circulation reversed ... Making your planning and your specifications quickly obsolete. How long will you spend planning this route unnecessarily and how will you react when you see the impossibility of applying your plan to the letter?
The idea of the Agile approach is to set a short-term goal (a big city, for example) and get started without delay. Once this first objective is reached, we take a break that is not too long and we adapt our itinerary according to the situation of the moment. So on until the final destination. This approach can be described as empirical.
As part of a software development, the customer defines his vision of the product to be produced and establishes the list of features of the latter. He submits this list to the development team, communicates directly with it (verbally), and estimates the cost of each item on the list. We then have a rough idea of the overall budget.
The team then selects a portion of the requirements to be achieved in a short time portion called iteration. Iteration includes design, functional specification, development, and testing. At the end of each iteration, the partial but usable product is shown to the client. The latter can therefore realize very early the work done and its correspondence to the needs. The visibility offered is a major asset of the method.
Transparency can also bring more trust and collaboration in the customer / supplier relationship. Risks as to the feasibility of the product are lifted very early on them.
If the customer has taken care to prioritize his needs carefully, he may choose to accelerate the "time to market" if he considers that the product in a partial state of manufacture can go into production. It will save its budget and reap a first return on investment. It can also change along the way the priority of features that have not yet been developed. It can, for example, delay a feature whose need is not obvious, add a new crucial feature in exchange for the withdrawal of another (without changing the budget), etc ... The flexibility offered by the method is a real asset to the customer.
The Scrum method is the most used methodology among existing Agile methods. There are plenty of books, blogs, trainings, videos, and lots of these resources are free. We can almost speak about it from an Agile standard. The SCRUM method has another advantage: it is easy to understand. On the other hand, it is not easy to control.

"predictive" method. Then we will briefly present the agile movement and the Scrum method, showing what differentiates "agile" methods from predictive methods.

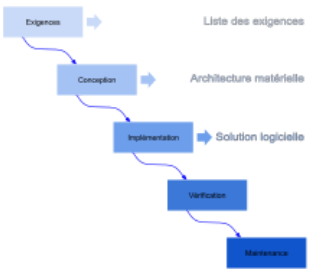
Project management, an organizational problem

By observing the business world, we find that everything is process: accounting process, human resources process, sales process and marketing.
Each process is carried out in specific departments. Thus we can note the existence of an accounting department, a department of human resources and a department of sales and human resources.
Each department has a leader (also called director) at the level of level "n". Below it are the "n-1", "n-2" ... to the more operational level.
IT is no exception to such an organization. In fact, we find the Director of Informatics Services (DSI) at hierarchical level "n", the Area Manager ("n-1"), the Project Team Managers ("n-1"), the Project Manager ("N-2"), and the computer developer.
We thus observe a strong notion of hierarchical dependence where each one strives, on a daily basis, to make respect the rules which are imposed on him. The project manager establishes the schedule, distributes the work to the developers of his team and ensures that the project does not drift. As for the developers, they code the application, making sure to respect the rules of the functional specifications of the future product.
To manage a computer project, it consists therefore of respecting rules of good behavior in order to register correctly in the process of the company.

The cascading model of project management

The rules to be applied in most companies regarding project management are as follows:

1. Have an idea of the project
2. Have a written expression of needs, according to a formalism established by the company, identifying the functional requirements of the client
3. Maybe write a contract, based on this expression of needs, where we find commitments such as deadlines, costs ...
4. Writing of technical specifications by computer designers (UML diagram)
5. Start of IT developments based on functional and technical specifications
6. Overall application revenue (unit test, functional test, integration test).
7. Delivery and installation of the application



The notion of dependencies between phases posed many problems. Imagine, for example, that a misunderstanding has been introduced in the technical specifications (step 4). If we take our model, we can deduce that it will be detected, in the best case, only during functional tests (step 6).
To correct this error, it will be necessary to modify the technical specifications, redo the development, rewrite and replay the tests. This will be done at a cost that is not better to imagine.
The solution to this problem was found in the 80s by the creation of the V model well known to all and still used in business.

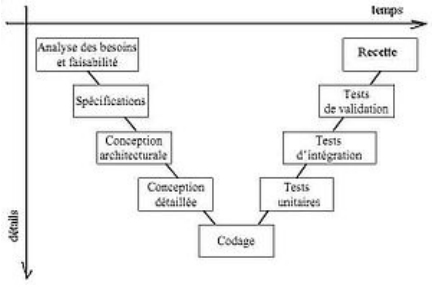
The V model

This model remains the most widely used IT management model in the enterprise.
To compare it with agile methods, it must be said that it remains a sufficient model for projects that are not very complex.
This model must be put into perspective because it comes in two forms. There is a theoretical V model that we are going to address now and a practical V model actually applied in business.
The theoretical approach of the V model is that any action taken in the cascading model has its anticipation of "how it should happen". This requires some explanation.
The V-shaped model has 8 steps according to the following list:

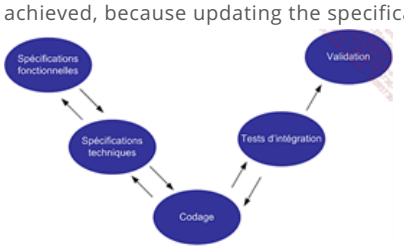
1. Writing the expression of needs
2. Writing of functional specifications in the form of a written document. This is the specification of the final product as desired by the customer.

5. Realization of unit tests. These tests make sure that each functional brick works properly and without any apparent bug.
6. Performing integration tests. These are the first full-scale tests of the finished product. It must be ensured that the delivered product complies with the recommendations of the technical specifications.
7. Realization of functional tests. The product must comply with the functional rules described in the functional specification.
8. Production and recipe. The product is checked one last time in a "pre-production" environment and then installed on the production environment. The customer proceeds to the recipe. This means that it checks that the software meets the functional requirements and that there are no major bugs.

The diagram below shows a notion of anticipation and parallelism of tasks. We see that while the functional specifications are written, it is possible to anticipate the validation phase by writing the validation tests (step 1 to step 8). The same goes for the integration phase (step 6), which can be anticipated once the drafting phase of the technical specifications has been completed (step 3).

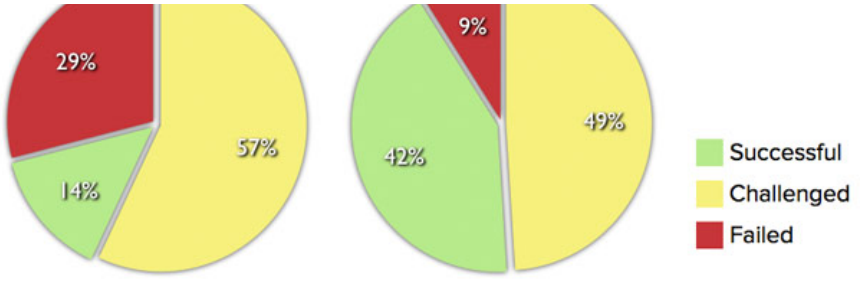


This anticipation makes it possible to carry out tests very quickly and thus to anticipate problems. For example, an error or inconsistency in the early detected functional coverage will allow developers to avoid unnecessary coding.
The practical approach of the V-shaped model, however, is different.
We observe in the real world upward and downward communication in the drafting phases of specifications and coding.
In this way, if during the writing of the technical specifications is perceived a lack in the functional specifications, an update of these is made and vice versa.
The same goes for coding. An error or an oversight was made in the specifications? It is then sufficient to update the technical and functional specifications.
This method has a disadvantage that must be pointed out: that of creating discrepancies between what has been imagined and what is achieved, because updating the specifications is often omitted.



The V model incorporates roles, generally known to all, as the MOA and the MOE.
The MOA (project management) is responsible for the functional aspect of the product. She is responsible for defining the need and ensuring that it is respected during the recipe phase.
The MOE (project management) group including project managers, developers and infrastructure designers is responsible for writing the detailed specifications, carry out the development without forgetting the integration tests.
The V cycle is unfortunately victim of a very disadvantageous phenomenon: that of the tunnel effect.
When we are interested in a tunnel, we know the point of entry, exit but what happens inside is completely unknown to us.
Thus, the users (that is to say the MOA) are very involved in the drafting phase of the needs but when it comes to the more technical phases (architecture, development,) they are totally set to gap.
These, once the technical phases are completed, are then invited to visualize the work done by the MOE.
Between the phase of needs expression and delivery of the product, it can happen a few months and the need of customers may have changed (because of business constraints, legislative changes,). As a result, the developed product does not meet the new requirements.
The tunnel effect of the V-cycle has the effect of depriving users of the vision of the product as long as it is not finished and so we end up with unused or non-compliant features.
The company Standish Group has highlighted this phenomenon in a report established in 2009, which is entitled "Report Chaos", which describes the percentages of use of the features of an application.
According to this, only 20% of the features in an application are really useful to users. This means that 80% of the features that have been developed and required an investment have been in vain.
This report also details the successes and failures of the projects.

- 32% were completed on time, with no budget overruns and expected features
- 44% were overdue or over budget or with incorrect or missing features
- 24% were discontinued or completed but never used



Source: The CHAOS Manifesto, The Standish Group, 2012.

This comparison shows that the agile model has 3 times more success than the cascade model.

Agility in project management

The Agile Manifesto is composed of 4 values and 12 principles.
The agile method was created in 2001. 17 people from the computer world came together and established what will become the Agile Manifesto. It is available for download at the following address: http://agilemanifesto.org/.
The 4 values that make up the agile manifest are the following:

- Promote individuals and their interactions rather than setting up processes and tools
- Promote collaboration with the customer rather than contractual negotiation
- Make sure to get immediately available software rather than well-documented documentation that turns out to be ultimately useless
- Be responsive to change rather than following a plan

The objective is to privilege the interactions with the individuals while delivering a technical result of quality.
The 12 principles contained in the manifesto make it possible to understand the philosophy of the agile method:

- Give high priority to customer satisfaction through close software deliveries
- Accept change of needs even late in development
- Frequently deliver software that runs on a regular schedule, from 2 weeks to 2 months
- Make users and developers work together daily
- Build projects around motivated people. Give them the environment and support they need.
- Focus on face-to-face communication as the most effective way to convey information to development teams
- Consider operational versions of the software as the key measures of progress
- Sponsors, developers and users must be able to maintain a constant pace indefinitely
- Give continuous attention to technical excellence and good design to improve agility
- Privilege simplicity. That is, the art of maximizing the work not to do
- Consider that the best architectures, needs and designs emerge from self-organized teams
- Reflect at regular intervals on how to become more effective

Scrum, an agile method

In 2001, Ken Scwaber and Mike Beedle officially release the description of their agile method named "Scrum" in Agile Software Development With Scrum.
The name "Scrum" which means "Melee" expresses the strong concept of the method revolving around team cohesion. Indeed, it is the entire team (Product Owner, Scrum Master, development team) who will work to achieve the same goal.
The team must be welded as the players of a rugby team during a scrum in order to seize the ball. The goal is of course not immediately attainable but through several iterations allowing the team to visualize its progress and in particular to take into account the modifications of the need, not forgetting to remove the obstacles that it encounters.
The Scrum method, as with any agile method, is characterized by respect for the values and principles of the agile manifesto:

- The existence of a self-organized
- Defined roles: Scrum Master, Product Owner, Development Team
- A set of ceremonial (daily meeting, sprint review ...)
- The constant presence of the client
- The establishment of mechanisms favoring frequent deliveries (sprint)
- An acceptance of change (but not at any price)

However, it is necessary to emphasize that Scrum is a method and not a set of software engineering techniques. Scrum is therefore a methodological framework that must be complemented by technical practices. For example, we can say that the method advocates the implementation of tests, but it does not indicate how to perform them.

Lean, Kanban and eXtreme Programming

Instead of just describing the Scrum method and getting an overview of agility, we'll also describe other agile methods like Lean Management, Kanban and XP.
Indeed the agility is not necessarily associated with Scrum and on the other hand we will see that some practices used in these methods have been taken over by Scrum.

Lean Management

The goal of Lean Management is primarily to reduce waste during production phases. It translates to "fine management" and was invented and implemented by Toyota. It consists of analyzing all production processes and optimizing them in order to eliminate any non-value added factor.
The implementation of Lean is first initiated by a phase of analysis of production methods: time spent, tasks performed, etc. Once this phase is over, decisions are then made to reduce any unnecessary action: the moving an employee to bring a piece may be deleted if the piece is directly brought to him.
Lean also advocates increasing the skills of employees. In the automotive world, if the person is able to mount a vehicle door, it can be optionally formed to make the seals.
Optimization involves the human, but also the methodological and financial factors of the company. For example, it is not necessary to store useful parts for the construction of a vehicle if no order is placed. It is better to order these parts when the need arises to minimize storage costs.
This method is based on the following principles:

1. Base decisions on a long-term philosophy, even at the expense of short-term financial goals
2. Organize processes in a "piece-by-piece" flow to update problems
3. Use drawn systems to avoid overproduction
4. Smooth the production
5. Create a culture of immediate resolution of quality issues the first time
6. Standardization of tasks is the foundation for continuous improvement and empowerment of employees.
7. Use visual control so that no problem remains hidden
8. Utiliser uniquement des technologies fiables, longuement éprouvées, qui servent vos collaborateurs et vos processus
9. Train leaders who know the job well, live the Lean philosophy and teach it to others
10. Train exceptional individuals and teams who apply your company's philosophy
11. Respect your network of partners and suppliers by encouraging them and helping them to progress
12. Go to the field to understand the situation
13. Decide by taking the necessary time, by consensus, by examining in detail all the options. Apply decisions quickly
14. Become a learning enterprise through systematic thinking and continuous improvement

After a few years of implementation, there is a 30% improvement in costs for companies that apply this method.
The Lean method is indeed agile because it allows an adaptation to the context of the company by using techniques such as introspection, visualization and constant sharing of information. This method is widely applied in Japan, it suffers from findings against it when it is implemented in Western countries, because of a culture and perception of work different.

Kanban

This method is also derived from the industrial world and also the automotive world, because it was also developed by the company Toyota.
In fact, by stating the principles of Lean Management, we have referred to the notion of kanban in principle n ° 3 related to the concept of fugitive flows.
The idea is simple and starts with the translation of the Japanese word kanban which means etiquette. The customer's request is translated into labels representing each production requirement. Each label defines the product to be manufactured, as well as the quantity needed.
The person in charge of a production unit receives all the labels. He deposits them on a board and sorts them according to their orders of reception. However the number of labels that he can display on his board is limited in order to make possible the realization of the products to realize.
Once the work is complete for a given label and production service, the label is passed to the next production unit manager where other tasks are performed.
The first advantage of Kanban is to allow an immediate visualization of the tasks to be done because of the presence of a table listing them. The second advantage is that the number of labels is limited, hence limiting the workload for members of a production unit.
We can make a connection between Kanban and Scrum by first transcribing the need on a label. This need becomes a User Story and labels it a post-it in the Scrum method.

The XP or eXtreme Programming method

EXtreme Programming (XP) is an agile method based on common sense and observation of everyday practices for application development. The goal of this method is to push these practices to the extreme and to group them together as a coherent whole.
The eXtreme Programming is based on the following variables:

- The time
- The cost
- The quality
- The scope

Rather than considering monolithic development, XP advocates modular development. The time is therefore easily assessable which goes hand in hand with the cost that is better controlled.
Quality is a fixed variable that can not vary over time, unlike others. It wants to be continuously optimal. Finally, the extent of the application, ie the number of functionalities developed, depends on the budget and the time allocated by the client to the project.
The implementation of XP is organized around 13 practices:

a. Frequent deliveries
   The method advocates the division of an application into modules, which are then delivered at regular intervals
b. sustainable pace
   In order to deliver on a regular basis, it is necessary to respect a constant and sustainable pace of work (this is one of the principles of the agile manifesto)
c. Customer on site
   Only the customer has the possibility to define the functionalities to be developed according to his budget and the time allotted to the project
d. Simple design
   At the level of development, the rule is as follows: "Develop in the simplest possible way and respond correctly to the need".
e. Implementation of coding rules
   The encoding must respect rules in order to create homogeneity in the code of the application, which will further facilitate the recovery of the code or its modification.
f. The team is responsible for the code
   The team is responsible for the code made by one of these members to be free of bugs. The team must also have a collective knowledge of the code.
g. Use an understandable naming
   The names of classes, variables, objects and functions must have obvious meaning for all. We obtain a code that is readable and interpretable by anyone who wishes to reuse or modify it.
h. Using unit tests
   EXtreme Programming advocates the implementation of test-driven development (TDD). The difficult parts of the code must be the subject of 2 cases of unit tests, one verifying the good functioning of the developed code, the other causing an error. These tests are regularly repeated in order to note the technical non-regression of the application during the various functional additions.
i. Recipe test
   These tests are intended to verify that the application meets the functional requirements of the customer. They must be completely replayed at each delivery to ensure non-functional regression.
j. Establishment of continuous integration
   When a development is completed, it is immediately integrated into the final product.
k. Realize code refactoring
   This operation consists of reworking the code to make it more efficient. It is necessary to avoid duplication of method, unnecessary loops or other technical cases that alter the performance of the application.
l. Pair Programming
   This principle consists of performing the programming task with a partner. The developer responsible for writing the code is the Driver, he is assisted by the Partner who suggests other coding methods or detects possible problems related to the code set up.
m. Estimation with Planning Poker
   Il s'agit d'utiliser un jeu de cartes dont les valeurs faciales sont basées sur la séquence de Fibonacci. Ces cartes sont ensuite utilisées pour estimer l'effort de réalisation d'un scénario.

Le cycle standard de la méthode XP comprend les étapes suivantes:

1. Le client écrit ses besoins fonctionnels sous forme de scénarios
2. Les développeurs évaluent le coût de chaque scénario en collaboration avec le client
3. Le client choisit les scénarios à inclure dans la prochaine livraison
4. Chaque développeur prend la responsabilité d'une tâche dans un scénario
5. Le développeur choisit un partenaire
6. La paire écrit les tests unitaires correspondant au scénario à mettre en œuvre

Scrum, un mélange de méthodes

Nous allons maintenant mentionner différents éléments de la méthode Scrum, directement inspirés des méthodes que nous venons de décrire.
La méthode Scrum conserve les éléments suivants du Lean Management: adaptation, introspection, utilisation post-it, grand intérêt pour la qualité.
Il conserve les éléments suivants du kanban: écrire le besoin du client sur une étiquette, prioriser le besoin, limiter la quantité de travail pour éviter la surcharge
Enfin, il conserve les éléments suivants de la méthode de programmation eXtreme: effectuer des livraisons fréquentes, avoir un rythme de travail durable, une responsabilisation renforcée, mettre en place des tests et utiliser le poker de planification pour l'estimation

Conclusion

Le but de cet article était de décrire le contexte dans lequel la méthode Scrum a été construite: nous avons d'abord décrit la méthode classique de gestion de projet, en la contrastant avec le mouvement agile dans lequel s'inscrit Scrum. Ensuite, nous avons brièvement décrit d'autres méthodes agiles que Scrum a repris de certaines des pratiques. Nous sommes maintenant bien mieux placés pour discuter de la description de la méthode Scrum elle-même. Cette présentation fera l'objet d'un deuxième article intitulé Scrum, une méthode agile partie 2.

Références bibliographiques

Site Web: http://www.agiliste.fr/introduction-methodes-agiles/
Site Web: https://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall
Livre: Scrum, une méthode agile pour vos projets (éditions ENI)