infolink®

OUR WORK PROCESS:
THE AGILE METHODOLOGY

THE MANIFESTO

WE ARE UNCOVERING BETTER WAYS OF DEVELOPING SOFTWARE BY DOING IT AND HELPING OTHERS DO IT. THROUGH THIS WORK WE HAVE COME TO VALUE:

> INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS
>
> WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION
>
> CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION
>
> RESPONDING TO CHANGE OVER FOLLOWING A PLAN

THAT IS, WHILE THERE IS VALUE IN THE ITEMS ON THE RIGHT,
WE VALUE THE ITEMS ON THE LEFT MORE.

Manifesto for Agile Software Development Retrieved from www.agilemanifesto.org/

THE BREAKDOWN

THE COMPONENTS OF THE MANIFESTO.

INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS

Recognizing that software is made by people, not processes or tools, agile places a higher premium on people working together effectively. Processes and tools can aid in that but can't replace it.

WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION

A highly detailed, accurate, and comprehensive specification document is of no value if it doesn't result in working software that meets users' needs. Working software may involve documentation, but agile only uses it in service to creating working software, not as an end (almost) unto itself.

CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION

While agile isn't ignoring the reality of contracts, it values active collaboration throughout the software development process as a better way to deliver value instead of a carefully worded contract.

RESPONDING TO CHANGE OVER FOLLOWING A PLAN

Except for the most incredibly simple systems, it's massively difficult to think of every feature, every piece of data, and every possible use case for software. That means, in a collaborative process with the customer, a lot is discovered during the process of developing software. Also, the world changes pretty fast: Business needs and priorities can shift in the months or even years it can take for a large system to be fully built. Agile values the ability to change in response to new discoveries and needs over sticking to a plan created before everything was known.

THE PRINCIPLES

THE AGILE MANIFESTO FOLLOWS THESE PRINCIPLES:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to 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 designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective and then tunes and adjusts its behavior accordingly.

THE ROLES

ON A DISCIPLINED AGILE PROJECT, ANY GIVEN PERSON CAN BE IN ONE OR MORE ROLES, AND HE CAN ALSO CHANGE HIS ROLE(S) OVER TIME.

STAKEHOLDER

A stakeholder is someone who's impacted by the outcome of the solution. A stakeholder may be one of the following:

- A direct or indirect user
- A shareholder or investor
- A client or a vendor
- An executive, senior or department manager
- An external regulatory agency

PRODUCT OWNER

The product owner is the team member who speaks as the "one voice of the customer." This person represents the needs and desires of the stakeholder community to the agile delivery team. He clarifies any details regarding the solution and is also responsible for maintaining a prioritized list of work items that the team will implement to deliver the solution. While the product owner may not be able to answer all questions, it's his responsibility to track down the answer in a timely manner so the team can stay focused on its tasks.

- Communicates the project status and represents the work of the agile team to key stakeholders
- Develops strategy and direction for the project and sets long- and short-term goals
- Understands and conveys the customers' and other business stakeholders' needs to the development team
- Gathers, prioritizes, and manages product requirements
- Directs the product's budget and profitability
- Chooses the release date for completed functionality
- Answers questions and makes decisions with the development team
- Accepts or rejects completed work during the sprint
- Presents the team's accomplishments at the end of each sprint

SCRUM DEVELOPMENT TEAM

The role of scrum member focuses on producing the actual solution for stakeholders. Team members perform:

- Business analysis
- System Architecture and design
- Writing Test Scripts and testing
- Programming
- User Interface and Experience
- and many more activities as appropriate throughout the project.

SCRUM MASTER

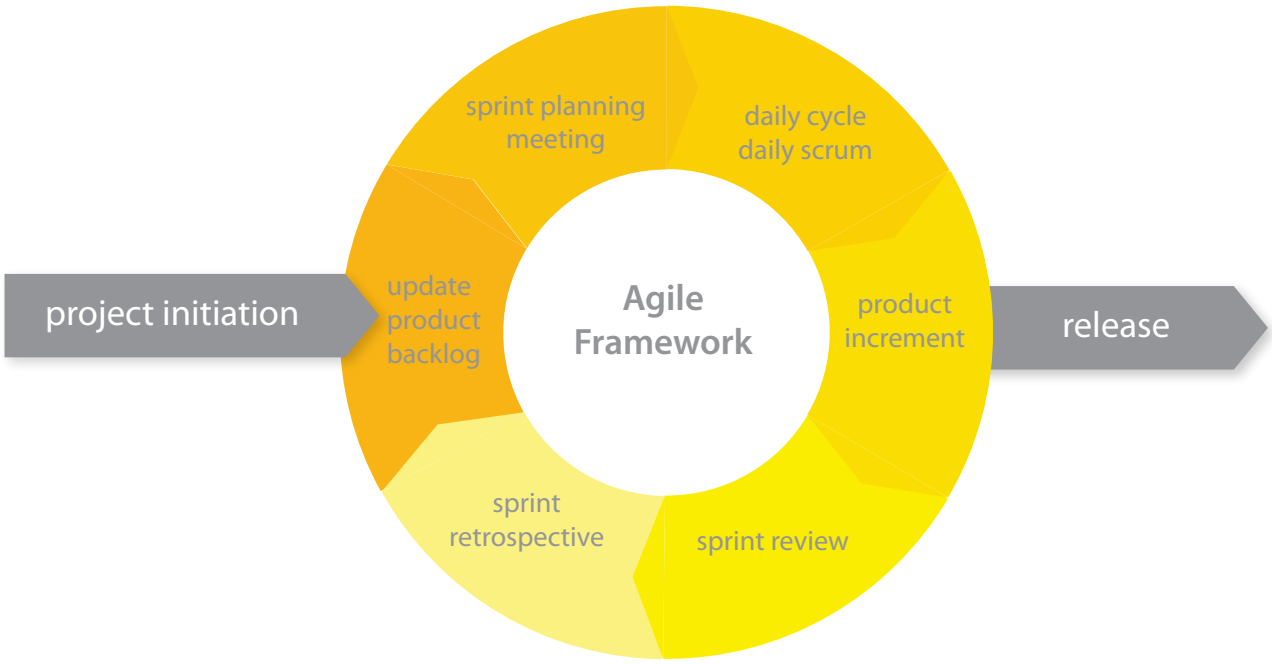
The team lead guides the team in performing management activities instead of taking on these responsibilities herself. She's a servant-leader to the team, upholding the conditions that allow the team's success. This person is also an agile coach who helps keep the team focused on delivering work items and fulfilling its iteration goals and commitments to the product owner.

ADDITIONAL ROLES

Project may include the need to add some or all the following roles:

- Domain expert: Someone with deep business/domain knowledge beyond that of the product owner.
- Specialist: Although most agile team members are generalizing specialists, sometimes, particularly at scale, panelists such as business analysts or even project/ program managers are required.
- Technical expert: Technical experts are brought in as needed to help the team overcome a difficult problem and to transfer their skills to one or more developers on the team.
- Independent tester: Sometimes we are supported by an independent test team working in parallel that validates work throughout the life cycle.
- Integrator: For complex environments, we may require one or more people in the role of integrator responsible for building the entire system from its various subsystems.

HOW IT WORKS



PROJECT INITIATION

Understanding your business will add context to our work and will make sure that the developed solution is aimed toward accomplishing the business objectives and is aligned with your organization's mission.

We typically start our projects with a management level workshop (2 – 3 hours) that will enable us to:

- Answer all your questions
- Understand your business model (Tool used: Business Canvas)
- Understand your vision, strategic directions and what problem we are solving
- Discuss the Agile approach and the impact on your environment
- Gather current artifacts (documents related to the project you currently use)

STORIES

Agile takes a lightweight approach of writing down brief descriptions of the pieces and parts that are needed. These become work items and are captured in the form of user stories. A user story is a simple description of a product requirement in terms of what that requirement must accomplish for whom. Use stories needs to have, at a minimum, the following parts:

TITLE: Creation of Document Template
Name for the User Story

AS A: Manager of the Credit Department
User or Persona

I WANT TO: Create Template Documents
Take this action

SO THAT: We improve speed in document delivery and we maintain consistency.
I get this benefit

The story is not a formal requirement but rather an invitation to have a conversation. Anyone on the project team can create a user story.

For user stories that are too large we use EPICS. Epics are basically a higher-level story that's fulfilled by a group of related user stories. As an example, developing the Inventory Module of the solution could be an Epic. Ordering inventory items could be user story related to that Epic. The product owner gathers and manages the user stories.

User stories aren't the only way to describe product requirements. We could simply make a list of requirements without any given structure. However, because user stories include a lot of useful information in a simple, compact format, they tend to be very effective at conveying exactly what a requirement needs to do. The team usually has a conversation with the Product Owner, and if necessary the Stakeholders, to have better understanding of what should be done.

ACCEPTANCE TEST

The story should also include validation steps, steps to take to know that the working requirement for the user story is correct.

The acceptance test will help the developer and tester understanding and testing the requirement and it will also helps the Product Owner testing and accepting the completed work. That step is worded as follows:

GIVEN THAT: A template is created for a specific purpose
Condition Required

WHEN: I select this template from a client file
Take this Action

THEN: Client data is merged and document is automatically created.
This Happens

- Each story will have a unique identifier.
- Stories are ranked according to dependency and priority.
- Stories are reviewed and approved by Product Owner.
- Team expects re-prioritization, additions to the list, and subtractions from the list throughout the process but embraces them as a means to deliver the most value and the best possible solution.

ESTIMATION

We allocate value points to each story. Points represent a size-based, complexity-based approach to estimation. Points are assigned in whole numbers and represent relative sizes and complexity of work items. Small and simple tasks are one point tasks, slightly larger/more complex tasks are two point tasks, and so on. Our velocity, or the number of value point we can complete on average during a two week sprint is 50 points.

SPRINT

We schedule a series of fixed-length development iterations of one to four weeks. Each iteration is called a Sprint. Shorter sprints are generally better.

Planning involves selecting the stories to include, scheduling the work and assigning individual work items to members of the team. We use an automated planning tool "Analyst", developed by infoLink to manage our sprints.

SCRUM MEETINGS

We make plans throughout the entire sprint daily. Our development teams start each workday with a 15 minute (or less) daily coordination meeting to note completed items, to identify impediments, or roadblocks, requiring team lead involvement, and to plan their day. If unable to attend in our office, we use WEBEX to communicate with Product Owner and Stakeholders.

TEST-DRIVEN DEVELOPMENT

Testing occurs throughout the sprint work. Performing TDD means that before the developer writes a piece of code, a small test is written that validates the code the developer is about to write. The test is run to make sure it fails and then writes the code that makes the test pass.
We also use unit testing. When these tests are run in batches all at once (automated), they become very powerful. Our teams write a lot of unit tests, automate them, and run them frequently against the code they write as individuals and against their combined code that makes up the entire application.

CONTINUOUS INTEGRATION AND DEPLOYMENT

When the build is successful on our environment, tested and accepted by the product owner, we automatically deploy the changes to the client environment.

RETROSPECTIVE

The iteration retrospective is a meeting where the team lead, the product owner, and the development team discuss how the iteration went and what they can do to improve the next iteration. If the team wants to, other stakeholders can attend as well. If the team regularly interacts with outside stakeholders, and it should, then those stakeholders' insights can be valuable.

THE MYTHS

AGILE IS A FAD

The agile approach to project management is far from a fad. Agile has been in use for many decades even though it was only recently formalized with the Agile Manifesto and its associated principles. Agile exists because it works. Compared with traditional project management approaches, agile is better at producing successful projects.

AGILE ISN'T DISCIPLINED

Sometimes agile can seem chaotic because it's a very collaborative process. Agile is a departure from the rigid assembly-line process. The iterative approach requires rapid response times and flexibility from the team. In fact, agile demands greater discipline than what's typical of traditional teams. Agile requires teams to reduce the feedback cycle on many activities, incrementally deliver a consumable solution, work closely with stakeholders throughout the life cycle, and adopt individual practices which require discipline in their own right.

AGILE MEANS "WE DON'T PLAN"

With agile's reliance on collaboration instead of big documents, it may seem like no planning occurs. But in reality, the planning is incremental and evolutionary, which has been proven successful (instead of planning all at once early in a project).

AGILE MEANS "NO DOCUMENTATION"

Agile teams keep documentation as lightweight as possible, but they do document their solutions as they go. They follow strategies, such as documenting continuously and writing executable specifications.

AGILE IS ONLY EFFECTIVE FOR COLLOCATED TEAMS

Sure, ideally agile teams are located within proximity of one another, but in this day and age, most development teams are distributed. Just remember, to succeed, you need to adopt practices and tooling that build team cohesion. If you use the proper tools, your team doesn't have to be collocated to work effectively together.

AGILE DOESN'T SCALE

Agile definitely scales. Large teams must be organized differently. They need more than index cards and whiteboard sketches. Large agile teams succeed by using proper tools.

AGILE IS UNSUITABLE FOR REGULATED ENVIRONMENTS

Regulated environments are those that are subject to some regulatory mandates, such as medical device companies, business in the finance area, governmental departments and offices, the healthcare field, and more. These organizations are audited from time to time for compliance with regulations. With agile, these organizations can feel confident when they endure these audits. They benefit from faster delivery of data and higher quality of their output.

AGILE MEANS WE DON'T KNOW WHAT WILL BE DELIVERED

Because agile is an iterative process, it provides the opportunity not just for greater control but better control over building the right things in the life cycle than one would have with the more traditional Waterfall approaches. At the end of each iteration, the development team presents a completed product to the product owner for feedback. Furthermore, Disciplined Agile Delivery (DAD) teams explicitly explore high-level requirements at the beginning of the project and seek to gain stakeholder agreement around the requirements.

AGILE WON'T WORK AT MY COMPANY

For many companies, the biggest challenge they face when considering changing to agile is the cultural change making the change requires. Agile has explicit means of frequent feedback and loops, which means that developers and managers may feel more exposed to scrutiny. But that doesn't mean that agile won't work at your company.

Agile is a team approach. It's not like football where one player at a time moves the ball down the field to score a touchdown. Agile is more like rugby — the whole team moves the ball down the field together. Roles are cross-functional and shared. Developers become testers and more frequent delivery creates more exposure and personal accountability.

For an organization to successfully adopt agile, executive support is also critical. Agile can succeed without it, but if you hit a bump in the road, you'll want that vote of confidence to help you keep everything moving in the right direction.

IT'S ENOUGH FOR MY DEVELOPMENT TEAM TO BE AGILE

For agile to work properly, all teams have to buy in. So if your development team is gung ho, but your testing team is blasé, you won't get your best results. Your agile delivery process is only going to be as effective as your slowest group. To make agile succeed at its greatest potential, make each piece of the chain as efficient as possible.

AGILE IS A SILVER BULLET

Agile isn't needed for every team in every situation. It isn't a cure-all. Agile is a superb solution for projects that are in development or undergoing radical changes. For other projects, such as those that are in maintenance mode, agile isn't as good a fit. If your project has a stable customer base and isn't undergoing a lot of change in the code, you may not need to use agile for that particular project. But for projects that are under new product or rapid development, agile really is the best way to go.