The Agile Compass checklist can help development teams identify which outcomes they have accomplished

### Beginning to try agile
The team is trying to use an agile method with empirical learning. Most of times, not all practices were implemented. Benefits are not fully realized.

- We are learning on the fly
- We did not realize the expected benefits of agile adoption
- We are still learning how to deliver value to our customer

### Learning about agile
The team is learning the agile method. The method is used "by the book" usually by getting training or coaching. Practices are fully implemented, as described in books and tutorials. It is important for the team to learn the value of each practice to be able to, later, tailor or even abandon them.

- We are following an agile method "by the book"
- We are getting training or coaching
- We are working with pilot projects

### Making sense of work processes
The team understands agile practices and the value they add. They can now tailor the practices to their context and feel secure to be flexible in work processes, without abandoning agility.

- We learned the agile method and understand the value of each practice
- We are currently tailoring the practices
- We feel the work processes are more flexible according to the characteristics of each project

### Comprehending the situation
The team has information about the work they perform. Information is simple and usually based on physical visibility or simple metrics. They use this information to make decisions and implement improvements in the process. These improvements are usually small and incremental.

- We are able to explain what happens in our work processes
- We have some metrics
- We use our understanding to make decisions

### Responsive team
The team needs close leadership to drive activities. They do not have/were given autonomy to make decisions. The team does not have the confidence to protect their work, that is, overtime is usual because of extra demands.

- We have partial autonomy to make decisions (about design, technology, project etc.)
- We feel the need to have the Scrum Master with us most of the time
- We have weak politics to protect team work

### Confident team
This team still needs the Scrum Master around, but feel secure to make little decisions mainly about the project. Their communication is fluent. Changes in work processes need to be approved by management.

☐ We are fluent with practices and feel autonomous to some decisions in the project
☐ We know each other and are free to communicate
☐ We suggest improvements in the process, but they usually need approval

**Assertive team**

This team feels responsible for the project. They also have the autonomy to change the process. They make their own decisions and inform management.

☐ We have the autonomy to improve our work processes
☐ We have the capability to drive our own work during the project
☐ We have politics to protect our work

**Team leads continuous improvement**

This team is focused on technical excellence. They create their work processes, supported by management. They need a supportive environment to continue learning and, thus, motivated. It is possible to feel the focus on improving constantly.

☐ We define our work processes and change them whenever needed
☐ We feel highly motivated to continue learning
☐ We feel leadership supporting the implementation of our ideas

---

**Code finished at end of sprint**

Sprints are planned to finish code and integrate in repository. Most of times, functionality is not tested yet. The team is learning how to deliver value in the end of the sprint.

☐ In the end of the sprint, we have code finished, but we do not deliver it
☐ Our sprint size sometimes varies
☐ We are not sure of what adds value to our customer

**Functionality finished at end of sprint**

This team identifies the value to be delivered but cannot deliver in the end of the sprint. The environment is not fully supportive for agile build and code integration. Test is usually manual.

☐ We are capable of having functionalities ready to deliver in the end of the sprint, but we do not deliver
☐ We can identify a minimum releasable product

**Deliveries usually late**

Deliveries are planned and done, but usually late. Sometimes extra demands appear and the team has to address them. Environment is more supportive for automated build and integration.

☐ We plan our deliveries and do it, usually late
☐ Our stories sizes vary

**Deliveries done by deadline**

Deliveries are planned and done within deadline, sometimes before. Stories are usually small.
Environment supports automated build and integration.

- We plan our deliveries and do them, sometimes before deadline.
- Our stories sizes are similar, usually small

Requirements gathering

Requirements are defined in the beginning of the project, usually with texts or diagrams. The team does not feel comfortable with changes or emerging requirements during the project.

- We define most of the requirements in the beginning of the project
- We do not know exactly how to deal with changes in requirements during the project

Requirements discovery

Requirements are vaguely defined in the beginning of the project, and detailed as sprints are initiated. Changes and emergent requirements are welcome. Spykes might be used for requirements elicitation.

- We define with details the requirements for the next sprint
- We are comfortable with changes or emergent requirements
- We can identify a minimum releasable product in our stories

Requirements quality

There are clear initiatives to guarantee the quality of the requirements, for them to be in accordance to customer needs.

- We are concerned with whether what we deliver to our customer is exactly what the customer needs
- We endeavor initiatives to improve the quality of the requirements

High-level source code

Coding is an important activity and there are initiatives to guarantee the code is clear and robust, using best-practices available.

- We care about our code
- We endeavor initiatives to guarantee our code is fine, such as reviews, refactoring, pair programming or others

Awareness of failures

There are failures in deliveries that need to be treated as soon as possible, as well as failures happen because of problems in development process.

- We spend some time in the sprint correcting bugs from the previous sprint
- We spend some time in the sprint with issues in code building and integration

High-level software

There is a concern with the quality of the feature being delivered. For this reason, test is a priority, either automated or not. Also, there is a concern with the integrity of the code in repository.

- We see test as a priority
- Code is integrated as soon as possible
Our code repository is always healthy

 Efficient coding

Coding, integration and testing is performed with an infrastructure that allows agility. The team is concerned with removing unnecessary delays in work processes. Devops is a common infrastructure to allow efficient coding.

- We have automated tests, either unit or functional tests
- Our build and integration are automated
- We are concerned with removing delays in work processes, by automating manual tasks

Team awareness of customer

The team is learning how the customer business is and the demands dynamics. This is the reason their contribution to customer requirements is still incipient.

- We are getting to know our customer and the demands
- We have partial ability to help customer defining requirements

Customer awareness of team

It is the learning process of the customer about the team. The customer is getting to know the processes, but is not fully used to that. Sometimes it seems there is a lack of trust on the team.

- Our customer is learning how we work
- The customer knows what we are going to deliver
- The customer is not fully comfortable about reprioritizing requirements

Confident customer

The customer is confident about team's work. He trusts the team is concerned about delivering what he needs. The customer is usually more flexible about deliveries and is more comfortable with reprioritizing requirements. There is transparency between the team and the customer.

- We assist customer in requirements definition
- Customer is aware of its role in the process
- We feel the customer trusts our work

Partner customer

The customer recognizes the partnership with the team: while he feels the team is committed with his business, the team feels the customer as being part of the team.

- We identify and suggest improvements for our customer business
- Our customer feels as begin part of our team, with shared responsibility about deliveries

A few teams using agile

Some isolated teams are starting to work with agile. The company knows, but is not concerned in acknowledging the process or the results.

- We see isolated agile initiatives in our company
- We feel weak acknowledgment of top management about these initiatives
Agile commitment

Top management decided to implement agile in the company and initiatives are official. There are investments with training, coaching, communication and infrastructure for agile transformation.

- There are official agile pilot projects in the company
- We feel company support (training, coaching or infrastructure) for agile transformation

Agile priority

There is full support from top management for agile transformation. Departments, roles, teams change to support agility. When it is a top-bottom initiative, resistance still appears in some teams.

- We see that the company' structure (physical and departmental) has changed to support agile transformation
- We feel there is some resistance to change in some teams

Agile business

Software development companies that are created with agile methods are, usually, agile businesses. The company management strategies are focused on people and leaness, not just for software development, but for the whole company.

- Our whole company is recognized as being agile
- Our teams adopt an agile foundation, but customize the way they work according to the projects characteristics