Engineering meets Agile

PAPER

Rev. 1.5.0 – 12 March 2025

AgileEngineering

an AgileConstellation Star

Summary

1. Introduction ... ***4***

2. AgileEngineering ... ***5***

2.1 AgileEngineering: an AgileConstellation Star ... **5**

2.2 AgileEngineering Fast Prototyping ... **7**

2.3 The phases of AgileEngineering ... **8**

3. The Inception phase ... ***10***

3.1 Set Based Design ... **10**

3.2 Intentional Architecture ... **12**

3.3 Metrics for the Inception Stage ... **12**

4. The Engineering phase ... ***13***

4.1 The organization of work ... **14**

4.2 The Teams ... **16**

4.3 Quality and metrics ... **18**

4.4 Engineering Metrics ... **19**

5. The Workout phase ... ***21***

5.1 Manufacturing Production ... **23**

5.2 Metrics for the Workout phase ... **23**

6. Conclusions ... ***25***

1. Introduction

The world **of Systems Engineering** is constantly looking for new operating models that allow it to innovate its products quickly, while making the production chain more efficient.

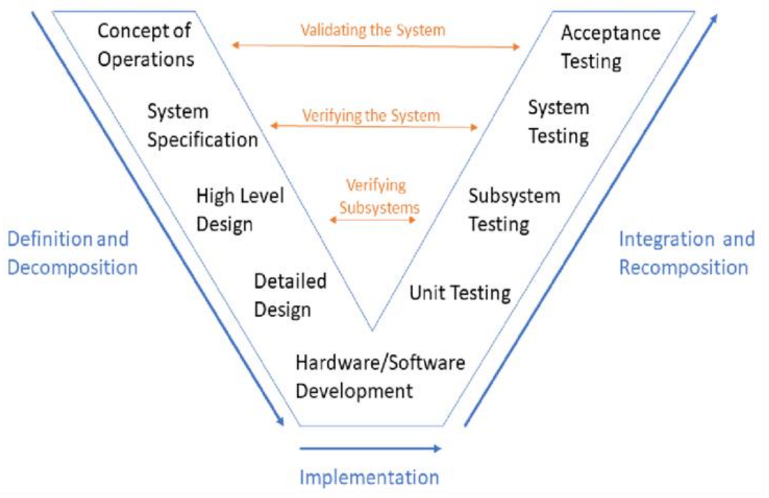
Typically, in it, a linear lifecycle is adopted that follows well-defined sequential steps and represented in the so-called *V-shaped model* characterized by 2 flows:

1. **Specification flow**, which is expressed in the activities on the left
2. **Test flow**, which runs along the right line

The two flows are connected by the **implementation,** where the artifacts are made.

However, this model is not very flexible, with a clear division between design/planning and implementation, making changes and adaptation to emerging needs expensive.

Therefore, the fragility due to the detail of the requirements remains intrinsic, which clashes with reality where customers are able to understand what they really want only in the most advanced stages of product creation, precisely where the costs of change often become prohibitive. In addition, projects are often long-term (several years) and impact on many different domains, which strongly emphasizes the importance and complexity of the overall governance and integration of the various constituent elements.



Pushing towards Agility, it is clear that traditional models related to the digital world must be adequately adapted, taking the form of the metaphorical zeroing of the *V angle*: the *specification flow* and the *test flow* tend to overlap completely, marrying development activities internally, reducing the use of external teams.
This makes it possible to speed up the **feedback system**, in particular that relating to customers, in order to capture the inevitable changes to be made when the product is still in a "raw" phase, where the related costs are less disruptive.

2. AgileEngineering

AgileEngineering operationally supports these aspects, stimulating engineering teams in the adoption of a "*fail fast*" (or *succeeding fast*) logic, directing development in relation to specific objectives that are developed on validable elements of the short, medium and long term.

Since the products of system engineering are "tangible artifacts", AgileEngineering suggests dividing the operational phases of development into three specific moments: *Inception*, *Engineering* and *Workout*.

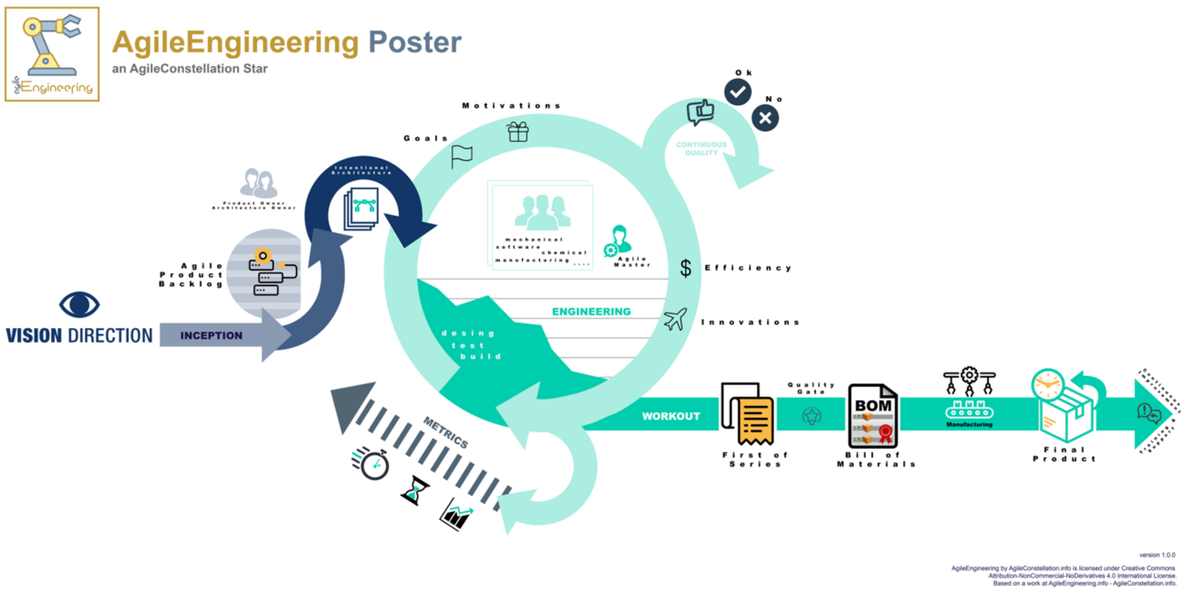


Figure 1 – AgileEngineering Poster

2.1 AgileEngineering: an AgileConstellation Star

AgileEngineering is constitutionally based on the Agile and Lean mindset, boasting the declination developed as part of the AgileConstellation project that looks at domains other than software and digital in general.

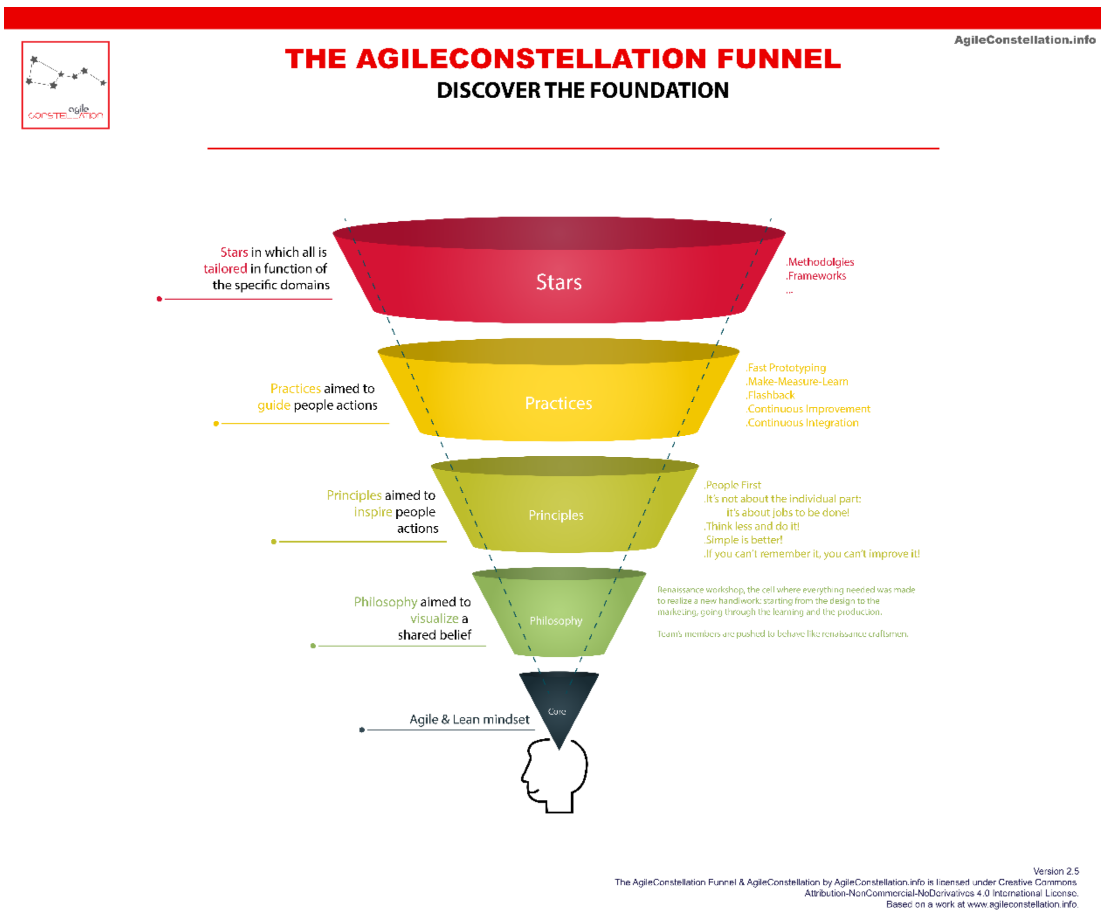

AgileConstellation.info
THE AGILECONSTELLATION FUNNEL
DISCOVER THE FOUNDATION
Stars in which all is tailored in function of the specific domains
Stars
.Methodolgies
.Frameworks
...
Practices aimed to guide people actions
Practices
.Fast Prototyping
.Make-Measure-Learn
.Flashback
.Continuous Improvement
.Continuous Integration
Principles aimed to inspire people actions
Principles
.People First
.It's not about the individual part: it's about jobs to be done!
.Think less and do it!
.Simple is better!
.If you can't remember it, you can't improve it!
Philosophy aimed to visualize a shared belief
Philosophy
Renaissance workshop, the cell where everything needed was made to realize a new handiwork: starting from the design to the marketing, going through the learning and the production.
Team's members are pushed to behave like renaissance craftsmen.
Agile & Lean mindset
Core
Version 2.5
The AgileConstellation Funnel & AgileConstellation by AgileConstellation.info is licensed under Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
Based on a work at www.agileconstellation.info.

Figure 2 - AgileConstellation funnel

AgileEngineering therefore inherits the mindset, philosophy, principles (core) and practices (core) of AgileConstellation, declining and extending, in particular, the latter two aspects in the specific domain of IoT.

We therefore have:

- **Philosophy**, inspired by the **Renaissance Workshop**, or the cell that fulfills what is necessary for the creation of a new work: from design, to construction and marketing.
- **Principles (core):**
  - It is not a question of the individual parts: it is the whole that must be done well!
  - Think less and act sooner!
  - Simple is better!
  - If you can't remember it, you can't improve it!
- **Practices (core):**
  - *Fast Prototyping*, validating the sustainability of the solution
  - *Make-Measure-Learn*, quickly experiment with different assumptions and assumptions
  - *Flashback*, a quick alignment action in which the observer engages in observing the element being worked on

- *Continuous Improvement*, constantly improving every aspect
- *Continuous Integration*, constantly integrating the different souls of the solution

2.2 AgileEngineering Fast Prototyping

In accordance with the modular approach of the AgileConstellation Manifesto, Star *AgileEngineering*[1] adds 6 new domain bubbles to the Fast Prototyping practice to validate the sustainability of the solution:

- **Security**: focused on testing the hypotheses relating to the security aspects that characterize the entire solution and that directly influence its development.
- **Energy**: focused on the verification of hypotheses relating to energy aspects according to the operational continuity needs of smart devices.
- **Hardware**: focused on validating hardware hypotheses through one or more *Evaluation Kits* (EVKs). The selection of the most suitable EVK passes, first of all, from the choice of the CPU/MCU, after which you start to "build" the prototype by working on the other components (e.g.: RAM, USB, etc.).
- **Code**: focused on the prototyping of the firmware of the devices and on that of the support services for the acquisition of the data/events carrying the solution. In this phase, the use of fast coding frameworks and IDEs is essential to reduce the relative times.
- **Data**: focused on aspects related to the collection, cleaning and management of *Raw Data* from devices, implementing the aspects of transmission and serialization of the same, with the choice of the appropriate protocols and the most suitable formats. In particular, it is essential to pay particular attention to the estimation of data volumes and the related analysis methodologies, in order to approach information management in a *Polyglot Persistence key*, which is essential to ensure the possibility of quickly processing large amounts of information.
- **Cloud**: focused on the Cloud aspects of the solution, intended as a platform for managing data, events and key actions.

[1] www.agileconstellation.org

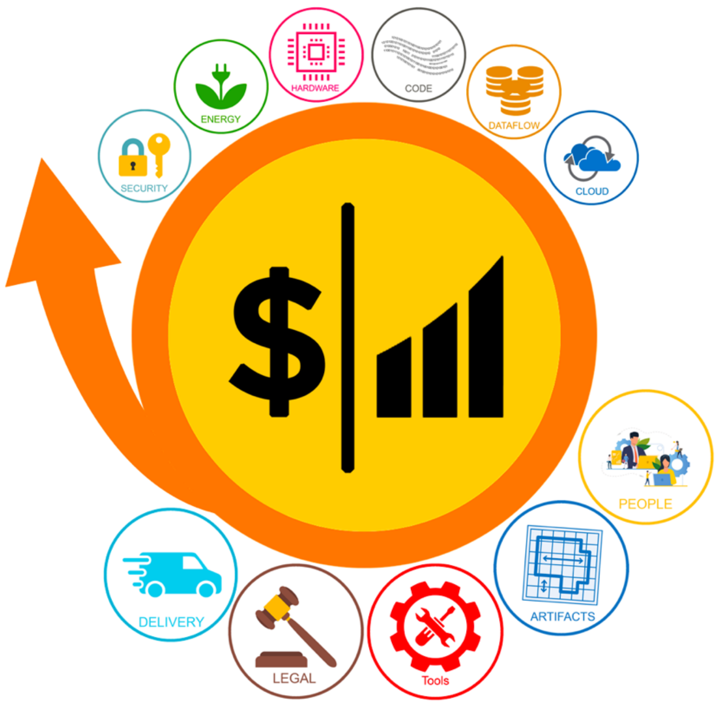


Figure 3 - The new 6 bubbles added by the AgileEngineering domain

2.3 The phases of AgileEngineering

In the *Inception* phase , what is necessary to validate the system that is about to be built is carried out, focusing on the development of those enabling components that will constitute its supporting infrastructure. All this is based on a "discovery", often characterized by a Lean-based approach, which allows rapid discussions with stakeholders. Rapid *prototyping* and *digital twin* methods are therefore adopted, exploiting emerging technologies (such as, for example, 3D Printing and Mixed Reality) in order to reduce costs and allow new prototypes to be quickly developed with which to engage users and collect feedback.
In this context, the use of *MVP* (Minimum Viable Change), understood in its correct sense as a *tool for learning*, is fundamental and supports everything in a concrete way.

When a condition of relative stability has been reached, the *most malleable parts (i.e. the most easily evolvable components) are engineered, making the appropriate consolidation developments on the less ductile ones (e.g., standard components acquired from the market,* buy-in*).*

Finally, the *Workout* phase deals with managing the industrial production (series) of the whole and governing the delivery aspects, as well as those of operation.

The operation in the different phases is entrusted to **multidisciplinary teams**, i.e. teams composed of a mix of experts in the disciplines involved (electronics, mechanics, chemistry, software, etc.), who carry out the various activities necessary to implement the capabilities that characterize the product.

The challenge of **AgileEngineering** is to use all the known techniques of the Agile and Lean world, to promote communication in teams and between teams, in which, as mentioned, there are experts from different domains often used to working in relation to their skills rather than according to a product objective that looks at the end customer. To this end, to encourage the creation of a "common language", it is essential to use *Visual Management* tools, such as task boards or, in more advanced cases, a real Kanban-like approach that allows you to maximize both transparency on the status of activities and the identification of bottlenecks on which to intervene as soon as possible.

Thanks to AgileEngineering it is possible to reduce the risks of becoming non-competitive, stimulating an adaptive approach capable of reducing lead time and generating innovative solutions while maintaining the reference quality standards. In fact, the impact that reference regulations have on the general process should not be underestimated, as they are often binding and cannot be waived for the marketing and use of the product itself.

Among the main advantages that can be observed from the adoption of AgileEngineering are:

- *Reduction of Lead Time, Time to Market and Time to Value*, thanks to the introduction of fast prototyping tools and the organic use of MVP
- *Improvement of the efficiency of processes or workflows*, taking advantage of solutions inspired by Agile and Lean, together with their explicit visualization.
- *Reduce the communication gap between engineering teams and* stakeholders by developing rapid feedback and constant alignment.
- *Improvement of quality*, both "real" and "perceived".

Some specific declinations of system engineering concern, for example, the Automotive, Aerospace, Rocket companies, Medical Device companies and so on.

Each of these domains can take clear advantage of what has been highlighted so far, further declining the specificities in its own context and identifying the explicit tools that best support the overall vision and current operations.

3. The Inception phase

One of the doubts that immediately arises when we talk about Agile applied in the field of System Engineering is how it is possible to iteratively develop a complex system, often seen as an all or nothing, composed of physical elements whose ductility is anything but an element to be underestimated.
Often, the approach followed is to focus on the creation of successive prototypes (*Set Based Design*), in which, from time to time, specific aspects of the different components are refined and then arrive at the "*First Series*" and the specific definition of the *Bill of Materials*.

To extricate itself from this aspect, **AgileEngineering** identifies a specific **Inception** phase, in which the primary aspects of the solution are refuted and the **development backbone** necessary to give the detailed operational start is created.

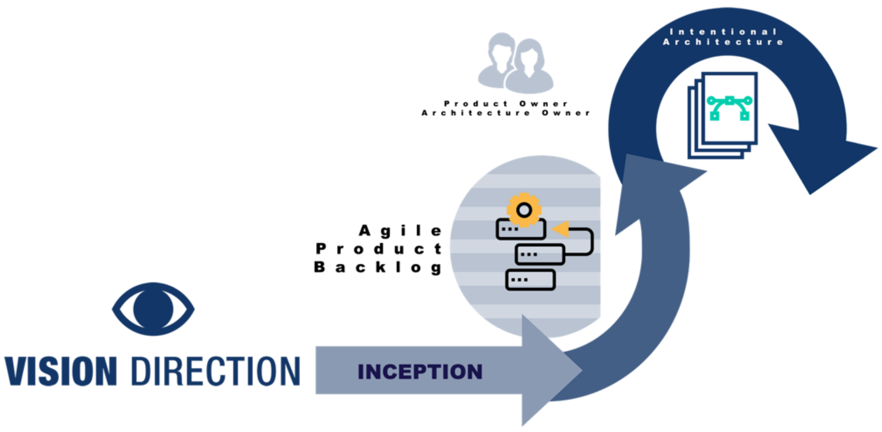


Figure 4 - AgileEngineering - Inception phase

In this phase, a *Lean based* approach is essential, given the variability of the actions to be carried out and the experimentation that generally characterizes the product in this juncture in a predominant way.

3.1 Set Based Design

In the action of identifying the solution, the approach called **Set-based Design (SBD) is used**, also known as the "second paradox of Toyota", which allows to consider a wide range of possible decisions acquiring, however, an advantage that then reduces the subsequent interventions and therefore the development cycle.

SBD is opposed to the traditional linear process known as **point-based design** that starts from an initial concept and proceeds with more and more details of the same. If, at first glance, this method seems very efficient, since you work on a single concept from start to finish, it does not take into account that the world is not linear and the challenges are often more complex than they seem.

With point-based design, if new constraints or requirements emerge from the customer during the design process, you are forced to start all over again, sometimes even going back to square one.

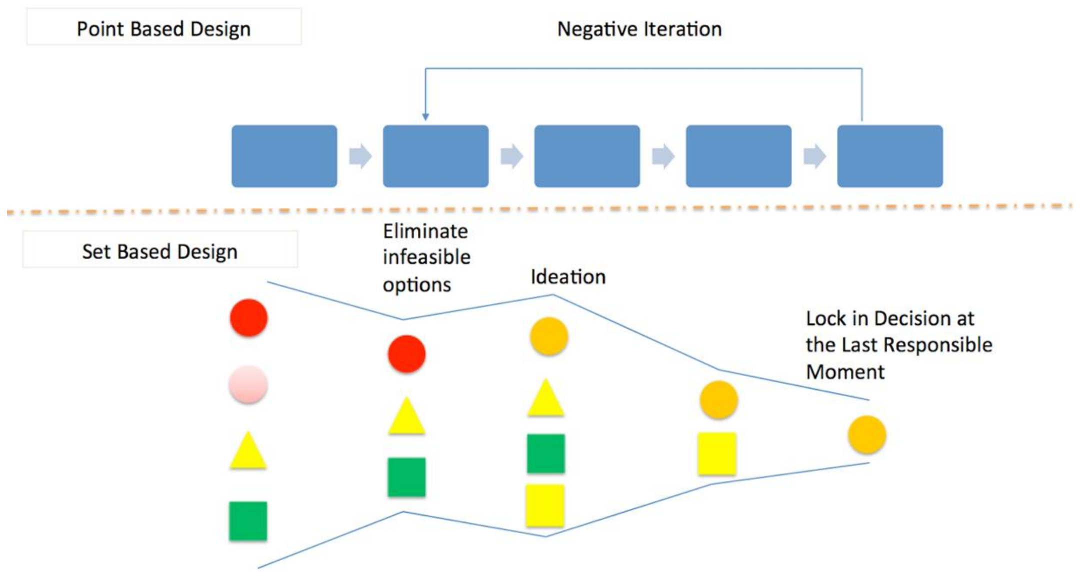

Point Based Design
Negative Iteration
Set Based Design
Eliminate infeasible options
Ideation
Lock in Decision at the Last Responsible Moment

Figure 5 - Point Based vs Set Based Design

Dually, rather than selecting a single promising option and working exclusively on that, *Set-based Design* explores a wide range of possible options: sets *of solutions* are progressively reduced until they converge towards a final solution. During the design process, some options are eliminated due to rigid constraints, feasibility issues, or lack of suitability. In addition, ideation can be used to generate new options: only at the **Last Responsible Moment** (LRM), a decision will be made and one of the remaining options selected.

By starting the process with a broad set of options and progressively discarding weaker solutions, you can evaluate more possibilities and find better solutions.

If there are unexpected constraints or new needs, you reduce the risk of having to start over, as one or more of the remaining options may already meet the new requirements. If necessary, you only go back one or two steps, instead of starting from scratch.

The key principles of Set-based Design are:

- *Define feasible regions*
- *Explore trade-offs by designing multiple alternatives*
- *Communicate Possibility Sets*
- *Look for intersections between feasible sets*
- *Impose as few constraints as possible*
- *Progressively reduce sets by increasing the level of detail*
- *Staying within sets once decisions have been made*
- *Make decisions and select the final option at the Last Responsible Moment.*

In practice, it is as if the Lean principles, generally designed for the optimization of as-is and the reduction of waste, undergo a **shift-left** effect, hooking into the development of new products and finding application when decisions can have the greatest influence on both the product and the operation.

The solution identified, together with the assets built and the desired plan, will constitute the passe-partout for the **Construction** phase in which everything will be developed in detail and finished to obtain the final product that can satisfy the user and that is, clearly, industrially sustainable.

The central aspect of the Inception phase is the ability to generate know-how among team members and between the team and the customer, driving continuous sustainable innovation.

3.2 Intentional Architecture

A key aspect of the Inception phase is **Intentional Architecture**, an approach that balances the flexibility of architectural choices with the need to provide a solid structure on which to build the product.

Unlike emerging architecture, which evolves exclusively based on iterative developments, Intentional Architecture involves:

- *Defining the system foundation*: Identify the key modules and components that must be stable to ensure scalability and sustainability.
- *Guiding principles for architectural decisions*: Create a framework for technology and design choices.
- *Adaptability*: Structure the architecture so that it can evolve over time without requiring a total redesign.

3.3 Metrics for the Inception Stage

Key metrics for assessing the effectiveness of the Inception phase include:

- *Time to Prototype (TTP):* the time it takes to develop a first working prototype.
- *Stakeholder Feedback Cycle (SFC):* Average time it takes to collect and integrate stakeholder feedback.
- *Experiment Success Rate (ESR): The* percentage of prototyping experiments that provide valid results for subsequent iterations.
- *Change Cost Index (CCI):* A measure of the average cost of changes implemented during this phase compared to those made later in development.

4. The Engineering phase

In the **Engineering phase** , the product is implemented in the various functional aspects*, customer side* following a more typically Agile logic.

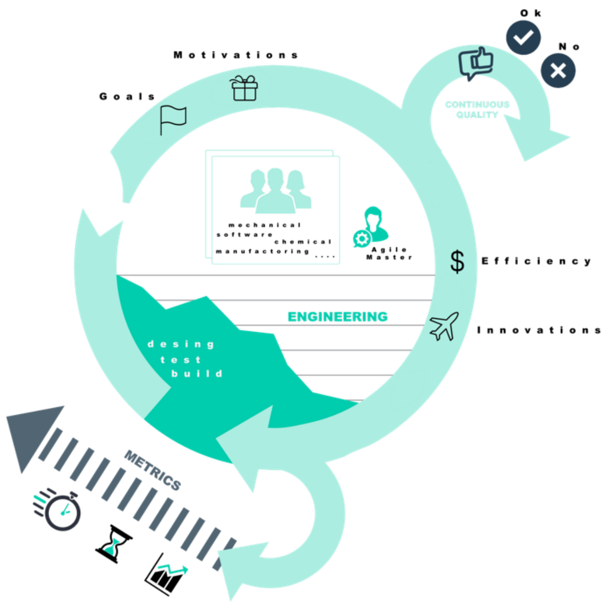


Figure 6 - AgileEngineering - Engineering phase

This phase is guided by a "**functional tree**", an output/outcome element of the previous *Inception phase*, in which the various components characterizing the system have been defined, at medium-high level.

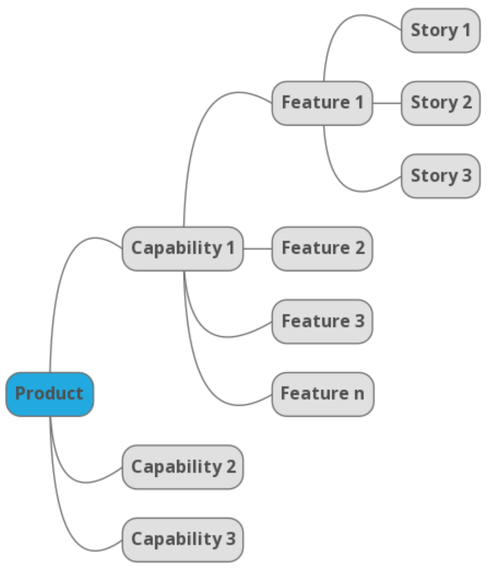


Figure 7 - Functional Shaft

The use of the functional tree helps to focus on the functionalities to be implemented in relation to the customer's needs, favoring the organization into cross-functional teams. This is a very delicate aspect, especially in the field of system engineering which, normally, is based on a strong sectoralization of skills and subdivision of activities into similar "*working-packages*" (i.e. WBS).
The functional tree is clearly accompanied by an *architectural view* that helps to understand the components to be used for the implementation of the required functionalities, allowing an initial feasibility and sustainability assessment to be made, which is also the basis of the backbone development activities, already presented in the discussion inherent to the Inception phase.

4.1 The organization of work

The composition of the teams is a particularly delicate aspect to take care of: generally, there are several teams focused on functional parts of the system that do not have continuous contact with the end customer, also because for the latter the minimum object to be evaluated is a **Capability**, which represents a significant functional (but also at the level of components) sub-set. This Capability typically has a long development time, which can even exceed a year, so it is necessary to find a way to shift the focus of review and alignment at least on the *Features* (which reasonably have a time declination of a few months).

Even in this case, however, there are considerably longer time intervals than generally expected in the Agile world (2, 3 weeks on average), which leads to reconsidering the role of Product Owner of each team more from an **Engineer Owner** perspective who helps the team in the detail of the activities and in the coordination with the other teams or work areas involved. A "**product ownership board**" is therefore set up consisting of Engineer Owners, generally one for each team who, as a whole, coordinate with the Product Owner (of the product) who supports them in customer relations and in the review of the **Agile Portfolio** through which the objectives are defined in the various time references (*Annual, Quarter, Monthly, Iteration, etc*.), as well as operational backlogs.

In particular, the *Flight Level Architecture* is used, in which the Strategic level is moved from the Operational level (and vice versa) for a complete view of how product development is progressing.

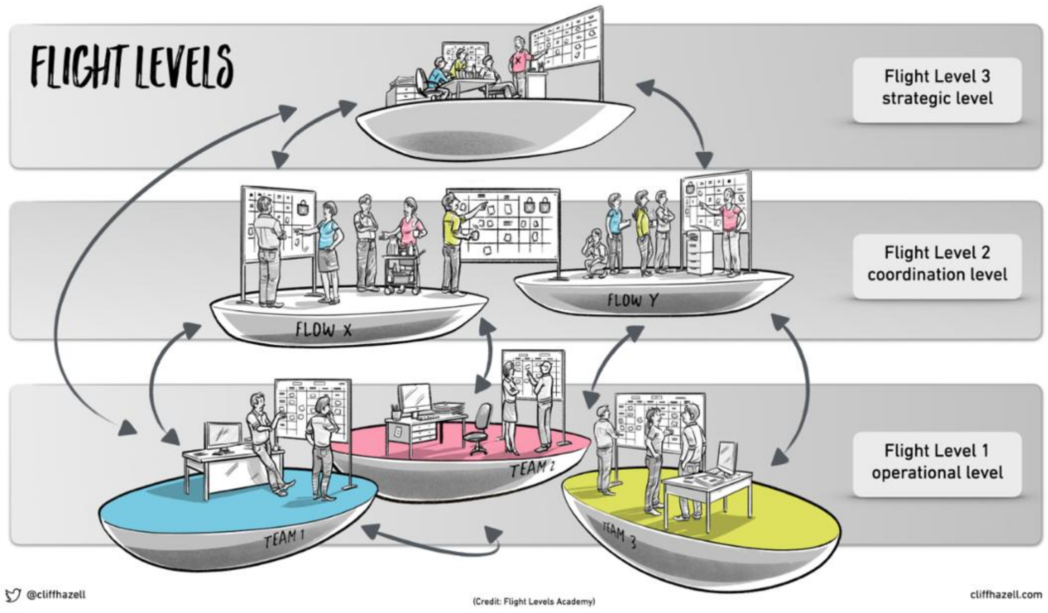
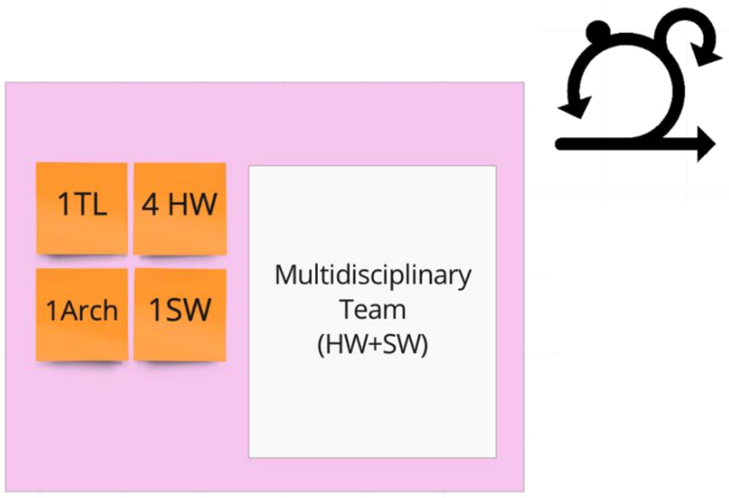


Figure 8 - The Flight Levels Architecture

Each capability can be traced back to the concept of **Minimum Business Increment** (MBI), representing an **atomic element** that has meaning from the point of view of business sustainability, i.e. both for the customer but also for the organization's objectives.

The real focus of cross-disciplinary development (software, electronic, mechanical, chemical, etc.) is concentrated on it, thanks to its progressive breakdown that gives life to the *operational backlog* in which the direct interventions to be carried out on the product are carried out.

A team inspired by AgileEngineering is, therefore, a team made up of experts from the various related sectors who coordinate with each other to complete a capability in its different dimensions, significantly reducing the complexity of integration and testing that is found when development is instead centered on domain components made independently of each other.



4.2 The Teams

In line with the Agile philosophy, it is essential to carefully distinguish between: *intangible resources* and *human resources* (*People*), so as never to lose sight of the focus on the development of a collaborative environment capable of constantly improving.

However, one clear fact remains: having a team with all the skills necessary to deal with the development of a product is not always efficient from the point of view of optimizing their allocation. This is also valid for the same *motivation of people*: a person who makes a marginal contribution during an iteration and has to "find" the activities to do, does not develop a valid sense of belonging because he cannot find his specific operational vocation.

This highlights 4 specific factors, here defined as **GEMI** , to be taken into account in the creation of teams, in order to adequately stimulate people and create an "environment" suitable for achieving increasingly cutting-edge goals:

- **Goals**, i.e. the ability to have shared, clear and well-identified objectives
- **Efficiency**, all activities must always take into account the optimization of available resources in relation to context constraints
- **Motivations**, i.e. the ability (here managers play a key role) to adequately motivate People, creating an appropriate path of "social" and professional growth
- **Innovations**, always working on challenging initiatives that stimulate a continuous realignment of the *State of Flow*, i.e. the condition of greatest stimulus for people.

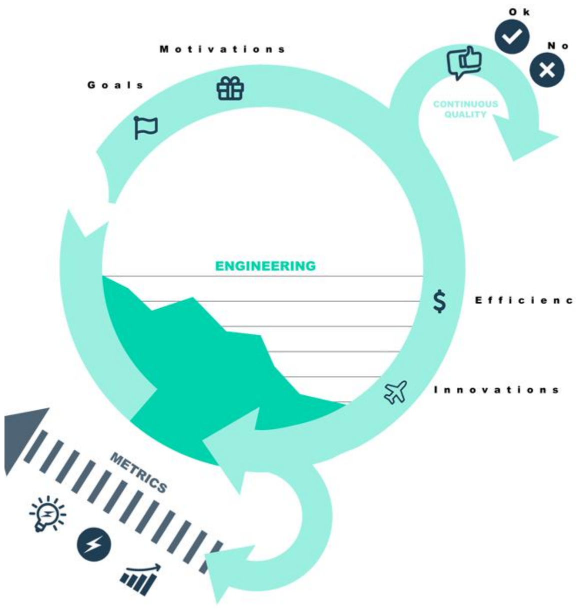


Figure 9 - GEMI factors

The different specialists are organized as much as possible into *multidisciplinary teams* focused on objectives, although a specific effect must be emphasized. If it is true, in fact, that this greatly favors the ability to be focused on an overall objective of value (and not local), it is also true that when the activities are extremely specialized (for example relating to aspects deriving from regulations impacting the product) this choice is not always successful, as it is not possible to "replicate" the related skills for each team and, objectively, it does not find reasonable sustainability, cost and engagement.

There is therefore a need for real *cross-domain* teams to coexist, made up for example of experts in: *mechanics*, *electronics*, *software*, *chemistry*, etc. with a support group that supports the different teams and is readily available. It is essential to highlight that we are not talking about creating an organizational "function", but a sort of *loan team* dedicated to the project.

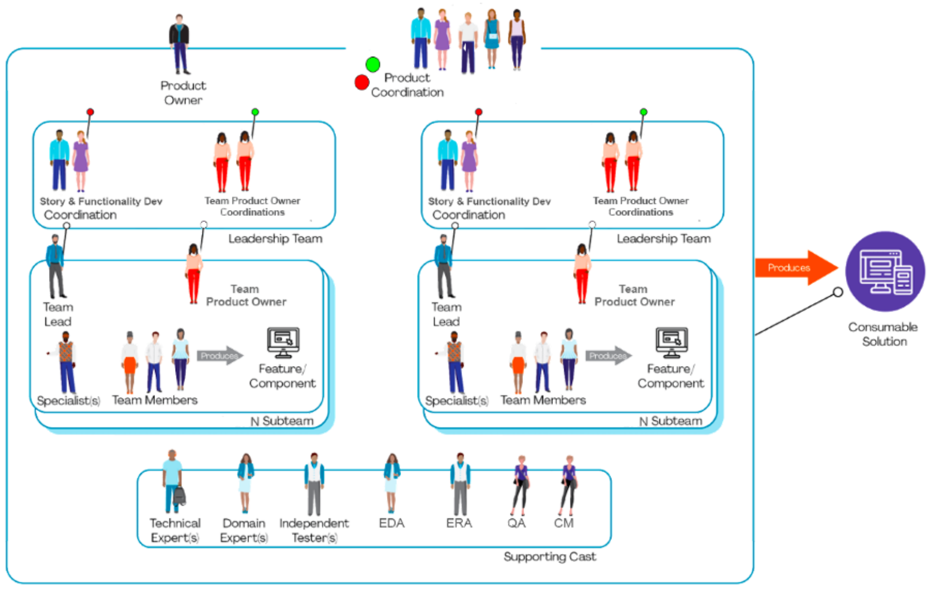


Figure 10 - Cross-domain teams and Loan team (by PMI Disciplined Agile)

The different cross-domain teams self-organized, therefore with a strongly bottom-up approach, with the support of a Product Owner (or *Engineer Owner*) who helps them to stay focused on the general objectives and settle as much as possible the dependencies between the specific activities on which they have engaged.

There are therefore two reference scenarios:

- A *product developed by a single team made up of all the necessary specialists*. This scenario, while simplifying the various problems of dependency and coordination, is the least realistic given the complexities faced by system engineering and the need to have reasonable times in the development of new products.
- A *product developed by several teams in parallel*. In this case, the most common one, it is essential to develop an appropriate *coordination leadership* to allow the different teams to make their activities more efficient thanks to effective and streamlined collaboration.

- In both scenarios, the aforementioned *loan support team* on specialized topics that are not necessary in everyday life remains valid.

4.3 Quality and metrics

Quality is a fundamental element that must be placed at the center of the product vision, even if its declination depends a lot on the specific domain, the related standards and the related regulations.

Despite this, it is possible to define an abstract model (inspired by the *Test Pyramid*[2]) that we will call Quality *Pyramid* here and which can be represented as follows:

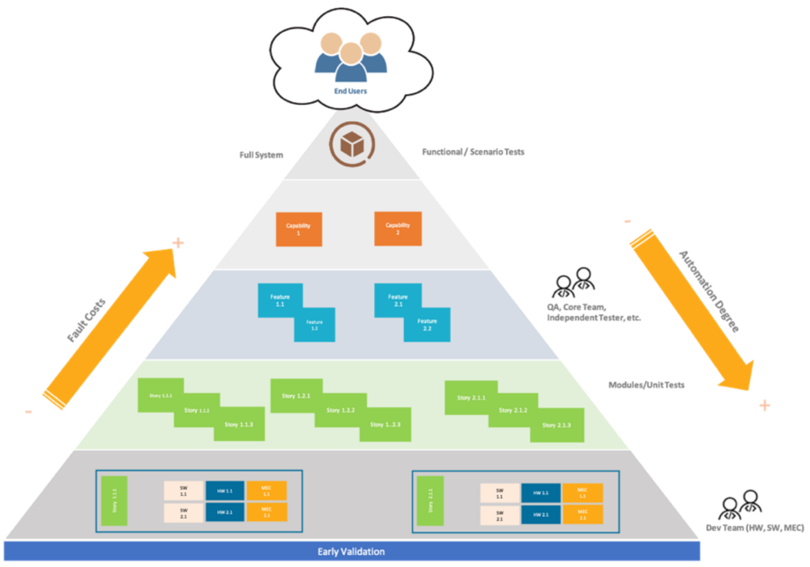


Figure 11 - Quality Pyramid

The fundamental aspect is that quality passes through a *series of tests and validations at several levels*, trying to take automation to the extreme in the lower levels of the pyramid where, normally, the different components of the product are made and where developers use the practices and tools most suitable for the related domain.

It is evident that the more you can optimize testing at the lower levels, the fewer problems should arise when moving towards the tip of the pyramid, where, generally, the focus is *end-to-end* (towards the end user) and any problems have a much higher resolution cost, sometimes not absorbable.

[2] https://martinfowler.com/articles/practical-test-pyramid.html

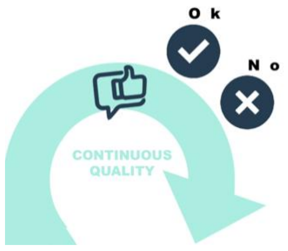


Figure 12 - Continuous Quality

In a very abstract way, AgileEngineering considers *Quality* as an enabling "gate" (*Quality Gate*) that is based on an approach oriented towards *Continuous Quality*: there is no "no" in quality, i.e. either the relative expectations are met, or the component/module under analysis is not ready for integration with the others or for delivery to the customer.

The approach to Continuous Quality indicates the attention to continuous improvement in similar practices and tools, highlighting once again how contingent and pragmatic evolution is the basis of good management of everything related to a complex product, with respect to which knowledge is refined during its creation.

However, improvement cannot be left (only) to intuition, but must be based on objective data, numbers, which can highlight *facts* and not give rise to *opinions* that are always affected by one's own interpretative capacity and declination.

Precisely for this reason, AgileEngineering defines a series of basic metrics that help to objectively assess the development and reaction capacity of work teams:

Figure 13 - Metrics

4.4 Engineering Metrics

The Quality Pyramid model defines the levels of product validation, emphasizing Continuous Quality.

The related metrics are:

- *Cycle Time (CT):* The average time taken to complete a single development iteration.
- *Lead Time (LT):* The total time from the start of development to the delivery of a feature.
- Mean Time to Recovery (MTTR): The average time taken to resolve a critical issue.
- *Deployment Frequency (DF):* The number of releases made in a given period of time, useful for assessing the team's ability to iterate.

- *Defect Escape Rate (DER):* percentage of defects detected after release compared to those detected during development.
- Code Churn Rate (CCR): The amount of code that has been changed or rewritten within a short period of time after the first commit, which is an indicator of the effectiveness of technical planning.
- *Test Coverage (TC):* percentage of code covered by automated testing to ensure greater software reliability.
- *Team Velocity (TV):* The amount of work completed by the team in an iteration, often measured in story points or completed tasks.
- *Failure Recovery Rate (FRR):* The average time it takes to restore the system in the event of critical failures.

These metrics help maintain effective control over the development team's performance and product quality.

5. The Workout phase

As repeatedly emphasized, the primary output developed by a system engineering action is composed of different elements pertaining to different disciplines, very often also strongly different from each other.

For this reason, delivery (understood as delivery of the product and what is necessary for its correct use by customers) is decidedly articulated and requires the use of specific **validation** and **preparation techniques**, as well as **training** and **support**.

The set of these actions, which bring the product "into the hands of the customer" is defined in AgileEngineering as the **Workout phase**.

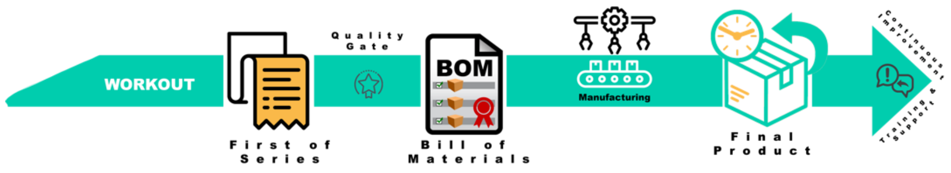


Figure 14 – AgileEngineering - Workout phase

In the Workout phase, some fundamental elements are identified, which, although they must be contextualized, are a reference for a delivery management structure:

- *First of Series:* the industrial "prototype" of the product. In practice, there is an industrialized product representative of what will subsequently be massively produced (or in any case of what will be delivered to the customer) on which all pre-production evaluations can be carried out.
- *Quality Gate*: a set of actions, tests, validations and verifications carried out on the first series that leads to the development of all the final elements and related specifications for manufacturing.
- *Bill of Materials*: These are the final specifications, and what is needed, for manufacturing.
- *Manufacturing*: is the process of industrial production of the product.
- *Final Product*: is the final industrial product.
- *Training and Support*: the training and support actions for the use of the product by customers.
- *Product Continuous Improvements*: support the final product with continuous improvements and updates

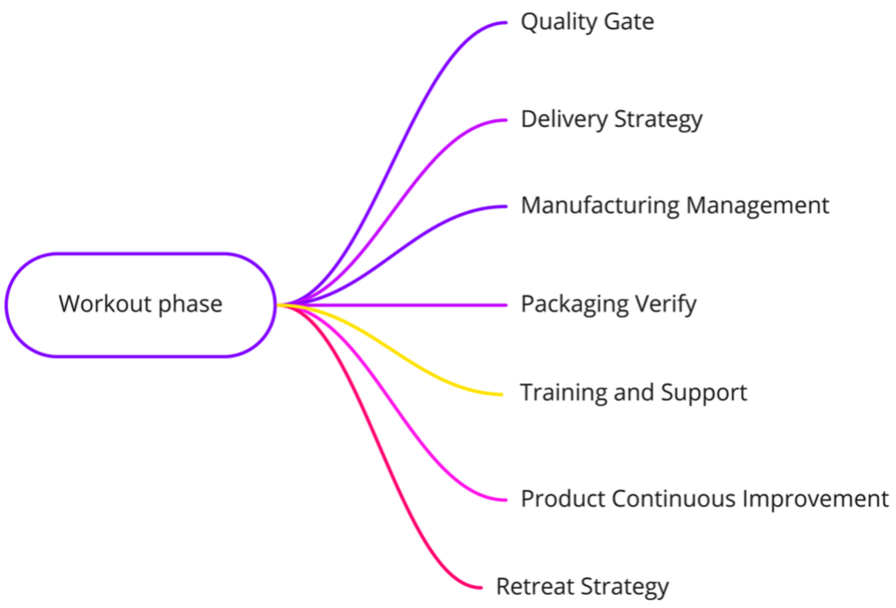


Figure 15 - Workout goals

As you can see, the Workout phase is in itself extremely articulated and requires the support of many heterogeneous professionals.

In particular, it should be emphasized that the manufacturing part almost certainly requires the engagement of an external partner for the production of the product, which shifts the focus to aspects of management and relationship with external suppliers. Normally, the *first series* is used to move from the rapid prototyping tools used in the Engineering phase to industrial, custom or market elements, to understand their impacts and evaluate their costs. Here, the choice of a trusted supplier is essential to cope with the inevitable problems typical of the transition and work in an agile perspective of constant feedback.

A particularly delicate aspect is that of the *physical realization of the product (*supported in particular by the practices attached to the *Manufacturing Management* goal). In general, the final product will certainly be based on a mix of *Market Components and Custom Components,* identifying most components on the market that can be reused through their adaptation (to reduce production costs), while the remaining part will be made specifically according to the needs of the product.

In any case, the manufacturing aspect becomes a central element of the Workout phase and as such must be adequately managed by creating a bridge between the design team and the production team (whether internal or external) in order to solve the initial inevitable problems of implementation.
This, in particular, will lead to the *First Series* (which could also be followed by further prototypes if necessary) which, in addition to having the purpose of evaluating the functional effectiveness of the

product, will help to set up the correct production processes and review any construction hypotheses for overall efficiency.
All this finds a home in the Quality Gate goal that will have to lead to the precise definition of all validated product and process aspects, creating the *Bill of Materials* for the start of production.

5.1 Manufacturing Production

Whether production is carried out by a third-party manufacturer or by a different team/department of the same company, it must be integrated into one's governance activities, so as not to incur unpleasant surprises that undermine what has been done so far.

The related operational management is typically entrusted to a (chief) *Project Manager* who has the task of establishing a profitable relationship of *collaboration* and *trust* between the different production realities, starting from the creation of the *First Series*, in which, although we are still working on a prototype basis, we begin to share the know-how necessary for the subsequent industrial realization (which can also be massive) of the product.
Normally, *the Manufacturing* action follows a more *Lean-oriented* logic (Lean manufacturing specifically) focusing on the efficiency of production and, therefore, on the rationalization of related costs (marginal cost).

Once the product is operational, i.e. installed and in use at the customer's site, support for the same becomes bidirectional, enabling the logic of *Product Continuous Improvement*:

- *Predictive update*: it takes place by the manufacturer's actions and can also be carried out remotely (think of firmware updates).
- *Corrective update*: this takes place following the report of a malfunction or an anomaly reported by the customer.

The objective of the approach presented is to ensure that any updates are focused, almost entirely, on the most *ductile* elements of the system, avoiding acting on the parts that involve high intervention costs (think for example of mechanics), unless a total revision of the product.

A critical action that should not be underestimated is to adequately structure, and in time, the *Training and Direct Support* plan for customers for the use of the product. This is crucial to increase the level of perceived quality, which transforms a technically quality product into a successful product appreciated by customers.

5.2 Metrics for the Workout phase

The metrics of the Workout phase are:

- *First Pass Yield (FPY):* percentage of products that pass the first stage of production without the need for rework.
- *Defect Density (DD):* number of defects detected per unit of product or production batch.
- *Manufacturing Cycle Time (MCT):* The total time it takes to complete the production of a unit, from start to finish.

- *Supplier Quality Index (SQI):* Quality of components supplied by industrial partners, measured by compliance with requirements.
- *Production Downtime (PD):* Total downtime of production lines due to breakdowns, maintenance, or technical issues.
- Customer Return Rate (CRR): percentage of products returned by customers due to defects or malfunctions.
- *On-Time Delivery (OTD):* percentage of products delivered to customers by the set date.
- *Cost of Poor Quality (CoPQ):* Total cost from rework, scrap, and returns.
- *Satisfaction Index (SI):* Evaluation of customer satisfaction based on feedback and after-sales surveys.
- *Mean Time Between Failures (MTBF):* average time between failures for products delivered.

6. Conclusions

As mentioned in the introduction, this paper has the task of telling the focal aspects of Agile applied to the world of System Engineering, highlighting the key aspects in relation to the different related areas.

If you are interested in the in-depth study and concrete application, you can contact us at the e-mail address info@agileconstellation.info or through our social channels that you find on the official website.