IoT meets Agile

Rev. 15.0 – March 1, 2023

AgileIoT

an AgileConstellation Star

Summary

Internet of Things

What is the Internet of Things

The term **Internet of Things (IoT)** means the capability, for two or more objects or real places to be connected, using the existing network infrastructure. This possibility makes them able to communicate and adapt their behavior on the basis of events and the evolution of the context.

In the IoT universe, there are two main elements: **Smart Things** ("T", smart devices) connected and managed through the **Cloud** ("I", Internet) also used to compute and analyze the Information. With the term Cloud we mean both meta-architectural approaches: *public* Cloud and *private* Cloud.

Although the idea is recent, the idea of getting devices connected to a network to have a constant flow of data could be dated back to 1982, when the ***Carniege Mellon University*** modified an automatic soda distributor to connect it to the network and finally get the amount of cans remained, in real time.

The first formalization of the **Internet of Things** is back to 1991, when *Mark Weiser* wrote an article titled "The Computer for the 21st Century" where he images a future with "*hardware and software, connected with cables, radio waves and infrared, will be so ubiquitous nobody will be aware of them anymore"*

It's a 30-year-old idea, now reality thanks to the evolution of the technology, the Internet broadband, the Cloud and the penetration of the IT in every part of our lives. *Gartner* estimates that, 26 billions of devices will be connected within 2020; *Pew Research Center* highlights (survey 2014) 83% of IT experts agrees on the fact that, within 2025 the Internet perception will completely be bound to the IoT thanks to the explosion of various market sectors like *domotic, robotic, wearables and automobiles*.

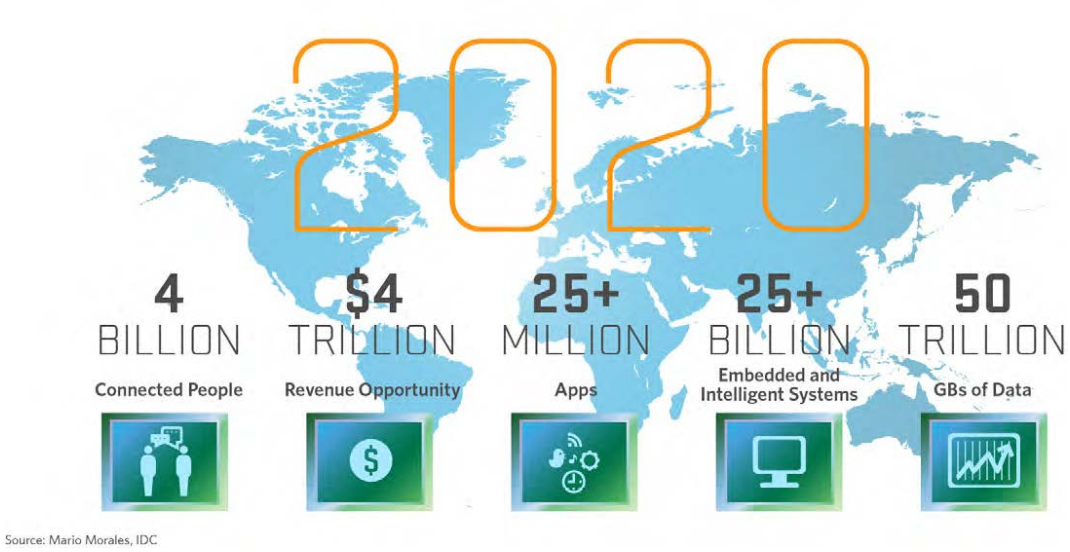

Figura 1 - IoT in the 2020, IDC Research

Main application fields

Smart Car, *Smart Home* and *Smart City* are main areas of growth for the IoT, highlighting the vocational multi-disciplinary of the IoT itself.
Let's think about the South Corea case: in *Songdo*, 65km away from Seoul, it was born the first *Smart City* in the world, a completely connected city, continuously gathering data regarding tens of areas, constantly computing them, without the need of human intervention.

The integrated city project in *Santander*, Spain, is interesting: about the 10% of the population use a Mobile App; the App gathers data from different sensors distributed along the city, providing real-time information regarding traffic, pollution, parking lots and other events.

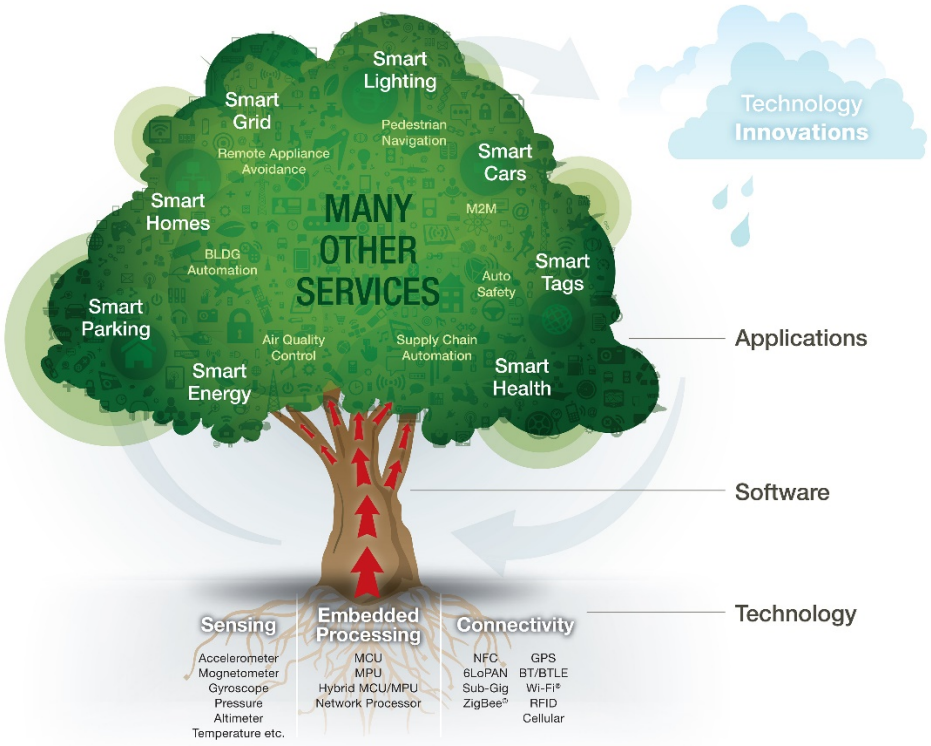

Figura 2 - Fields of Application

New challenges: privacy, security and energy saving

Privacy is one of the big problems related to IoT: more devices connected means more personal data traveling over the network. This requires to focus immediately on the implications on privacy that IoT solutions can bring.

The **data security** related to the equipment control and management is strictly linked to the privacy as it affects and impact the *real world*: an attack to in-car equipments, for example, could lead to real disasters.

Although these two topics are already analyzed and challenged, the **energy saving** topic is, on the other hand, less obvious since we tend to take for granted that the devices don't have energy problems.

What happens, though, if the device is, for example, on a marine platform where energy is limited and optimization goes to few watt-per-hour level?

It is clear that energy can become a driver for the design and development of the entire solution IoT.

IoT Project Governance

The multi-disciplinary scope

An IoT solution belongs to **different technological domains**, each of which, with their own rules and specifications. Thus, the project (or projects) should be, regularly, checked against different aspects: the *Hardware* – with strong and predictable processes – the Software, where the complexity and the change is always present, *the Cloud and Big Data* with its own rules and risks.

Today, typically, hybrid *software* and *hardware* solutions are developed in a parallel way with determined synchronization meetings useful to start the integration process and, subsequently, to the test process.

This is a typical *Big Bang Integration* approach, with different critical issues:

- *Development efficiency is heavily penalized as different Teams don't constantly communicate with each other, without sharing their knowledge*
- *Issues are discovered in the late solution development phase, with high fixing problems, mostly due to the hardware fixing.*
- *As of the need of shortening / respect the Time-to-market, the final product, often, has a compromised solution to problems (based on software workaround, wherever components cannot perfectly integrate with each other), thus lowering the global product quality.*

The following sections will detail the primary aspects of the different domains.

The hardware aspect

The development of **Hardware** typically follows an approach waterfall-like, with a linear process optimized over the years and quite rigid compared to changes during construction. Roles and skills are better defined, including, at least:

- *the hardware designer,* which generally has the task of creating the schematic, establishing, for example, that the processor is connected to the RAM and FLASH, that there needs to be a level translator, a resistor here ... a capacitor there, etc.
- *the "sorter",* which transforms the circuit diagram (for which there are no "physical" positions on a Board) in corresponding physical schema, in which, on the contrary, the components are placed on the actual circuit by placing the connectors matching the holes of the mechanical designer. His task includes the definition of "slopes" that connect the components, etc. The *Bill of* Materials is created starting from actual electronic components;

- the *mechanical designer*, defines *mechanical* parts of the devices. By example, he/she defines the box containing the device, holes positions for connectors, sizes etc.
- *The software developer* (*driver e Board Support Package*), is in charge of developing drivers for devices. In the case a real Operating System is used on the developed device, he/she is also in charge of generating the customized OS Image. This is mostly true for systems based on *Windows Embedded Compact* and *Linux*. We must emphasize that, typically, the developer has profound hardware and firmware knowledge as it also plays the role of test driver of the hardware product. With a series of tests, he validates the proper connection of the wires and the work of the sorter and/or designer to the extent that they have done something wrong during the wiring (for example, placing a pin of the processor I/O memory power pin);
- *The software developer* for the application part of the firmware, making the firmware interact with devices (through software drivers), analyzing signals, sending/receiving data, etc.

The software aspect

Today, the development process of the **Software Component** is typically governed by Values and Agile principles[1]. Teams use a mix of practices to maximize the solution value. The approach is *iterative* and *incremental*, in order to quickly get feedback from an even more complete versions of the product; thus, activities are re-align on those bases, Team maturity and the progress of project activities.

Another aspect to highlight is the cross-functional vocation of Agile Team members: developers can work on the entire project, leaving it free from specific roles or persons. Further roles can support developers: one can be the *Scrum Master*, in the *Scrum* process; He / She plays the role of facilitator and culture-spreading among the whole company context.

Typical Agile roles are:

- the *Product Owner*, who is responsible for enhancing what has been achieved by the Team according to the objectives of value required by the customer;
- the *Scrum Master*, who is responsible for disseminating the Agile / Lean Culture within the specific team and the general company environment;

the *developer*, who has the task of implementing the project effectively

[1] Agile Manifesto, agilemanifesto.org

The cloud aspect

Together with the classic Hardware/Software dualism, an IoT project adds the specificity of the **Cloud Component**. It's not as simple as "*publishing own services and database on a remote VM*", but a totally new development paradigm, able to achieve and make own solutions available as commodities: Do I need it? Then I'll pay for that, otherwise I turn them off and I don't.

In order to achieve this objective, it is fundamental to think "for the Cloud" since the beginning: an example is the one related to the development of "elastic" services. They scale horizontally which means computing power is increased by increasing the number of machines and then balancing the computing load among all of them. This architectural approach is totally opposite to the typical "vertical scaling" wherever the increased computing power is obtained just by upgrading the (single) machine hardware (with consequent physical limits).

The same approach is the one related to the *Big Data* that needs to be managed with extreme efficiency and modern solutions (think of the storage in a NoSQL Repository), in order to make data readily available to be (eventually) transformed before consuming them.

The Cloud component can be approached using Agile methodologies but platform aspects (or, System Engineering aspects) need to be considered; this is the reason behind we often refer to *DevOps* with both meaning of software development methodology and a role with competencies between Developer and System Engineer.

The governance complexity

As from previous descriptions, the **governance complexity** of an IoT project is not an easy aspect: it needs proper tools in order to harmonize all activities and the workflow of different roles involved.

In order to do so, we can't keep *wall*s between development processes of main components; it's needed to create a uniform process based on the daily sharing of the project status and the quick integration of all components with the final goal of releasing a working solution.

As from previous considerations, it is evident it's not realistic to approach the governance and the development of an IoT solution by "lending" methodologies from software, hardware or Cloud, but it is needed to approach the problem in a holistic way.

Using such an approach, the software development and the hardware development are going to merge into a completely new disruptive context with specific rules and mechanisms supporting the specificity of the IoT. Think of the terminology, suddenly overlooked, but fundamental for the relation with stakeholders and the customer itself.

AgileIoT

AgileIoT starts from those considerations and is created on the *Philosophy*, *Principles* and *Values* of the **AgileConstellation Manifesto**[2]. It proposes a consistent methodology for the creation of Values in the Internet of Things world.

It helps to approach in an explicit manner the governance of the creation of a new IoT solutions, building it on the experience gained in the various disciplines involved.

The aim of AgileIoT is not trying to sketch adaptations from others discipline because they may be inadequate. This because typically the idea is to translate specific aspects of a single domain in a context in which the multidisciplinary is a carrier.

Rather, AgileIoT it suggests adopting a new approach that looks at the Internet of Things in a holistic manner, suggesting an approach in which the various elements are considered part of a single integrated process.

[2] www.agileconstellation.org

AgileIoT: an AgileConstellation Star

AgileIoT is constitutionally based on the Agile and Lean mindset, boasting the declination developed within the AgileConstellation project which looks at domains other than software and digital in general.

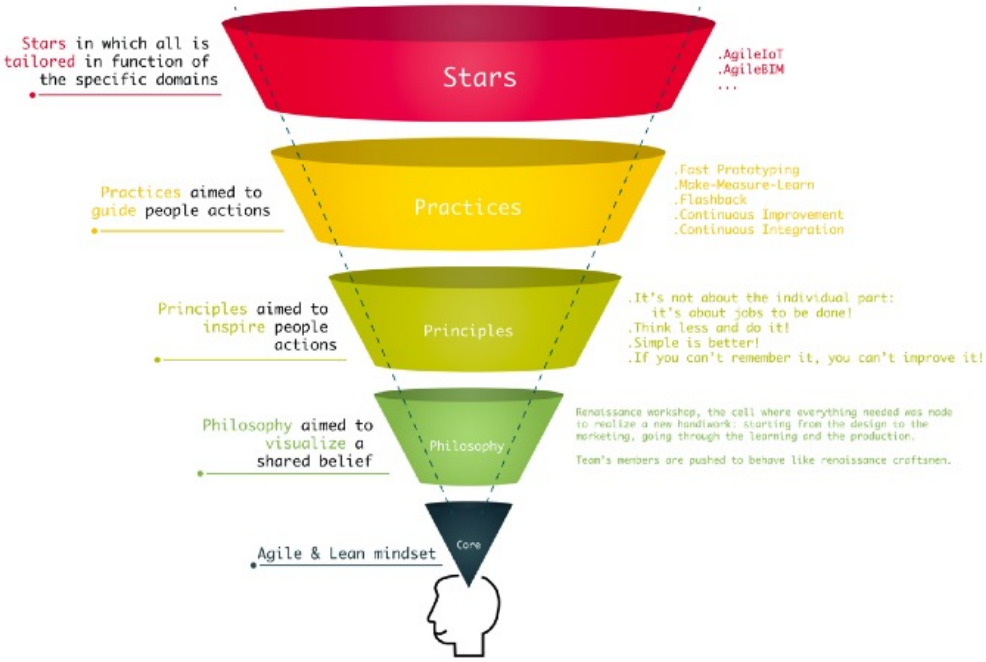

Figura 3 - AgileConstellation funnel

AgileIoT inherits the mindset, philosophy, principles (core) and practices (core) of AgileConstellation, declining and extending, in particular, these last two aspects in the specific domain of the IoT. So, we have:

- **Philosophy**, inspired by the **renaissance workshop,** where the *cell* does everything needed was made to realize a new handiwork: starting from the design to the marketing, going through the learning and the production.
- **Principles (core):**
  - 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!

- **Practices (core):**
  - *Fast Prototyping*, validate the sustainability of the solution
  - *Make-Measure-Learn*, quickly experiment with different hypotheses and assumptions
  - *Flashback, rapid alignment where is the observer to be responsible of it*
  - *Continuous Improvement*, constantly improve every aspect
  - *Continuous Integration*, constantly integrate the different souls of the solution

AgileIoT Fast Prototyping

According with the modular approach of the AgileConstellation Manifesto, the *AgileIoT Star*[3] added 6 new bubbles domains to the Fast Prototyping practices to validate the solution sustainability:

- **Security**: focused on the verification of the security aspects, which affect the realization of the solution;
- **Energy**: focused on the energetic-based aspects as a function of the needs of the operational continuity of smart devices;
- **Hardware**: focused on the validation of the hardware through one or more *Evaluation Kits* (EVK). The EVK is evaluated through CPU/MCU, and then with prototyping the other components, like RAM, USB, and so on;
- **Code**: focused on the prototyping of the firmware of the devices and the services made for acquiring the main data/events. In this phase, useful frameworks and productive IDEs can really cut the development time;
- **DataFlow**: focused on the aspects related to the gathering, cleaning-up and managing of the *Raw Data* that comes from the devices, with implementing the data transfer and serialization with the right protocols and the right format. It is very important to estimate the data volumes, the ways for analyzing them, in order to get the best approach possible, following the *Polyglot Persistence* pattern, which is fundamental for a big-data scenario.
- **Cloud**: focused on the Cloud aspects of the solution, as a data/event management platform.

[3] Iot.agileconstellation.org

Figure 1 - The new 6 bubbles added by the AgileIoT domain

The *Product Backlog* can be designed deeply now, due to the prototyping phase outcomes. The *Product Backlog* will be created through the **Product Backlog Planning** (2), which is managed by the Product Owner (or by the Chief Product Owner, if there is more than one team). In this ceremony, the (Chief) PO, key stakeholders and the team will define the **Solution Big Picture** of the solution, together with the epics and their priority.

It is important to note that this phase also impacts on the company organization that must adequately support the development of the solution.

Frameworks

You can choice to implement AgileIoT through two different frameworks: **Eclipse Duttile** and **Fiotto.** The related descriptions are available on the official website www.agileconstellation.info and https://projects.eclipse.org/projects/iot.duttile.