# WHAT ARE THE PROBLEMS

In the recent years, the number of projects using machine learning has increased exponentially, as has the amount companies are investing in this technology.

This growth carries out with it a bunch of problems since the first models until now, such as the incorporation of machine learning models in production. Until 2022, up to Deborah Leff, former CTO for data science and AI at IBM, 87% of the data science projects never make it into production ( <https://venturebeat.com/ai/why-do-87-of-data-science-projects-never-make-it-into-production/> ) and among the 90% of companies that have made some investment in AI, fewer than 2 out of 5 report business gains from AI, improving this number to 3 out of 5 when we include companies that have made significant investments in AI (<https://sloanreview.mit.edu/projects/winning-with-ai/>) .

Those are the reasons why only a small percentage of the ML projects manage to reach production, being essential to find out what are the problems which come across since such an extraordinary inversion from the companies should never be wasted.

# STATE OF ART

In order to comprehend the current situation of the software industry, we must understand the previous steps of the software path.

# First software development: structured programming

The first software developments are about the 40s and 50s, when the first computer machines came out. Such was the rise of these new technologies that software´s developers had to give an extra twist to make them useful. So, they started developing software using *Structured Programming.*

Structured Programming is a trend that was born to make the life of the developers easier. It was not until 1966, when Böhm and Jacopini launched the structured program theorem, which says that any program could be wrote using just 3 instructions, when its consolidation began (https://www.edix.com/es/instituto/programacion-estructurada/).

In 1968, Edsger Dijstra published a well-known article, Go To Statement Considered Harmful (<https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf>), claiming the use of this new concept and the banish of the Goto sentence. Ending the consolidation of the structured programming.

As it has been said before, it is based in the 3 basic structures: sequence, selection or conditional and iteration. Those three basics made easier to understand the codes, with a more clear structure, better to optimize the testing and debugging and the maintenance expenses were reduced (<https://www.edix.com/es/instituto/programacion-estructurada/>).

# Waterfall methodology

At the beginning of the 70s the projects were complex enough having a long list of requirements, also needing a lot of documentation, alongside a proper design´s planning of the solution. Aiming to solve these new challenges, Waterfall methodology was born.

Based on a correct requirement´ definition, due they must be unchanged during all the process, it is a linear process of project management. Each step on the procedure cannot begin unless the previous phase is finished, and once finished, it is terminal, since Waterfall management does not allow you to return to a previous phase. (<https://www.lucidchart.com/blog/waterfall-project-management-methodology>)

Normally, waterfall methodology varies somewhat depending on the source, but they generally include:

* Requirements gathering and documentation.
* System design
* Implementation
* Testing
* Delivery/deployment
* Maintenance

# Agile methodologies

Due to the slowness and the delays caused by the traditional way of working, which used waterfall methodologies, the software industry ideated the agile methologies. The previous projects were based in fixed requirements, not able to change them once the process started, and the big efforts that were meant to suppose if a change was made drove to not-as-high-as expected quality projects.

In 2001, the Agile Manifest was created by the principal CEOs of the software industry in Utah (<https://agilemanifesto.org>). The Agile manifest was based in four keys:

* Iterations and individuals above processes and tools
* Functional products above exhaustive documentations.
* Partnership above deal negotiations.
* Change with the problem above and strict plan.

Diagrama

Descripción generada automáticamente

The main advantages of the agile methodologies that will make it be the base of a big amount of DevOps processes are:

* Ease and reduction of the process overload: being able to adapt at every stage of the development process makes easier to achieve what is expected in every milestone, with no extra effort at the end of it.
* Better quality of the product: at every iteration at least a minimum functionality, but at the end, an improvement from the previous ones. Moreover, the customer is at every moment involved in the development process, being able to ask for changes depending on the market realities.
* Improvement in the foresees through a better management of the risk: The agile methodologies were designed to respond the possible changes and problems that merge during the life cycle of the project.
* The customer is involved at every stage of the development so they could give feedback at every point, making a stronger relationship between developers and customer and an improvement in the satisfaction.

# Foundations of DevOps

The previous both methodologies have the same objectives, deliver production-ready software products. The problems came with the gap between the developer teams and the operation teams, reaching even point of isolation between them, competing against each other’s. Both teams had different KPIs, different squad leaders, objetives. So, a concept called DevOps emerged in the years 2007-2009 (<https://devops.com/the-origins-of-devops-whats-in-a-name/>) trying to solve all the differences. DevOps is more than a pure methodology and rather represents a paradigm addressing social and technical issues in organizations engaged in software development (<https://ieeexplore.ieee.org/document/10081336/references#references>).

During the life cycle of the product, DevOps uses agile methodologies by definition. Also is based in the automation of processes, it is managed through the continuous integration, continuous delivery and continuous deployment (CI/CD). It allows to deliver fast and reliable solutions too. Moreover, it is designed to facilitate continuous testing and quality assurance, and thanks to other applications and tools, monitoring, logging and feedback loops.

The most important assets of the DevOps methodology are (<https://ieeexplore.ieee.org/document/10081336/references#references>):

* Workflows and repositories, also called source code management: Tools such as GitHub, BitBucket or GitLab.
* Monitoring and Logging (e.g., Prometheis, Logstash).
* Build process (e.g., Maven).
* Continuos integration (e.g., Jenkins, GitLab CI)
* Deployment automation (e.g., Kubernetes, Docker, Ansible)

Nowdays, the main Cloud providers have already-built solutions for DevOps tooling, reducing the time needed to start giving value to a project.

# MLOps

## DEFINITION AND CHALLENGES

Therefore, summing up everything said before and at gross mode, MLOps is the application of the DevOps methodologies alongside the use of machine learning in a production environment.

In a more technical way, MLOps is the standardization and streamlining of machine learning life cycle management. (<https://www.oreilly.com/library/view/introducing-mlops/9781492083283/ch01.html>). For most organizations and companies, the process of the machine learning is relatively new and the number of projects in productions is not big enough yet to accommodate them to an automation environment, which is where it becomes more critical.

For those that have done it, the main challenges they face during the life cycle of the data are:

* Changing environment: data and the business needs shift constantly, is required to be sure that the model in productions aligns with the expectations and if it satisfies the original problem and goals.
* Misleading communication: despite of being in the same company or share the same goals, not everyone shares the same tools, procedures or skills.

With the gathered facts explained before, it was easy to come out with a solution truly related with the DevOps models, MLOps, which is quite similar but not identical since the software development used in DevOps is almost static, almost because some companies can iterate changes during the software life cycle, against the continuous changing data of Machine Learning. Machine learning models are always learning and responding to new environment it is in, also including both data and code.

So, the final model of the MLOps was born from the development and operations of the DevOps and the data engineering.

Diagrama, Diagrama de Venn

Descripción generada automáticamente

The greatest similarity between MLOps and DevOps methodologies lies in the concepts of Continuous Integration and Continuous Delivery, which allow software deliveries to occur with a high frequency, with reliable results. In most companies that use Machine Learning, they develop the different models and put them into production manually, without incurring MLOps. This also brings with it the problem that machine learning has no return of investment until it can be used. Therefore, all efforts must focus on the steps that follow the development of the model itself, specifically on the interfaces between the ML response and the infrastructure where it will be implemented.

(<https://ieeexplore.ieee.org/abstract/document/9792270>).

Diagrama

Descripción generada automáticamente

## PRINCIPLES AND ROLES OF THE MACHINE LEARNING

TIRAR CON TODO LO QUE TIENE ESTA PÁGINA <https://ieeexplore.ieee.org/document/10081336/references#references>

As with any methodology, MLOps is also governed by a set of principles. These are not fixed, but any process that aims to end up with a machine learning model in production will have to be governed by certain fundamentals such as the following:

* CI/CD automation: this process carries out the building, testing, shipping and deployment of software through continuous integration, continuous delivery and continuous deployment. As previously mentioned, this allows for early identification of process failures and continuous product value and increased productivity.
* Reproductivity: the ability to replicate the same processes obtaining the same results in different environments for the same inputs.
* Workflow orchestration: coordination of the different machine learning tasks in a pipeline according to acyclic graph directives.
* Versioning: different versions of data, models and code allow for greater control over regulatory compliance and audits.
* Continuous monitoring: by continuously monitoring the data, the model, the code, or the infrastructure itself, errors can be detected and/or changes can be made according to previous results in a more efficient and less costly way.
* Feedback loops: this type of loop is necessary to manage the relationship between the different stages of quality assessment and the engineering or development processes.
* Continuous re-training of the different models: as each re-training is based on new data, it is a consequence of the previous processes of continuous monitoring, feedback loops and workflow orchestration. One aspect to be managed is the expenditure to be allocated to this stage, as a higher frequency of re-training entails a much higher cost.

Continuing with the roles needed for an MLOps process, these are based on agile methodologies. As in any software development process, a good definition of the participants in the process is fundamental to design, manage, automate and operate any machine learning system.

* Business stakeholder: in charge of defining the business objectives is also in charge of taking care of the communication between the team and the client.
* Solutions architect: defines the architecture and technologies that will be used.
* Data scientist: converts business problems and requirements into machine learning problems. At a technical level, he/she oversees model engineering.
* Data engineer: designs and implements data pipelines and engineering features.
* Software engineer: converts the ML problem given by the data scientist into a well-engineered product by applying software framework design and best practices.
* DevOps Engineer: strives to link development and management processes under CI/CD methodology, ML workflow orchestration, model deployment to production and monitoring.
* ML Engineer/MLOps Engineer: combination of all the above roles, cross-functionally. Manages the ML infrastructure and ML automation flows.

Diagrama

Descripción generada automáticamente

AHORA METERÍA CICLO DE VIDA DE LOS DATOS Y GRADOS DE MADUREZ. TERMINAR CON LOS DISTINTOS FRAMEWORKS.