Formal Vindications SL

1 What do we do?

Formal Vindications is an **innovative startup of high scientific value** within the **software industry**, in direct collaboration with the University of Barcelona.

- What are we developing? In short: **Infallible software**.
- How is this possible? Using **logical-mathematical techniques**.
- Who can perform that kind of software? Only **logicians** with a very strong background in mathematics and proof theory.

2 The problem we are solving:

Humans make mistakes. In particular, in this case, **programmers make errors**. According to the book *Code Complete* [1], The average number of errors (or bugs) by line of code delivered is of the scale that the following table shows:

Code type	**Average number of errors per 1000 lines of code**
Traditional	200
Industry	10-15
Microsoft Applications	0.5
Shuttle	0.1

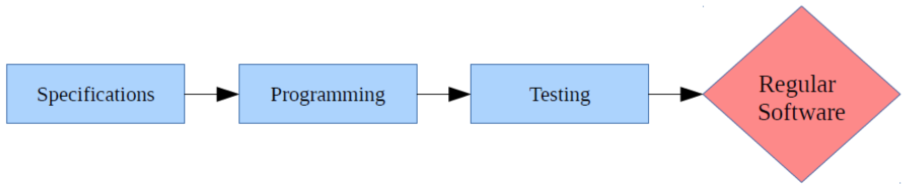
Software is present in all areas of our lives, from an alarm clock, telephone, car, road traffic control, trains, planes, banking control, nuclear power, etc. And the expansion and complexity of these applications continues to advance exponentially, as well as our dependence of them.

[1]Steve McConnell (2004). *Code Complete*. Microsoft Press, 2nd Ed.

Achieving the infallibility of the most critical systems for society is an increasingly necessary milestone, especially in a context where machines and their processes are starting to be automated and autonomous (stock-market robots that buy and sell stocks automatically, cars with autonomous driving, factories 4.0, automatic response missiles, etc.). Soon, market regulators, public administrations, will **start legislating** to guarantee the safety of all critical systems.

3 How does it work?

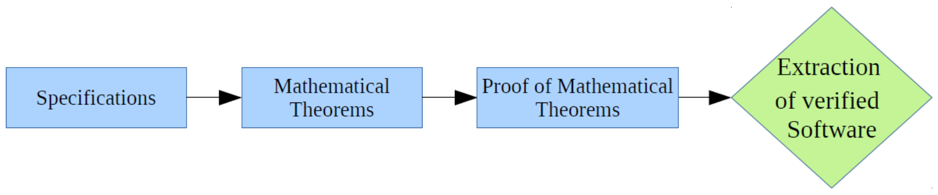
- **Traditional programming paradigm**



The traditional programming paradigm typically involves a standardized process in which the programmers are delivered the technical specifications, and code it using a bottom-up approach until the software edifice is built. In high precision industry, it is standard to use testing protocols that reduce the likelihood of software bugs. Nevertheless, **bugs routinely persist into the final version of the software**, which implies the reality of dangerous consequences and high economic losses.

Furthermore, the similarity between the resulting software and the formal specifications is verified using some testing procedures on each level. However, these tests are based on empirical methods which do not exhaust all possibilities of the software's behavior, and even worse, **are vulnerable to human mistakes**. We can do better.

- **Verified software paradigm**


Specifications
Mathematical Theorems
Proof of Mathematical Theorems
Extraction of verified Software

The **fundamental difference** that makes verified software a new programming paradigm is the fact that the **final code is not produced by humans**.

How is that possible?
The method involves writing down the formal specifications of the software in a precise way, and then they are stated as mathematical theorems. Those mathematical theorems are then proved using a proof assistant, which **automatically extracts the code from the mathematical proofs**. This is the

reason why **humans are not directly writing the code**. The code is a result of **mathematical precision** and infallibility, which guarantees the soundness of the final software, with respect to its formal specification. Furthermore, this method ensures that **there are no bugs present within the deliverable code**.

4 Current developments

Formal verification of software, as a new method of programming, is taking its first baby steps. **Formal Vindications SL is currently leading** the development of this technology into its application to innovative contexts, and striving to set the standards high for security and infallibility.

We are currently **innovating into applications for quantifiable laws**, which involves clarifying legal ontologies and developing precise methods for sound reasoning within a logical-legal language frame. One of our current projects is a specific application of a legal regulation for the transport industry.

Our scientific aim is to develop a formal standard with which we can design laws with desirable mathematical properties. This specifically means that the process of writing regulations and laws includes formal methods used in mathematical reasoning. **The main idea is to give a tool for producing computable laws.**

Actually, laws specifically intended to be implemented with computer software do already exists, and already form an integral part of our society. The advantage of the process we are proposing is that it gives law-makers the **opportunity to check consistency, applicability and precision** of their laws. Also, it implies the possibility of making a program that is able to compute the law itself.