FORMAL VERIFICATION OF SOFTWARE

Formal Vindications SL & Universitat de Barcelona

UNIVERSITAT DE BARCELONA

Conventional Software and its intrinsic problems

Conventional software creation is not error-proof

Code type	Average number of bugs for each 1000 lines of code
Traditional/conventional code	200
Industry	10-15
Microsoft applications	0,5
Shuttle industry	0,1

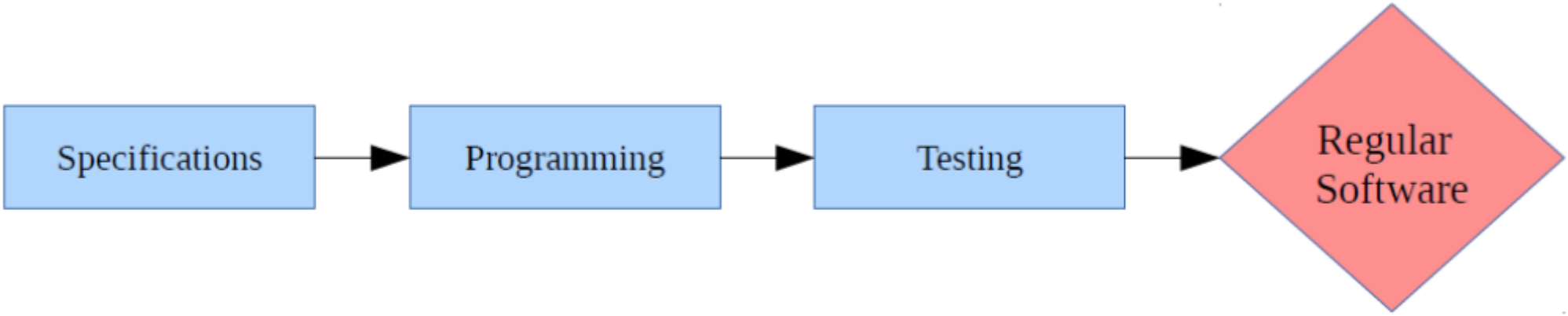
Source: Steve McConnell (2004). *Code Complete*. Microsoft Press, 2nd Ed.

Social and Economic consequences

CONSEQUENCES	LOSSES	INJUSTICE	TRAGEDIES
TYPE OF SOFTWARE	• High value industrial software	• Software involved in legal proceedings	• Software applied to human-related high risk industry
DIRECT CONSEQUENCES	• Expensive applications that fail	• Unfair application of laws	• Physical destruction
EXAMPLES	• Pentium processors	• European software for the control of goods transportation / DNA sequence software in USA	• Patriot anti-missiles system / Aerospacial program Arianne 5

Regular software and human mistakes

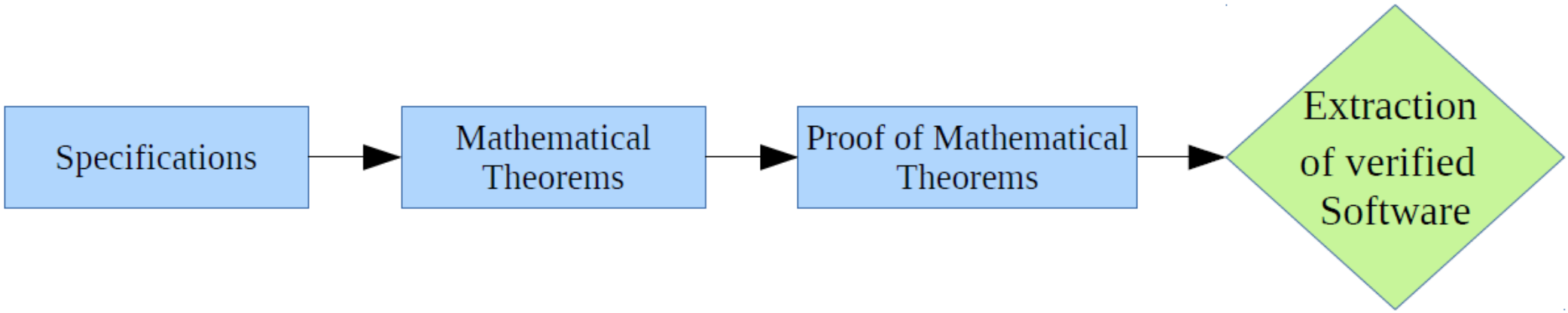
Regular software building process:



No matter how complex and accurate **(ex. DO-178-B protocol)** the process of building regular software is, in the end, **having internal bugs and code mistakes is a human issue**, an always open possibility.

Verified software and mathematical precision

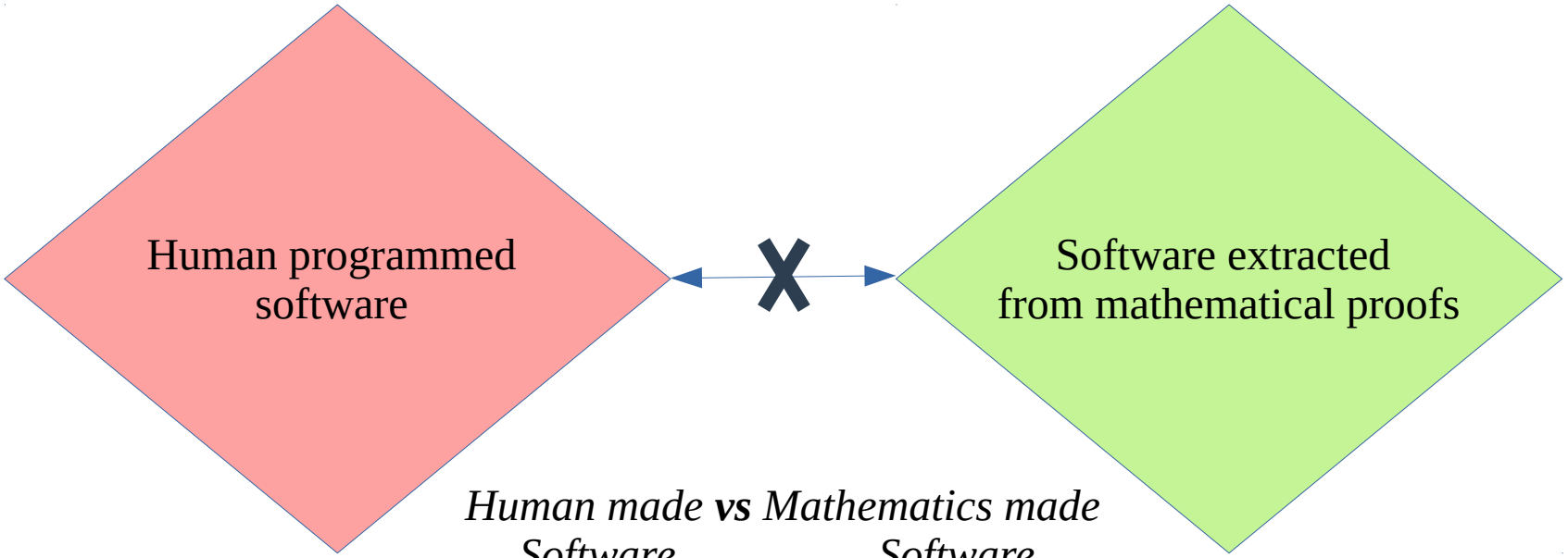
Verified software building process:



Regular Software vs Verified Software

In Formal Verification, the final code is not generated by humans, but by exact mathematical proofs.

Non-Reliable Code ***vs*** *Error-Free Software*



Human made ***vs*** *Mathematics made*
Software *Software*

Who is applying Formal Verification?

Many more private and public companies, developing both civilian and military applications, are starting to use formal verification every year...

In short, Formal Verification…

… is a highly specialized logical-mathematical technique to:

- **Extract Software** directly from **mathematical proofs.**
- **Have a certificate of Zero Error software.**

In order to…

1) Avoid losing large amounts of money

2) Make highly reliable products

3) Avoid potential disasters and dangerous consequences for human life and nature.

In short: Be ready for a software based society