Software Apocalypse

As a Don Quixote we regard an increasing flock of sheep as an army of professionals

19 April 2018

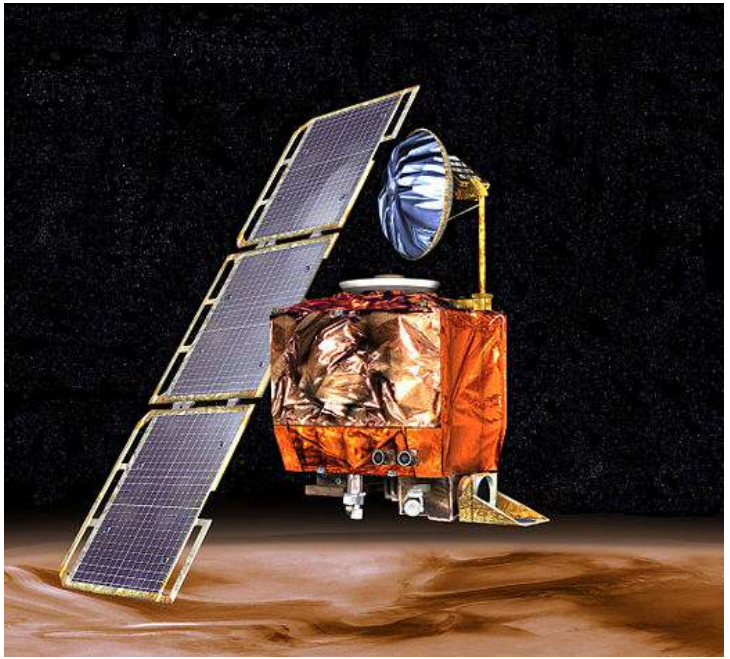
W.T. (Wim) Goes | Directeur Valori Software Improvement
VALORI | Orteliuslaan 1000 | Utrecht | NL
M +31 6 5 0 99 9 66 8
E WIMGOES@VALORI.NL | WWW.VALORI.NL

Contents

- The increasing risk of software failure
- What measures do we have in place?
- The road ahead
- Questions / Discussion

The increasing risk of software failure
A first example – The Mars Climate Orbiter

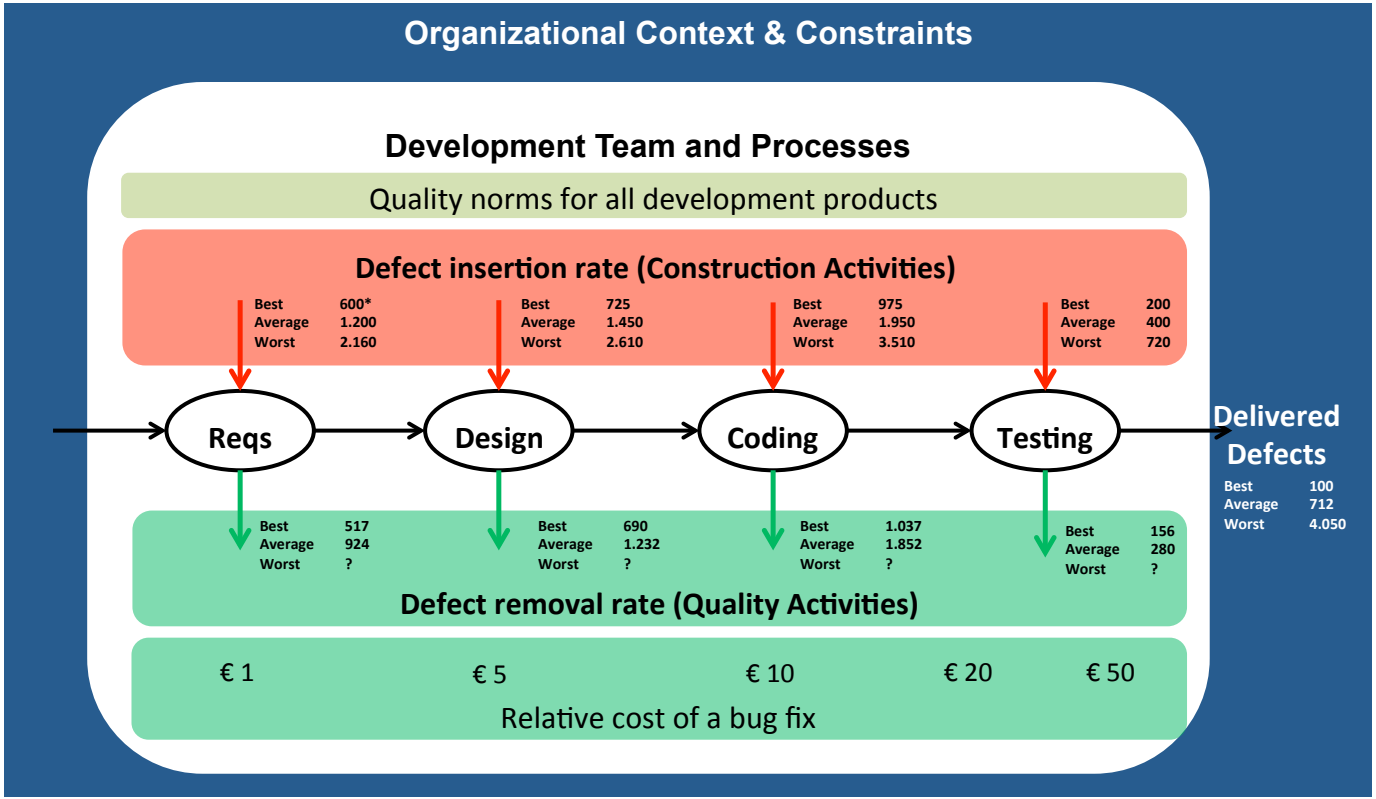
- We all know the examples of failing IT systems. Failing IT systems are an increasing risk for our society and will cost us more and more. Our exploration of failing IT systems starts with a classic example, the Mars Climate Orbiter.
- The Mars Climate Orbiter was a robotic space probe launched by NASA on December 11, 1998 to study the Martian climate, Martian atmosphere, and surface changes.
- NASA lost a $125 million Mars orbiter because a Lockheed Martin engineering team used English units of measurement while the agency's team used the more conventional metric system for a key spacecraft operation.
- With two teams following different unit systems and no strict conversion checks in place, this is a textbook example of why interface documents have to be precise. The end result was that the module came in too low and too fast and disintegrated.



The increasing risk of software failure
What did NASA do to prevent bugs and failures?

- Defects in software are called bugs. Constructing software is very time consuming and error prone.
  - The first ‘bug’ (1946) in the Harvard Mark II computer
- Intrinsic complexity of software development
  - The trouble with making things out of code, as opposed to something physical, is the invisibility to the eye.
  - "No one's skull is really big enough to contain a modern computer program" (Dijkstra, The Humble Programmer, 1972)
  - Software has enabled us to make the most intricate machines that have ever existed. And yet we have hardly noticed, because all of that complexity is packed into tiny silicon chips as millions and millions of lines of code.
  - But just because we can’t see the complexity doesn’t mean that it has gone away.

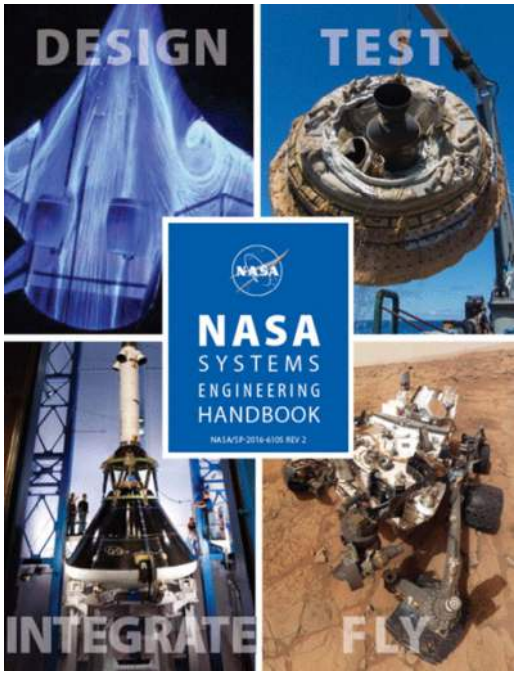
The increasing risk of software failure
What did NASA do to prevent bugs and failures?



*Based on a system of 1.000 function points

The increasing risk of software failure
What did NASA do to prevent bugs and failures?

- Guidelines and best practices for development processes and software work products



NASA Life Cycle Phases	Pre-Systems Acquisition	FORMULATION	Approval for Implementation	IMPLEMENTATION	Systems Acquisition	Operations	Decommissioning
Project Life Cycle Phases	Pre-Phase A: Concept Studies	Phase A: Concept & Technology Development	Phase B: Preliminary Design & Technology Completion	Phase C: Final Design & Fabrication	Phase D: System Assembly, Int & Test, Launch	Phase E: Operations & Sustainment	Phase F: Closeout
Project Life Cycle Gates & Major Events	KDP A, FAD, Draft Project Requirements	KDP B, Preliminary Project Plan	KDP C, Baseline Project Plan[3]	KDP D	KDP E, Launch	KDP F, End of Mission	Final Archival of Data
Agency Reviews	ASP[5]	ASM[5]					
Human Space Flight Project Reviews[1]	MCR	SRR, SDR (PNAR)	PDR (NAR)	CDR / PRR[2]	SIR, SAR, ORR	FRR, PLAR, CERR[3]	DR
Re-flights			Re-enters appropriate life cycle phase if modifications are needed between flights[6]		Inspections and Refurbishment	End of Flight, PFAR	
Robotic Mission Project Reviews[1]	MCR	SRR, MDR[4] (PNAR)	PDR (NAR)	CDR / PRR[2]	SIR, ORR	FRR, PLAR, CERR[3]	DR
Launch Readiness Reviews						SMSR, LRR (LV), FRR (LV)	
Supporting Reviews	Peer Reviews, Subsystem PDRs, Subsystem CDRs, and System Reviews						

FOOTNOTES

1. Flexibility is allowed in the timing, number, and content of reviews as long as the equivalent information is provided at each KDP and the approach is fully documented in the Project Plan. These reviews are conducted by the project for the independent SRB. See Section 2.5 and Table 2-6.
2. PRR needed for multiple (≥4) system copies. Timing is notional.
3. CERRs are established at the discretion of Program Offices.
4. For robotic missions, the SRR and the MDR may be combined.
5. The ASP and ASM are Agency reviews, not life-cycle reviews.
6. Includes recertification, as required.
7. Project Plans are baselined at KDP C and are reviewed and updated as required, to ensure project content, cost, and budget remain consistent.

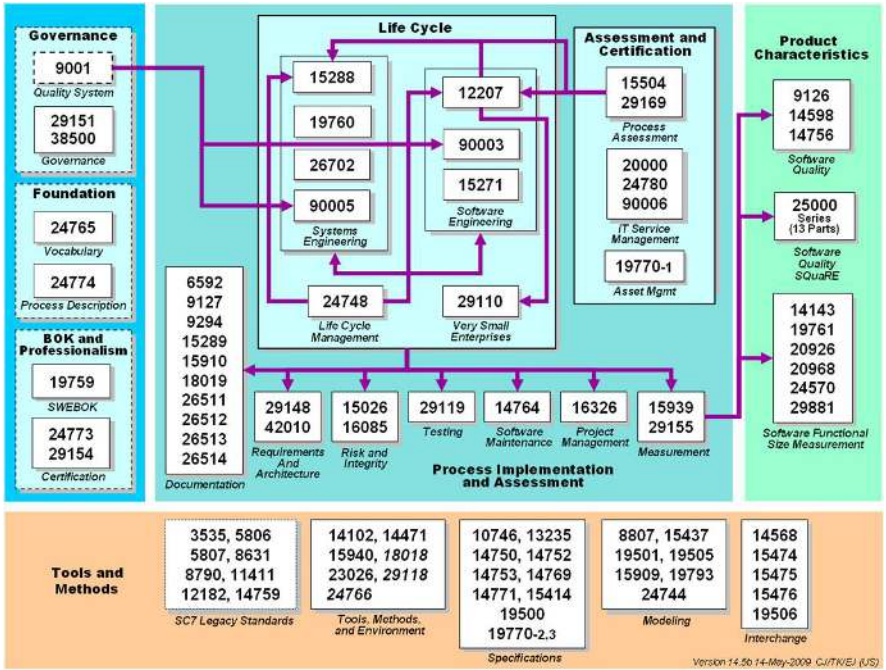
ACRONYMS

ASP—Acquisition Strategy Planning Meeting
ASM—Acquisition Strategy Meeting
CDR—Critical Design Review
CERR—Critical Events Readiness Review
DR—Decommissioning Review
FAD—Formulation Authorization Document
FRR—Flight Readiness Review
KDP—Key Decision Point
LRR—Launch Readiness Review
MCR—Mission Concept Review
MDR—Mission Definition Review
NAR—Non-Advocate Review
ORR—Operational Readiness Review
PDR—Preliminary Design Review
PFAR—Post-Flight Assessment Review
PLAR—Post-Launch Assessment Review
PNAR—Preliminary Non-Advocate Review
PRR—Production Readiness Review
SAR—System Acceptance Review
SDR—System Definition Review
SIR—System Integration Review
SMSR—Safety and Mission Success Review
SRR—System Requirements Review

Figure 5-2 – The NASA Project Life Cycle

The increasing risk of software failure
What did NASA do to prevent bugs and failures?

- Guidelines and best practices for development processes and software work products



SIL number	Required Safety Availability (RSA)	Probability of Failure on Demand (PFD)
SIL1	90% to 99%	0.1 to 0.01
SIL2	99% to 99.9%	0.01 to 0.001
SIL3	99.9% to 99.99%	0.001 to 0.0001
SIL4	99.99% to 99.999%	0.0001 to 0.00001

Generic (IEC 61508)	**(SIL 0)**	**SIL 1**	**SIL 2**	**SIL 3**	SIL 4
Civil Aerospace (DO-178C)	Level E	Level D	Level C	Level B	Level A
Medical (IEC 62304)	Class A	Class B		Class C	
Automotive (ISO 26262)	QM	ASIL A	ASIL B / ASIL C	ASIL D	--
Machinery (ISO 13849)	PL a	PL b / PL c	PL d	PL e	--
Household (IEC 60730)	Class A	Class B		Class C	--

The increasing risk of software failure
What about our modern society?

Everything Connected

Smart Contracts

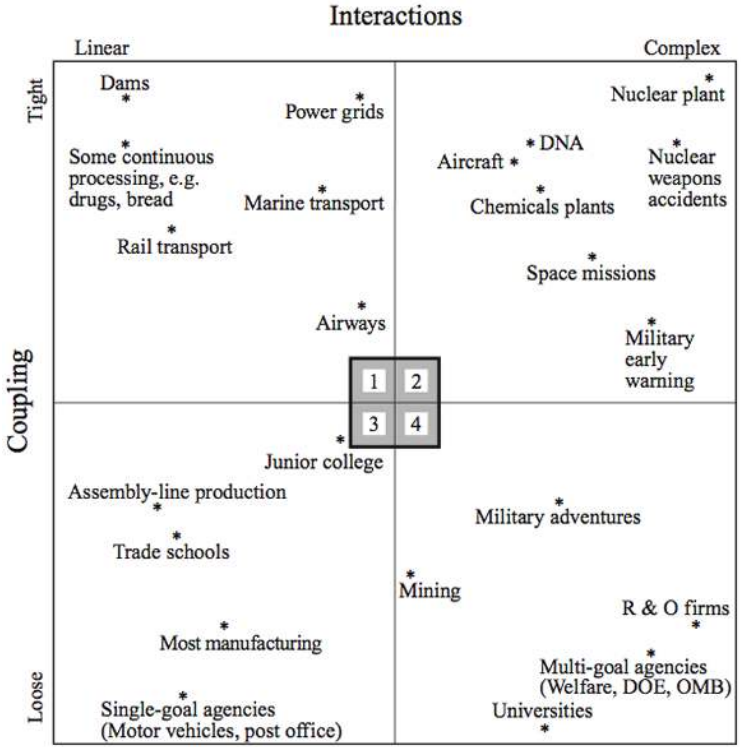
Mobile Medical Devices

The increasing risk of software failure
What about our modern society?

- May 6, 2010, the Flash Crash, was a United States trillion-dollar stock market crash, which lasted for approximately 36 minutes
- Just prior to the Flash Crash, the trader placed thousands of future contracts which he planned on canceling later.
- Computer based trading algorithms reacted on a spoofing algorithm used by a trader.

The increasing risk of software failure
What about our modern society?

- Be prepared for normal accidents when
  - Systems are tight coupled, and
  - Have complex interactions



What measures do we have in place?

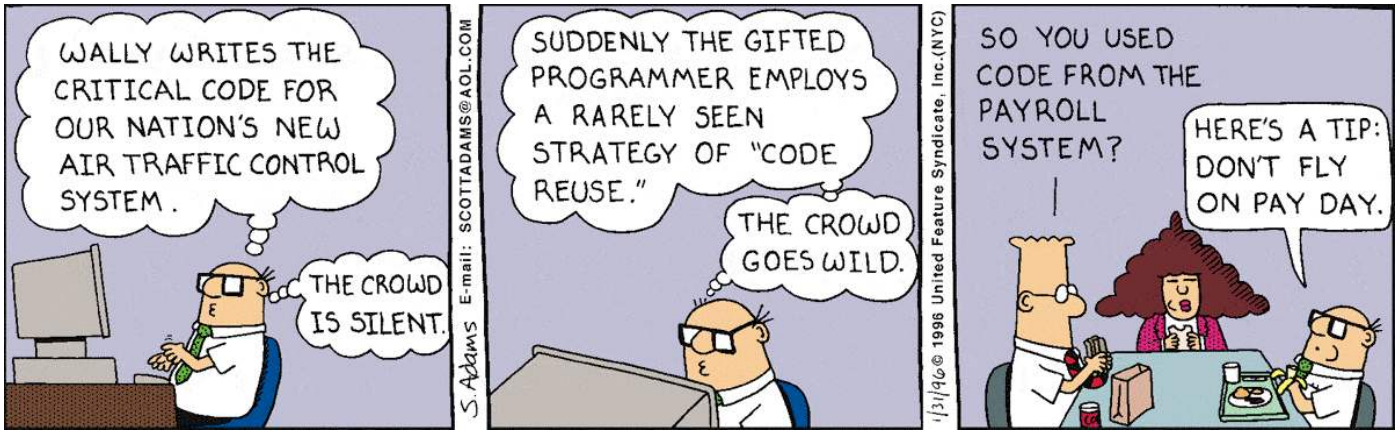
VALORI

- What did we learn so far?
  - Software enables us to build the most complex machines ever
  - It is hard to build high quality, even the best in class teams leave defects in software
  - There are many standards and guidelines for building (safety) critical software
  - In our increasing connected world, system failures have increasing impact, normal accidents
  - More and more systems and their connections become critical for our society and our lives

- So, …

- How do we develop our professional workforce to cope with this increasing complexity and criticality?

What measures do we have in place?
Do we trust the programmer?



What measures do we have in place?
Do we trust programmers and IT management?

VALORI

- Learning how to program is still the largest obstacle in higher education.
- Still the number of programmers roughly doubles every five year. Where do those programmers come from?
- So the market offers different possibilities to become a programmer
  - When you go to the website icttrainingen.nl you can apply for title Certified Java Programmer. It will cost you 40 hours of study and paying for the exam. When you pass the exam you receive the certificate
  - Programit.nl offers three month courses to become a junior C# or Java developer.
- And they are hired…

What measures do we have in place?
Do we trust programmers and IT management?

VALORI

	Kind of software			
	Throw Away	Business Systems	Mission Critical	Safety Critical
Example software	Campaign Site Prototypes Excel calculations One time reports	Internet site Inventory management Games Payroll system	Embedded software Packaged software Software tools Web services	Avionics software Embedded software Medical devices Operating systems
Requirements	Ad hoc, informal requirements specification	Informal requirements specification	Semiformal requirements specification As-needed requirements reviews	Formal requirements specification Formal requirements inspections
Design	Design and coding are combined	Design and coding are combined	Architectural design Informal detailed design As-needed design reviews	Architectural design Formal architecture inspections Formal detailed design Formal detailed design inspections
Construction	Individual coding No check-in procedure	Pair programming or individual coding Informal check-in procedure or no check-in procedure	Pair programming or individual coding Informal check-in procedure As-needed code reviews	Pair programming or individual coding Formal check-in procedure Formal code inspections
Testing & QA	Developers test their own code Trial and error No separate testing group	Developers test their own code Test-first development Little or no testing by a separate test group	Developers test their own code Test-first development Separate testing group	Developers test their own code Test-first development Separate testing group Separate QA group

What measures do we have in place?
Do we trust the programmers and IT management?

VALORI

	Kind of software			
	Throw Away	Business Systems	Mission Critical	Safety Critical
Example software	Campaign Site Prototypes Excel calculations One time reports	Internet site Inventory management Games Payroll system	Embedded software Packaged software Software tools Web services	Avionics software Embedded software Medical devices Operating systems
Requirements	Ad hoc, informal requirements specification	Informal requirements specification	Semiformal requirements specification As-needed requirements reviews	Formal requirements specification Formal requirements inspections
Design	Design and coding are combined	Design and coding are combined	Architectural design Informal detailed design As-needed design reviews	Architectural design Formal architecture inspections Formal detailed design Formal detailed design inspections
Construction	Individual coding	Pair programming or individual coding Informal check-in procedure or no check-in procedure	Pair programming or individual coding Informal check-in procedure As-needed code reviews	Pair programming or individual coding Formal check-in procedure Formal code inspections
Testing & QA	Developers test their own code Trial and error No separate testing group	Developers test their own code Test-first development Little or no testing by a separate test group	Developers test their own code Test-first development Separate testing group	Developers test their own code Test-first development Separate testing group Separate QA group

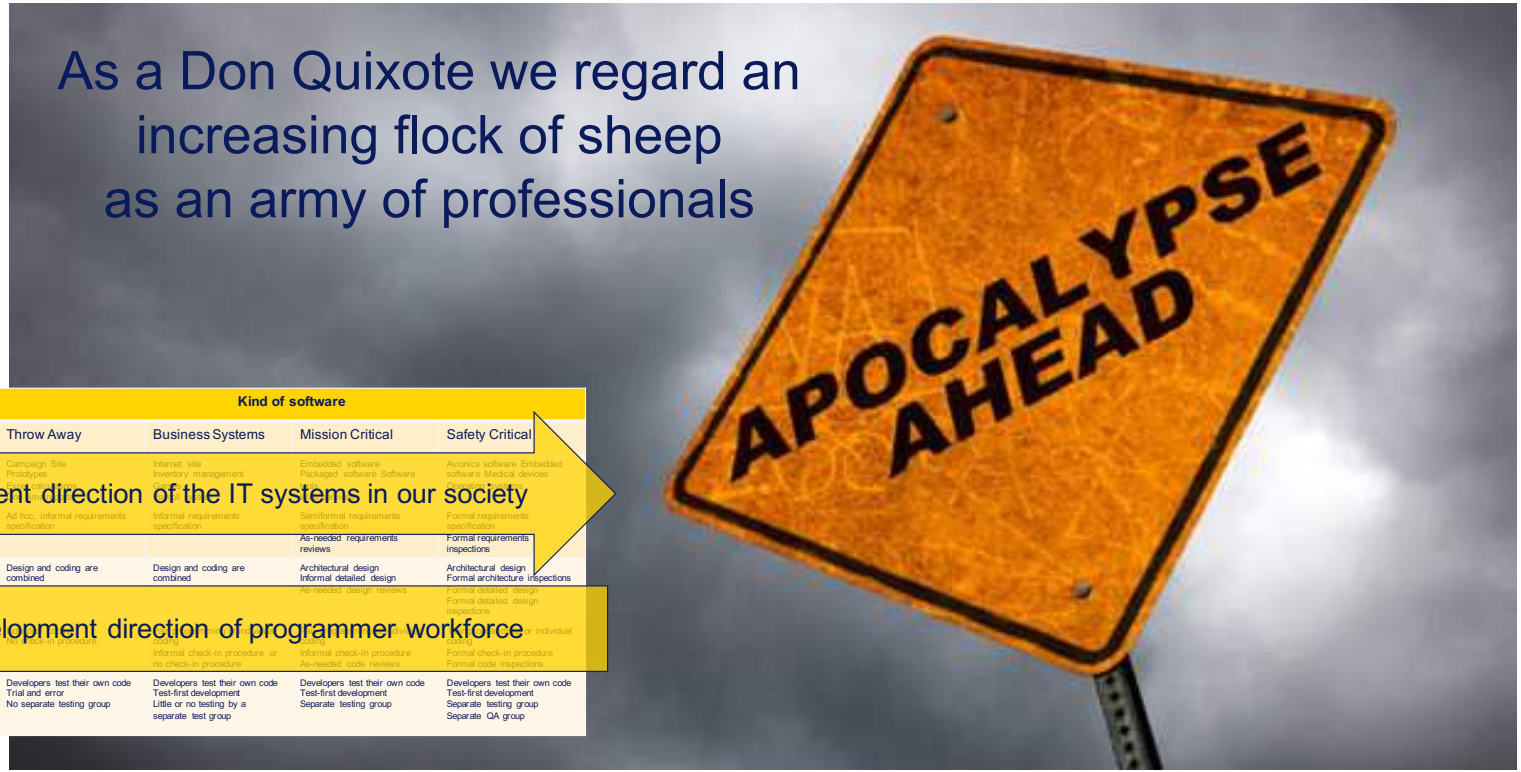
Development direction of the IT systems in our society

Development direction of programmer workforce

What measures do we have in place?
Do we trust programmers and IT management?

VALORI

As a Don Quixote we regard an increasing flock of sheep as an army of professionals



Development direction of the IT systems in our society

Development direction of programmer workforce

	Kind of software			
	Throw Away	Business Systems	Mission Critical	Safety Critical
Example software	Campaign Site Prototypes	Internet site Inventory management	Embedded software Packaged software Software	Avionics software Embedded software Medical devices
Requirements	Ad hoc, informal requirements specification	Informal requirements specification	Semiformal requirements specification As-needed requirements reviews	Formal requirements specification Formal requirements inspections
Design	Design and coding are combined	Design and coding are combined	Architectural design Informal detailed design As-needed design reviews	Architectural design Formal architecture inspections Formal detailed design Formal detailed design inspections
Construction	No check-in procedure	Informal check-in procedure or no check-in procedure	Informal check-in procedure As-needed code reviews	Formal check-in procedure Formal code inspections
Testing & QA	Developers test their own code Trial and error No separate testing group	Developers test their own code Test-first development Little or no testing by a separate test group	Developers test their own code Test-first development Separate testing group	Developers test their own code Test-first development Separate testing group Separate QA group

The rough road ahead

- Accidents will be increasingly normal…
  - … so what to do next?

- Create awareness for the upcoming apocalypse
  - Stimulate a Continuous dialog between developers and managers
- Create a craftsmanship culture
  - High quality education, certification
  - Master – Apprentice model
  - It will be hard work, requiring continuous attention
- Or stop writing source code
  - Create less code, we humans are not fit to write code
  - Develop new approaches to instruct machines

The rough road ahead..
Join the struggle for increasing craftsmanship!

- I am worried about the upcoming software apocalypse in a world which increasingly depends on software systems
- Increasing craftsmanship is required and should be created and promoted
- Will you join me?

Wim Goes
Valori Software Improvement
E: WimGoes@valori.nl
T: 06 50 999 666