Beyond modeling: let the system meet the specifications

www.altreonic.com
eric.verhulst (@) altreonic.com

From Deep Space to Deep Sea

Trustworthy Forever

Content

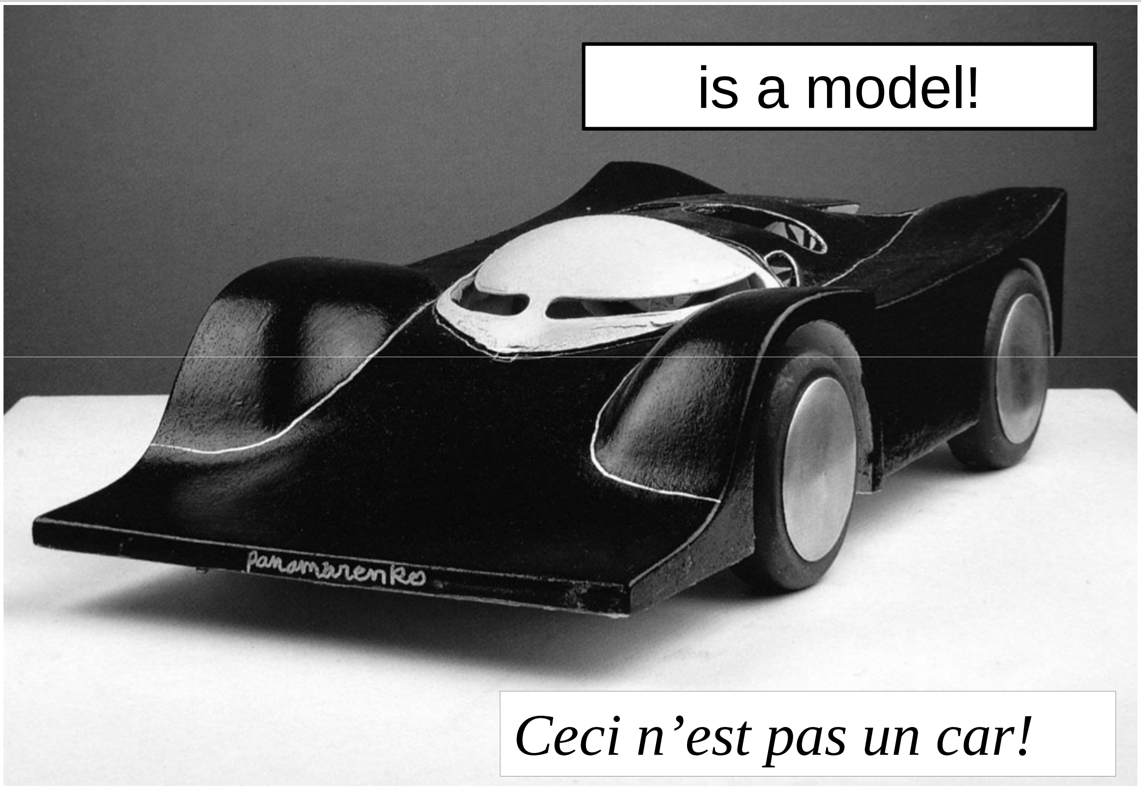
- What are „Models“ ?
- Systems, safety and other properties
- Models and processes
- Why it pays off
- The human factor: Maturity of an organisation
- Conclusion

Abstract

While modeling is an important step forward in getting software right, it is often thought to be a sufficient condition. The role of modeling is to verify that a specific implementation is feasible. Different types of models are needed. The implementation itself is to be seen as a model as well. To get it right, the process needs to start at the requirements level, even before any model is being considered.

An important aspect of modeling approaches is that they introduce the concept of meta-levels. This allows to reason at a more abstract level. The challenge is that different views need to be combined, in particular the development or design view and the process view. While these views are often seen as separate activities, in practice they are very much connected because engineering is also team work executed by humans. In the presentation we will show how a rigorous systems/software engineering approach can be combined with an agile process that takes both views into account.

The safe car that doesn’t move



What is a model?

Wittgenstein (in “Philosophical Grammar” 65 years ago)

*“A **blueprint** serves as a **picture** of the object which the workman is to make from it.*

*… for the builder or the engineer, the blueprint is used as an **instruction or rule dictating how** he should construct the building or machine. And if what he makes deviates from the blueprint, then he has erred, built incorrectly en must try again.”*

*… What we may call **‘picture’ is the blueprint together with the method of its application”.***

Wittgenstein defined Systems Engineering

before the term even existed.

Why models?

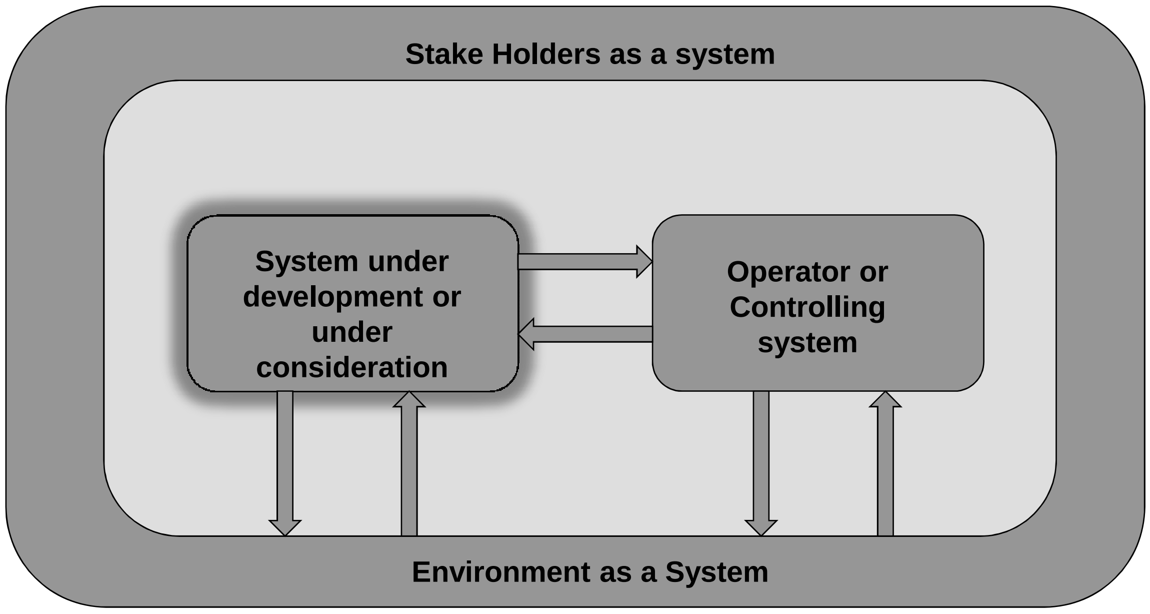
- Language is not just about syntax, it is about **semantics** as perceived and used by humans. Language is fuzzy.
- Models help to **reason** and **communicate**.
- Therefore models are about language and exist in different variants depending on the view and domain (meaning in language is context dependent)
- **What** is it that we want to build?
- **How** do we want to build it?
- What is the link between the “what” and the “how”?
  - **Building the right thing**
  - **Building the thing right**

What did Wittgenstein really say?

- A **model** is a **projected view**
- A model assumes a **methodology**
- We can only faithfully make the transition from model to system, if the model is complete
  - ➧ also the **implementation is a model**
- We can only faithfully generate the implementation, if we have a completely defined mapping between the model(s) as a set of projected views and the selected implementation.
- Therefore: **Models and Process are strongly linked**.
- The issue: both are actually very large state spaces!
  - ➧ **mastering the complexity** is the challenge!

What system?

Any system is part of a larger system


Stake Holders as a system
System under development or under consideration
Operator or Controlling system
Environment as a System

What system properties?

Any system has to meet different, often conflicting properties:

- Cost price
- Energy use
- Safety
- Security
- Size
- Ease of use
- Designed for “production”
- Must be produced by company X
- Designed to meet the requirements of the target market
- Life-cycle cost
- ….

➧ **Trade-offs**

- ➧ All properties are related
- ➧ The best technical solution is not necessarily the one selected

From a safety goal to a view of Trust

- **Historical:**
  - Concept that is related to the loss of lives due to a malfunctioning of the system. Post factum: what went wrong?
- **The right safety view:**
  - **Safety is an emerging system property resulting from a quality engineering process**
  - **Reliability is a pre-condition, but not sufficient condition**
  - Safety can be improved by using feedback
  - Bottom line: **do users trust the system/ product ?**
- **The expanded view: <u>Trustworthiness</u> =**
  - **<u>Safety</u>**: preventing damage or the loss of lives due to unintentional failures or malfunctioning parts +
  - **<u>Security</u>**: preventing damage or the loss of lives due to maliciously injected failures or malfunctions +
  - **<u>Useability</u>**: preventing damage or the loss of lives due to improper operator interfaces +
  - **<u>Privacy</u>**: preventing loss or misuse of personal data.

Safety, reliability, predictability

"Safety and reliability are different properties. One does not imply nor require the other : ***A system can be reliable but unsafe. It can also be safe but unreliable.*** In some cases, these two properties even conflict, that is, making the system safer may decrease reliability and enhancing reliability may decrease safety."

(src: *Nancy Leveson*, Engineering a Safer World)

- Predictability is a higher level property of any other property. It reflects our control of the engineering process.

Example of safety case

Risk of injury.
Under certain circumstances, due to a software issue, the product can unexpectedly apply reverse torque to the wheels, which can result in a rider falling and potentially suffering injuries. That this can occur in two situations: during a safety shutdown of the product, or when the rider exceeds the programmed speed limit.

Both situations involve specific sequences of events under narrow timeframes, and require that the handlebar be tilted back by the speed limiter and the rider come off and then back onto the rider detect switches on the riding platform within a short period of time combined with a traction control event. At least 6 incidents have been reported resulting in injuries to the head and wrist of users.

src: http://ec.europa.eu/consumers/dyna/rapex/rapex_archives_en.cfm

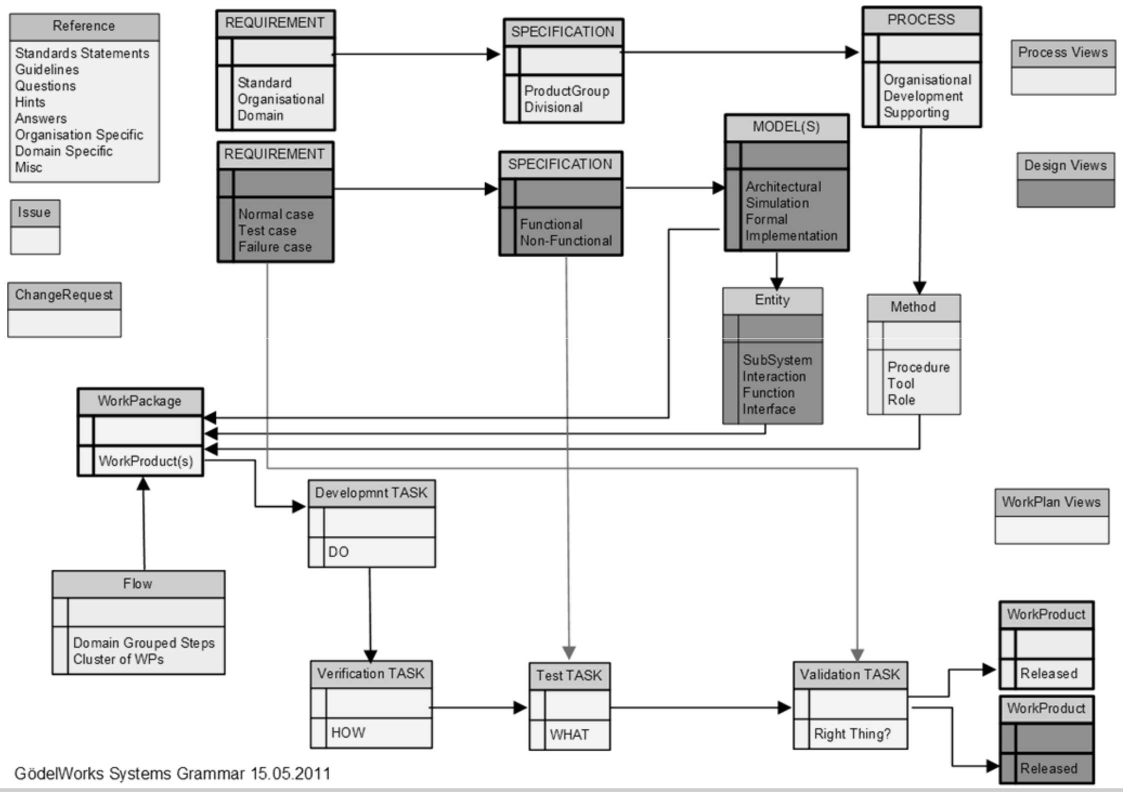
Is safety absolute?

1. **Safety can never be absolute:**
   - Always residual errors
   - Always residual risks
   - Design is always trade-off.
2. **Safety is a statistical property:**
   - Mean Time To Failure is what matters for malfuntions => mean time to safety hazards
   - If MTTF >> life time, mainly external factors remain:
     - Operator
     - Environment
3. **Safety level must be selected on the basis of acceptable risks**
   - Has a cost tag (insurance) attached to it
   - Safety Integrity Level 3 (SIL3)
     - Fail-safe mode, but still safety hazard
   - Safety Integrity Level 4 (SIL4)
     - Fault tolerant, but still residual risks
       - Common mode failures
       - Common design mistakes

➧ **A safety hazard can still happen ANY time!**

➧ **Most people are optimistic and then become negligent**

Models as part of the Process



GödelWorks Systems Grammar 15.05.2011

How is certification reached?

1. Follow a **formalised (safety) engineering process**
   - Compatible with safety standards (often domain specific)
   - Organisation must be set up for it: mindset issue!
   - A good flow is **iterative**
     - Step1:
       - **Requirements** capturing
       - Many stakeholders, nice-to-haves, must haves
       - Normal case, Fault cases, Test cases
     - Step2:
       - Select requirements and write up **specifications**
     - Step3:
       - **Model**: (virtual) prototypes, simulations, formal models of critical properties, implementation models
     - Step4:
       - **Verify the process**
       - **Test** against specifications
       - **Integrate and validate** against requirements
2. **Release**
3. **Certify**

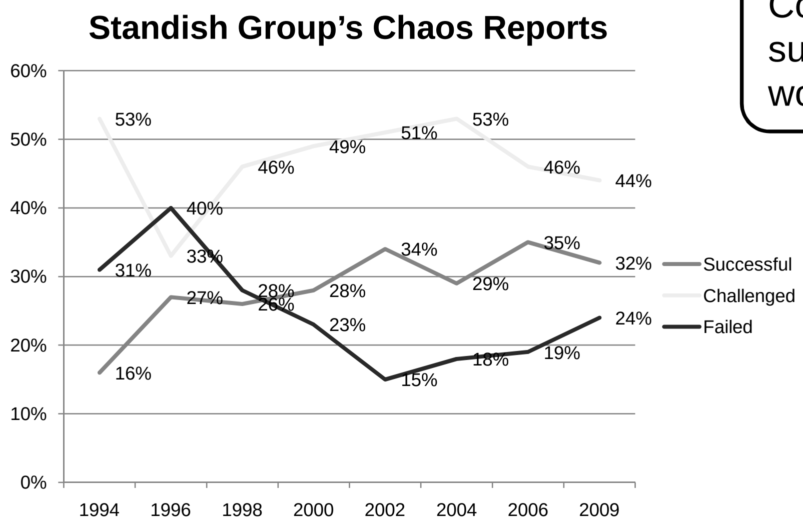
Why a unified and formalised approach is needed

- **Many stakeholders:**
  - Political
  - Financial
  - Marketing
  - Engineering
  - Users
- **Many domains** => many domain-specific languages
  - Requires unified semantics ("ontology")
  - Even in the technical domain!
- If no clear and common understanding is reached, there will be too many conflicting requirements, misunderstood requirements and hence the system will have hidden flaws.
- **Selecting the right system is the first step to develop it right.**
- **This work has to be done up front.**
  - This is also the cheapest phase for correcting mistakes
  - The further in the process, the more expensive
  - Risk of stopped projects
  - Risk of run-way costs

IT Project Success ratio

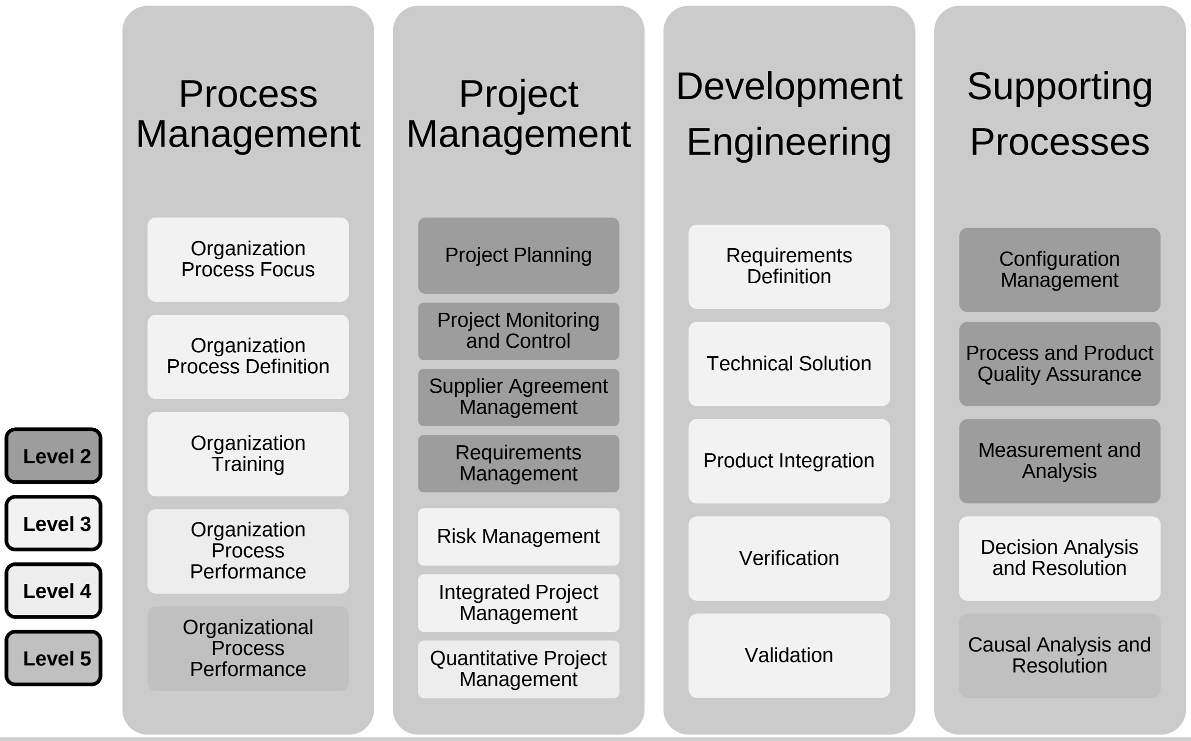
IT projects current scenario:

Continuous low % of successful projects worldwide.



Successful – delivered on-time, on-budget, on-quality and on-scope
Challenged – at least one element failed
Failed – project canceled

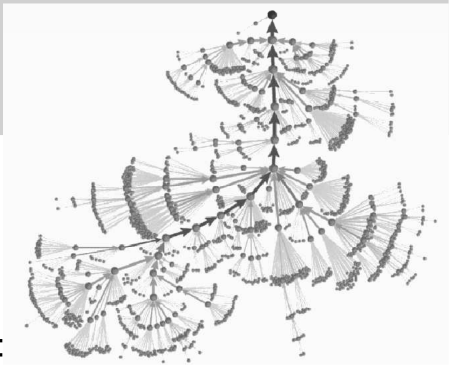
CMMI: levels of maturity


Level 2
Level 3
Level 4
Level 5
Process Management
Organization Process Focus
Organization Process Definition
Organization Training
Organization Process Performance
Organizational Process Performance
Project Management
Project Planning
Project Monitoring and Control
Supplier Agreement Management
Requirements Management
Risk Management
Integrated Project Management
Quantitative Project Management
Development Engineering
Requirements Definition
Technical Solution
Product Integration
Verification
Validation
Supporting Processes
Configuration Management
Process and Product Quality Assurance
Measurement and Analysis
Decision Analysis and Resolution
Causal Analysis and Resolution

The impact of electronics and software

- Why electronics and software?
  - Programmable => easy to change => also risk
  - Cheap, small
  - Allow more sophisticated functionality
    - Modern planes can't fly anymore without
    - Also key for lower energy and cleaner operation
- BUT:
  - Mechanical predecessors fail gracefully in the continuous domain
  - Electronic and software are clocked in the discrete domain
    - Fail within one clock cycle (typically 20 nanoseconds)
    - State space is huge (100 millions of states) because of data dependencies (1 integer = 2**32 states)
    - 10E-23 bit error rates => bit error becomes a certainty with time

The impact of complexity



- More functionality:
  - => complexity increases
  - => state space explodes exponentially
- **More dynamic behaviour means more complexity**:
  - Feedback loops needed to guarantee stability
  - Creates however difficult to find transition states between modes
    - Clearly an issue with Toyota Prius: older models have no issues, latest issues seem to be related to interplay of more advanced features like dual engine, ABS + ESP, regenerative braking, rough road, etc.
- But, where does the complexity come from?
  - Reusing old "proven in use" architectures (with layers of corrections)?
  - Or rather Layering of complexity?
- **Conclusion**: tackle complexity (and get safety) by cleaner architecture
  - Validate design and verify using formal approaches

3σ vs 6σ

99,0 %		99,9997 %
20,000 postal mails lost per hour	→	7 lost per hour
Unsuitable tap water during 15 min per day	→	Unsuitable tap water during 1 minute every 7 months
5,000 bad surgeries per week	→	1,7 bad surgeries per week
Two troubled landings per day	→	One troubled landing every 5 years
200.000 wrong persciptions per year	→	68 wrong perscriptions per year
Electricity outage for 7 hours per month	→	1 hour electricity outage every 34 years
68,200 defects per million		Only 3,4 defects per million

Data from 1 year samples recorded in the USA

Src: Critical Sofware

Benefits and Impact of CMMI

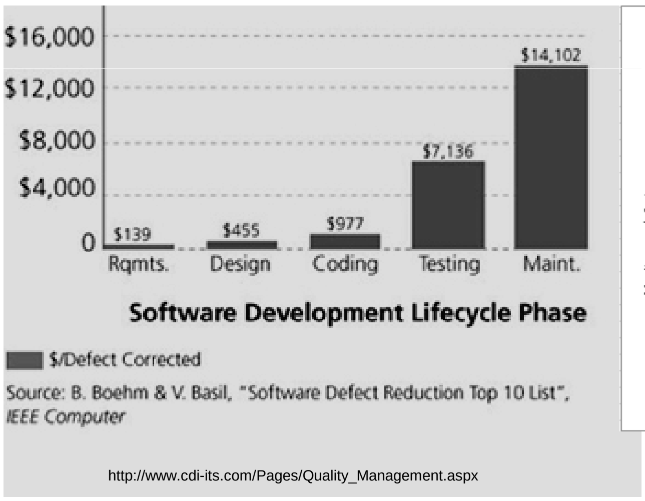
Area	Average improvement
Costs	20%
Schedules	37%
Produtivity	67%
Quality	50%
Customer Satisfaction	14%
Return Of Investment (ROI)	4.8 : 1

(sample size: 25 organizations)

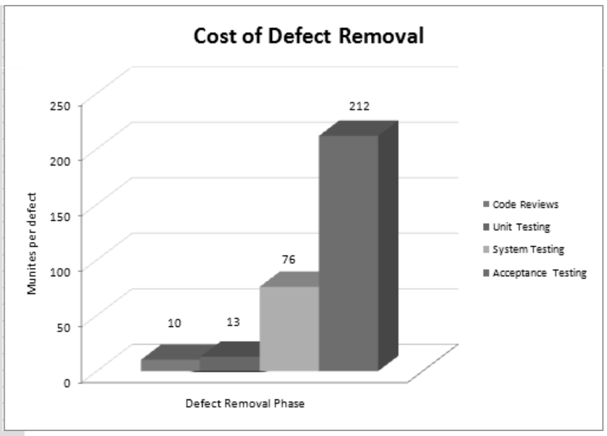
Src: SEI – Software Engineering Institute

Fixing up-front is cost-efficient!

- Finding and removing a nice to have requirement cost almost nothing
- Having to recall the product can cost billions



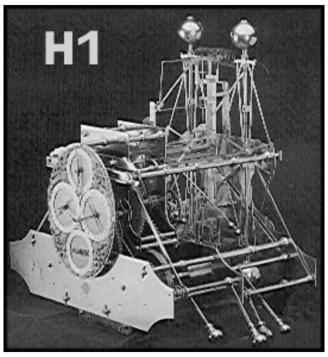
http://www.cdi-its.com/Pages/Quality_Management.aspx



Src: Critical Sofware

The biggest gain: the architecture

- Complexity is the enemy
- KISS: Keep It Simple but Smart
  - If a solution is complex, it means the problem is not well understood.
  - Simple solutions require a lot of thinking first.
- Technical notion of “elegance”: where engineering becomes an art.
- Examples:
  - NASA 1 million dollar space pen vs. 1 euro russian pencil
  - Harrison’s (1693-1776) time-keepers allowing navigation on sea:
    - “It has to be **practical**” => small and simple
    - Besides saving many lives, it made the British Empire possible.



Example: the OpenComRTOS project

- Original set-up:
  - How can we use formal methods to verify software?
  - Redevelop from scratch a distributed RTOS
- We used TLA+/TLC for formal modeling and following a process manually
- Results:
  - 6 months learning formal techniques, 12 months modeling, 2 weeks for running first code
  - Code size 10x smaller (5 kiBytes/node)
  - Much more scalable: heterogeneous

(see www.altreonic.com)

Some conclusions

- Systems must become **TRUSTWORHTY**
  - **Safety**: system operates as specified, always
    - Failures mitigation by redundancy
  - **Security**: integrity of system is assured
    - Freedom of maliciously induced failures
  - **Useability**:
    - The user/operator/environment is part of the system
    - What were his intentions ?
- How can we keep the system simple while increasing ROI (more functionality, more active safety features, less energy consumption, …)
- The key is a **formalised process, modelling helps**
- But pre-condition is the maturity of the organisation
- The root cause is always the **Human Factor**!

Thank You for your attention

Altreonic

“If it doesn't work, it must be art.
If it does, it was real engineering”