Innovation Assured

Software Engineering with Formal Methods: The storm surge barrier revisited

Klaas Wijbrans (Acision)
Franc Buve, Robin Rijkers, Wouter Geurts (Logica)

Company Introduction

- Acision is the world's leading messaging company
  - Over 50% of all SMS messages in the world are delivered by our products
  - Proven track record in Multimedia Messaging, Unified Messaging and Mobile Internet
  - Leader in standardization of Converged IP Messaging
  - Originated from the LogicaCMG Telecom Products division

- Logica is the leading IT company with a 40-year track record in innovative systems
  - Merged with CMG in 2002 to form LogicaCMG
  - Acquired WM-data, Edinfor and Unilog

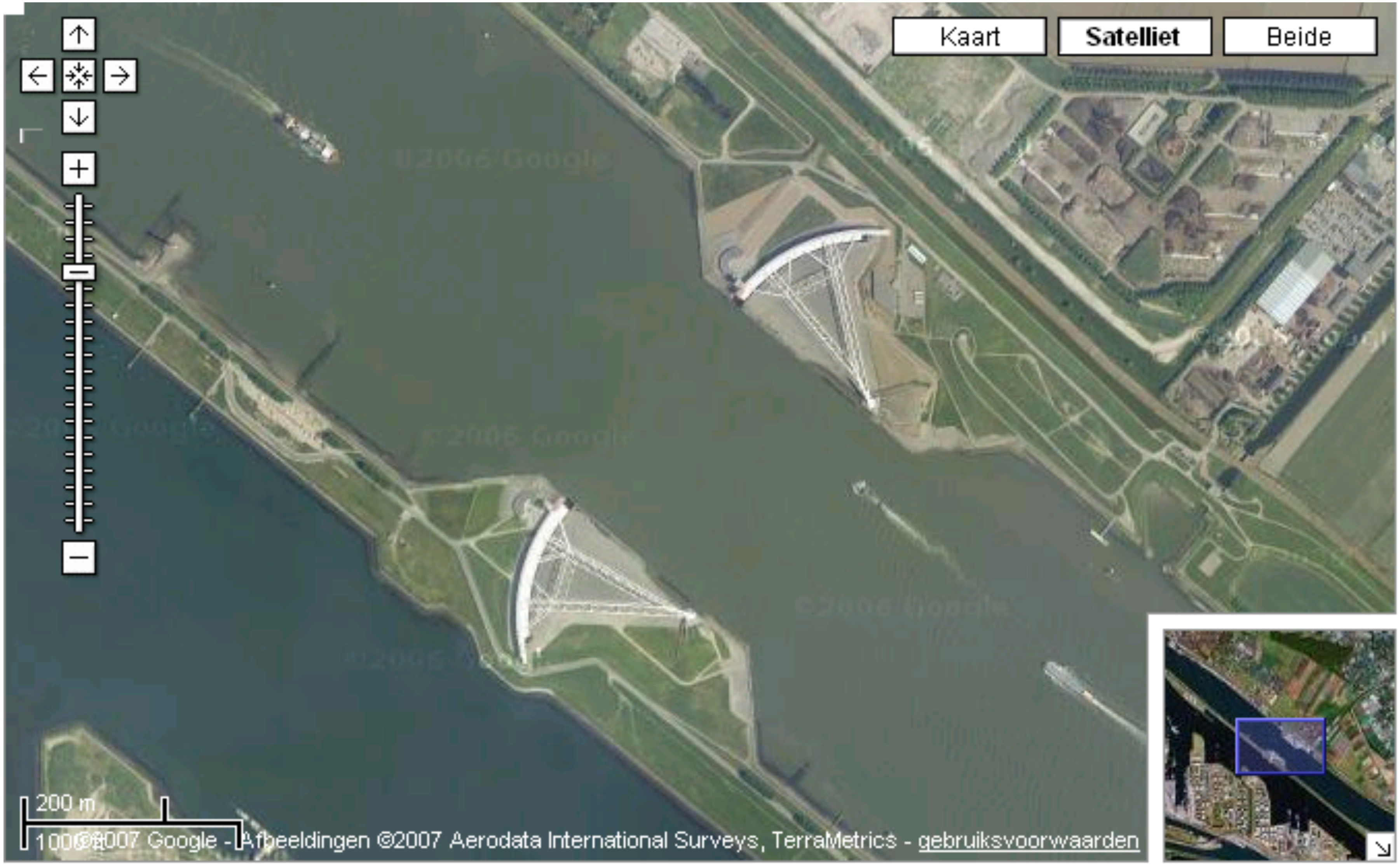
Introduction

Topics

- What is the Maeslant barrier and where is it located?
- Design principles behind the barrier
- Failure probability
- BOS
- Use of formal methods
- Lessons learned in operation
- The mid-life upgrade
- Current status and a look to the future

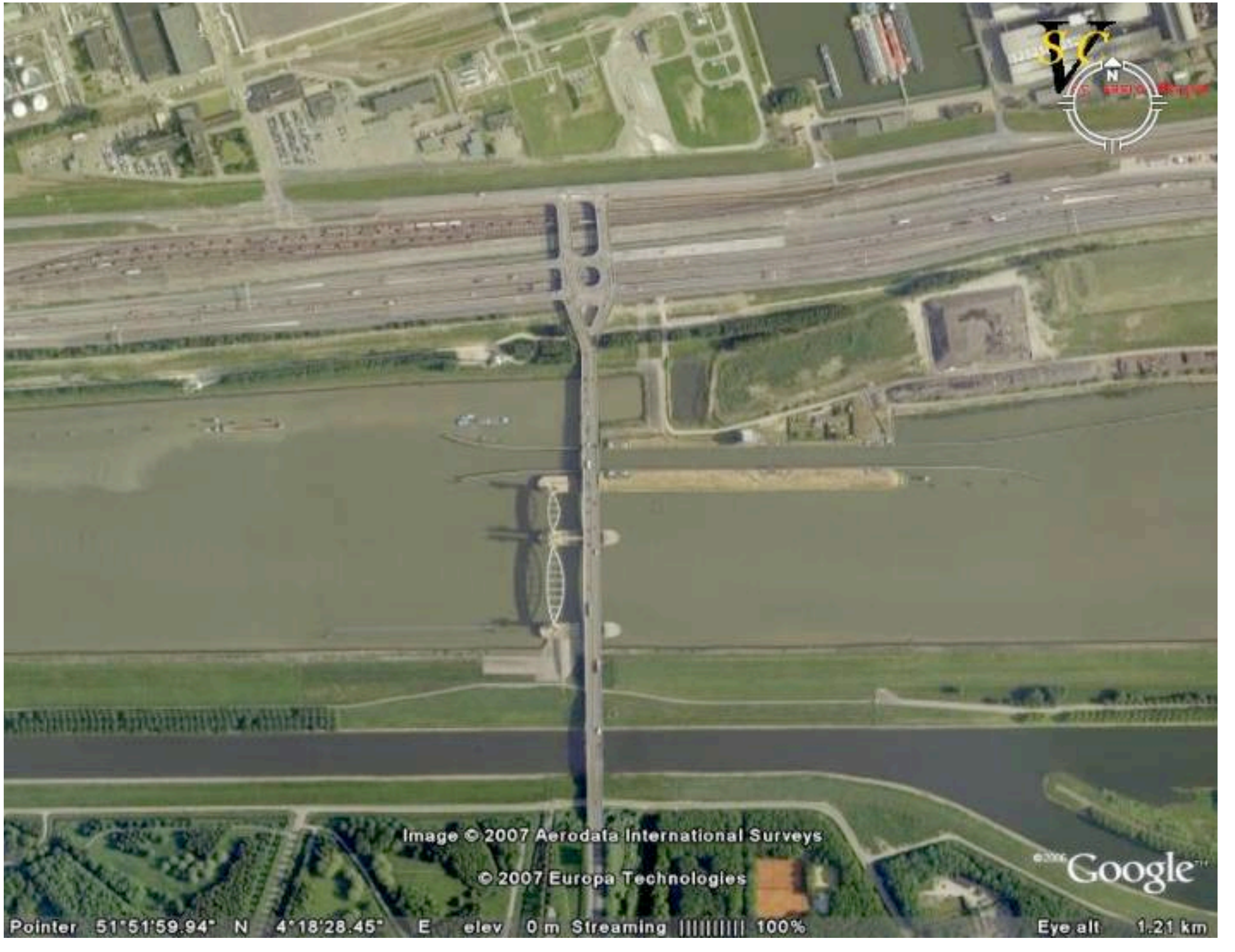
Location of barriers

Maeslantkering



Maeslantkering

Hartelkering



Hartelkering

Design Principles of the Barrier

- Conventional over-dimensioning for safety not feasible
- New approach in design
  - "Just good enough"
  - Failure probability analysis for every element in chain
- But:
  - Barrier must be just as reliable as a dike!

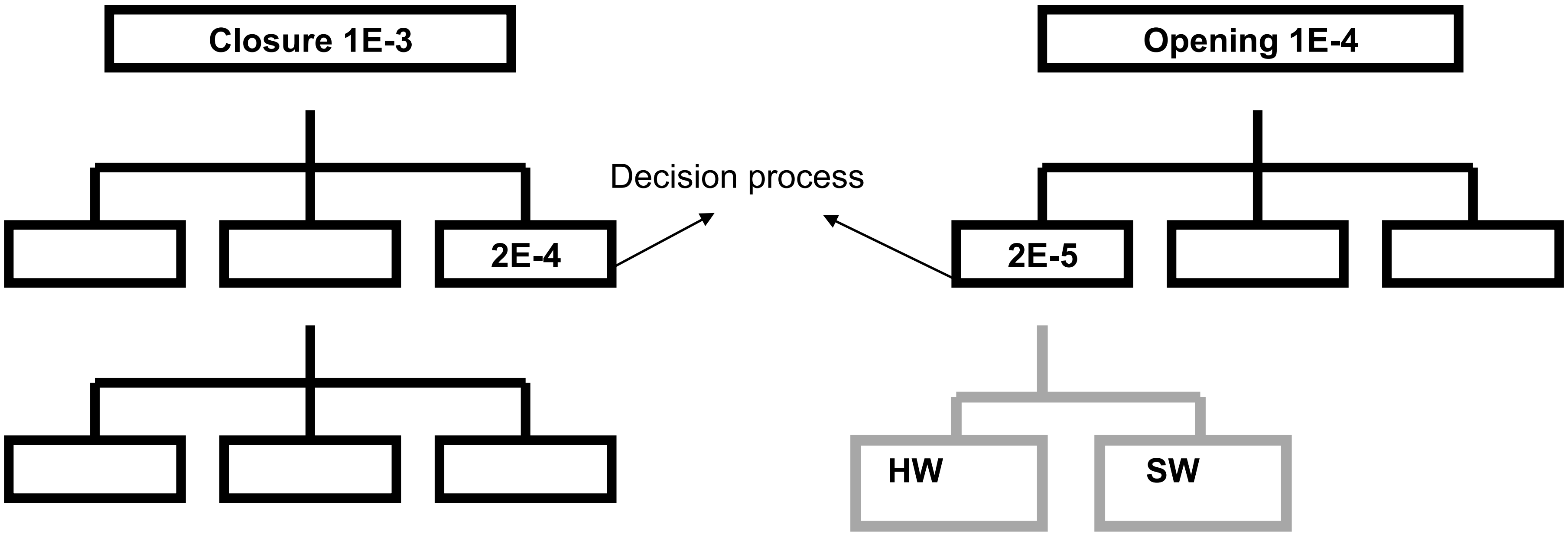
  - Acceptable risk of failure dike: 1 flooding in 10.000 years
  - Frequency of extreme high water: 1 storm in 10 years
  - Acceptable risk of failure barrier: **1 failure in 1.000 closures**

Not just an open/close decision

- Anticipate storm (minimal 8 hours) (to warn sea traffic) → predict
- Inform authorities → fax, pager
- Three barriers to control (Waterwegkering, Hartelkering and Hartelsluis) → mutual dependencies
- Unjustified closure very undesirable (economic interests) → critically tuned
- Unjustified not opening is dramatic → barrier destroyed
- Continually monitoring in submerged state (vulnerable for waves and water height from land side) → real-time monitor
- Detection of failure before it is too late → active monitoring
- Extensive maintenance procedures → support

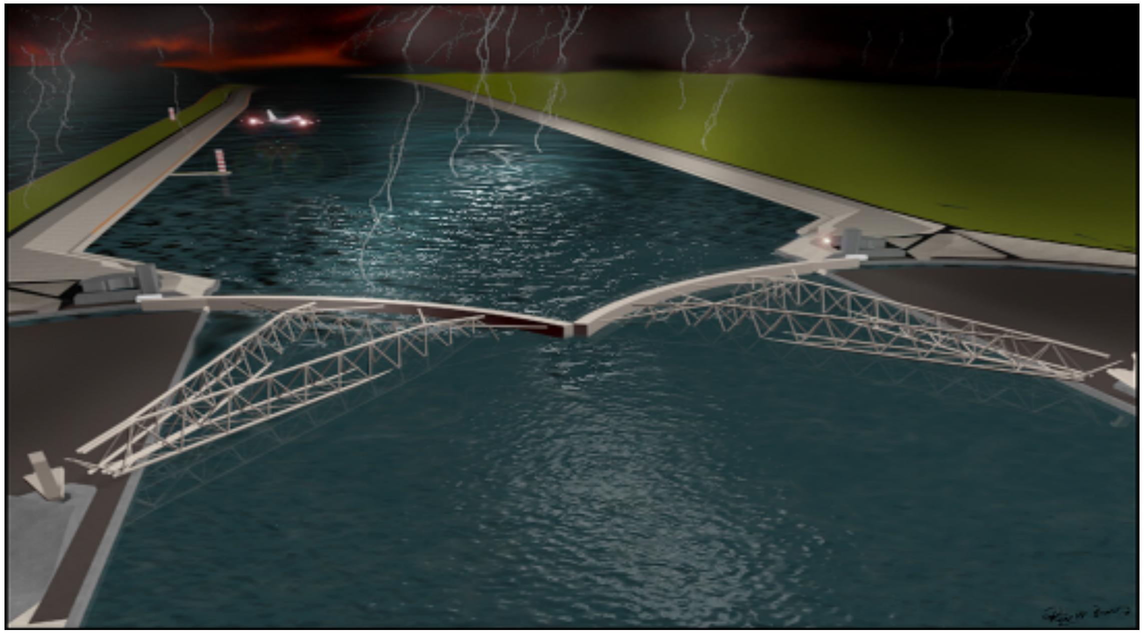
Failure Probability Tree

- Failure probability divided over components
  - Steel construction, joints, engines, electro-mechanics, decision system (BOS)
- Damage when not opening higher than not closing!
  - Failure to open: less than 1 in 10.000 (10^{-4})
  - Failure room for decision: 1 in 50.000 = 2 x 10^{-5}



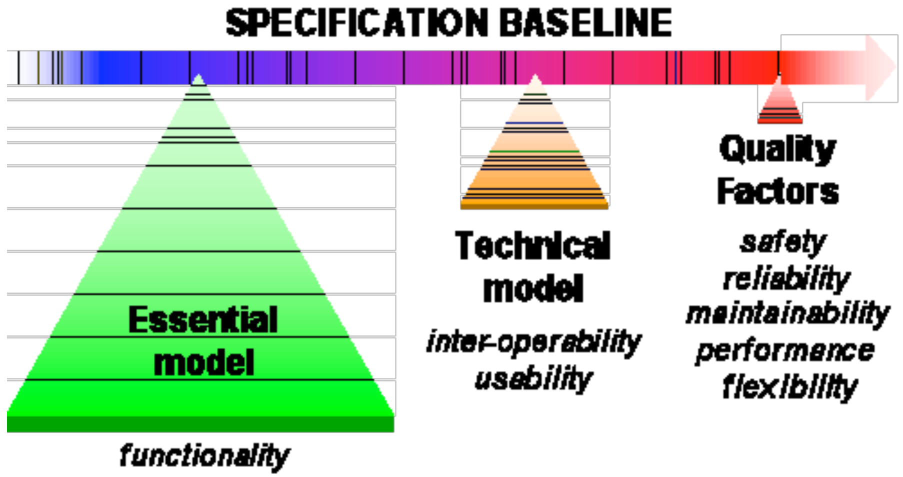
Failure Probability Tree

- Failure probability of decision of 2 x 10^{-5} impossible for humans
  - Average human 10^{-2}
  - Trained fighter pilot 10^{-3}
- Decision has to be automated =>
  - Beslis- en Ondersteunend Systeem (BOS)

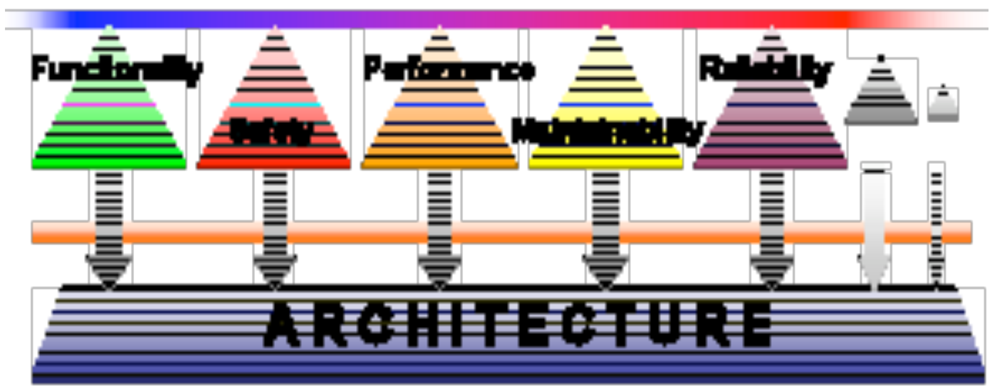


Design Approach

- IEC-61508 introduces Safety Integrity Levels for critical systems
- SIL-4 dictates use of risk-based approach
- Attention to non -functionals from the very beginning
- Rigorous development method including formal methods together with other techniques

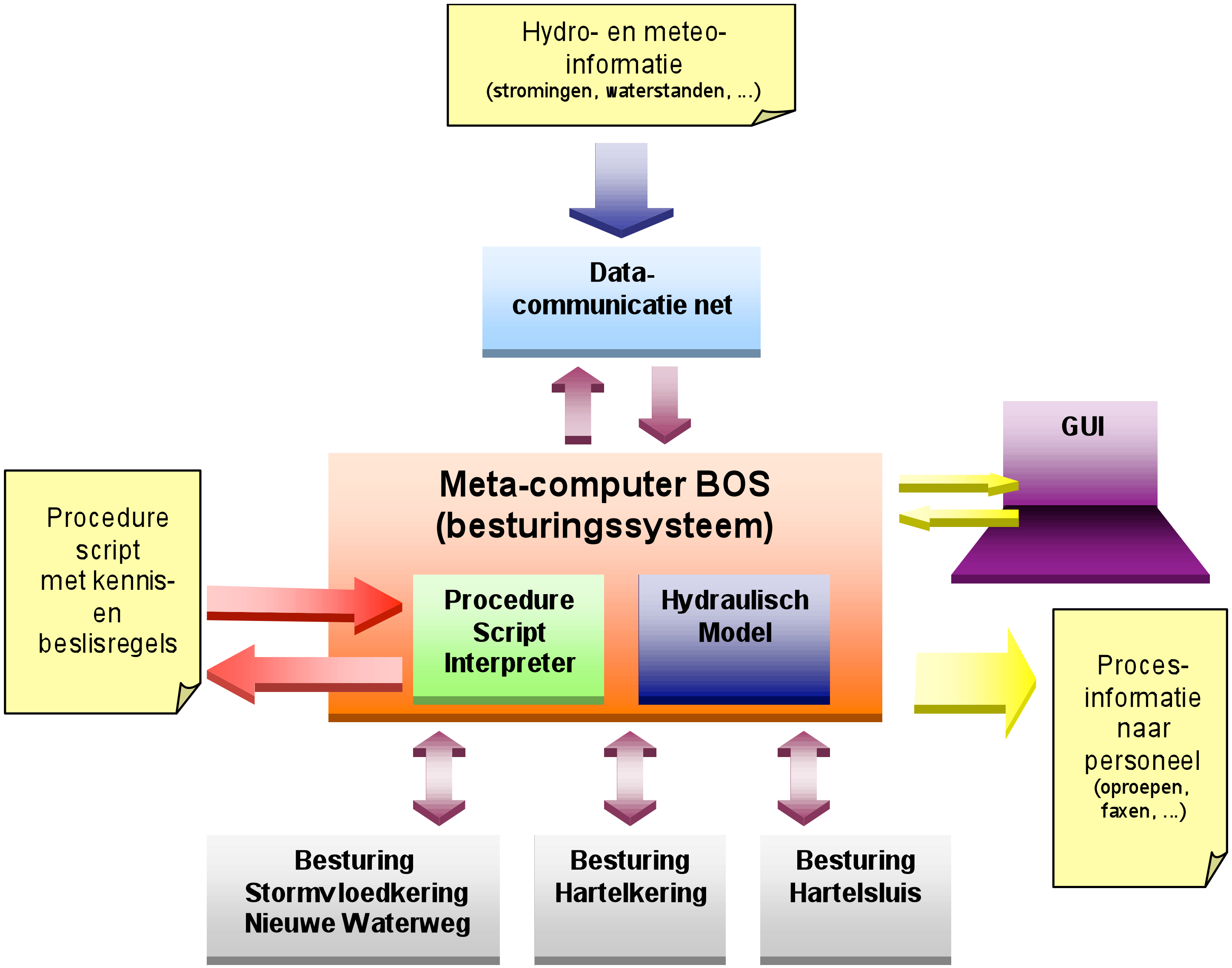


Conventional approach

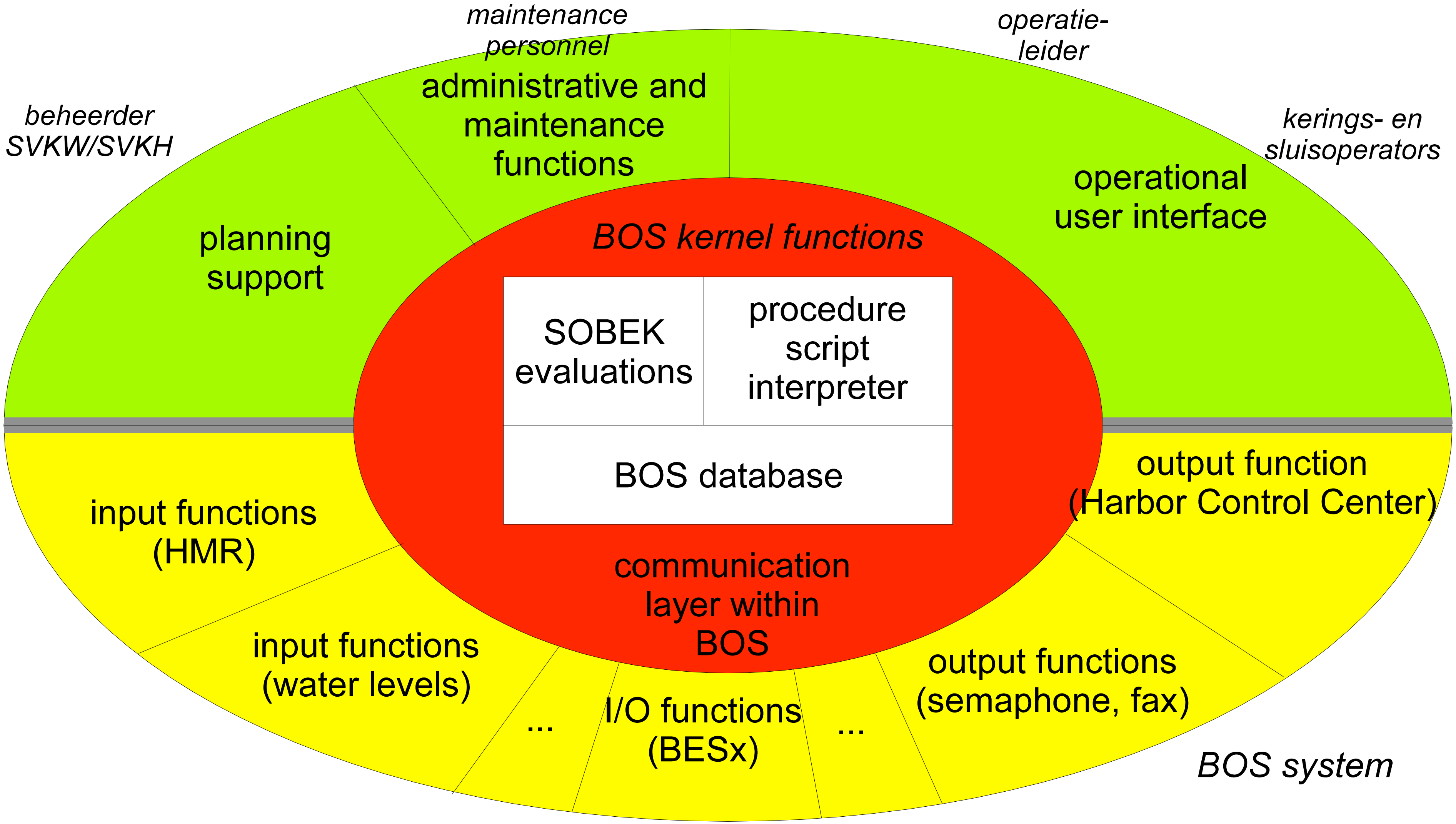


Risk-based approach

BOS Basic Concept

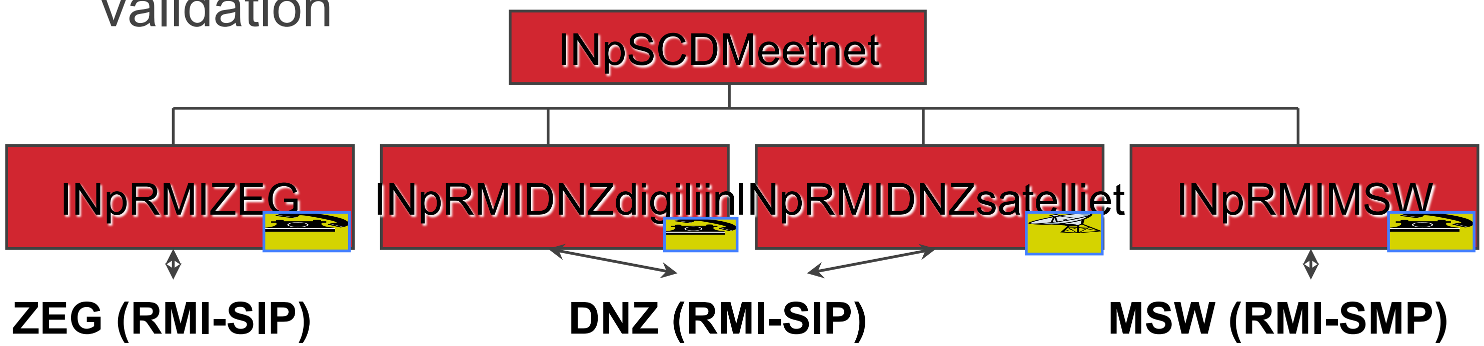


BOS architecture



Use of formal methods - 1

- Modeling and validation of communication and interaction
  - Process architecture modeled/validated in Promela/SPIN
  - Communication with external systems modeled;validated in Promela/SPIN
  - Ensures progress and absence of livelock/deadlock in core architecture
- Behavioral modeling proved to be easy to learn and very insightful
  - Significant changes at protocol level made because of formal validation

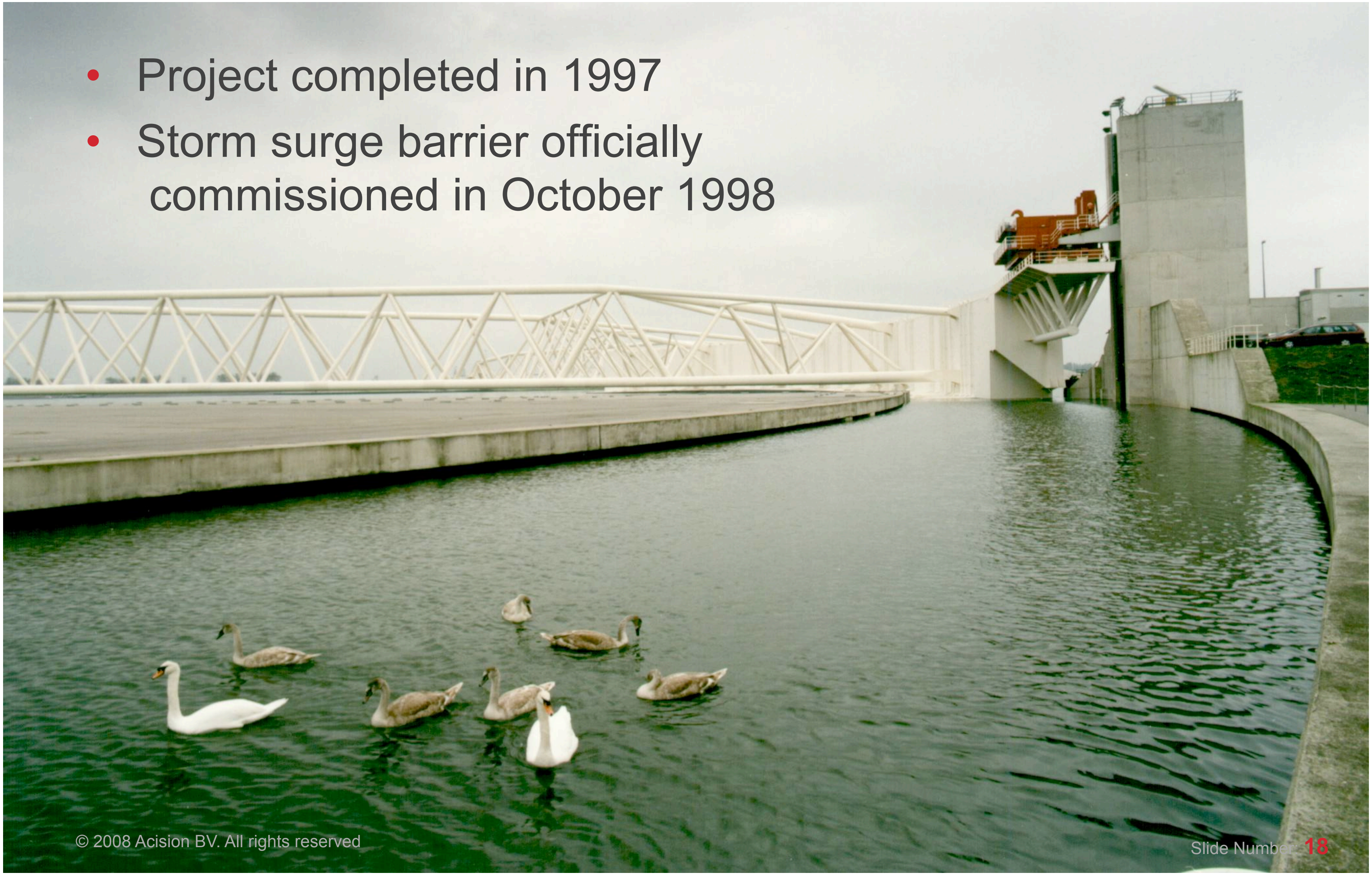


Use of formal methods - 2

- Modeling of data and algorithms using Z
  - Case tool for modeling BOS system using Ward & Mellor
  - Functionality and data in each process, store and flow modeled using Z
  - Design documentation generated from case tool using scripting and LaTeX
  - Input to Z Type Checker generated from case tool using scripting and syntactically validated
- Experiences with Z modeling
  - Difficult to learn, very steep learning curve
  - Excellent input to testers and reviewers who are much more effective in deriving test cases or reviewing code/design
  - Supports unambiguous communication between designer, programmer, tester and code reviewer

Delivery and operation

- Project completed in 1997
- Storm surge barrier officially commissioned in October 1998



Barrier reliability revisited

- 2006: concerns raised on reliability of the barrier
- Two reliability studies by independent parties performed for government
- Main conclusion
  - Pro-active maintenance critical for reliability
  - Availability of spare parts
  - Guaranteed repair times
  - Well-defined contracts and processes for operation, maintenance and repair

- Impact on BOS
  - Stricter repair times on specific hardware components

Results from actual operation

- Test closure every year since 1997
- First closure with an actual storm on November 11th, 2007
- No failures
- Software quality
  - No critical or major errors found that might affect barrier operation
  - Majority of changes requested on UI
  - Input validation was introduced

Lessons Learned (1)

- Operator/engineer is paged whenever some part is in error condition
  - In practice there is always something in error (though not fatal)
  - Most errors originate between 9:00 and 17:00 hrs
  - No errors between Christmas and New-Year!
- Do not under-estimate effect of human interactions such as maintenance
  - Repair on pumps and valves
  - Disconnected cables
  - Much more construction maintenance than anticipated in software design

Lessons learned (2)

- Very strict development/change process needed, but causing long cycles
  - Storm season October to April
  - Yearly trial in September (date set a year ahead)
  - Acceptance test consists of running 20 real storms on the test system (~60 days)
  - New release has to be ready for test in June
  - Normally not feasible => wait for next year

- Most changes requested in human interaction: GUI

- Extensive self-verification during start-up takes 2,5 hours
  - Not considered important: only started once a year
  - But… nightmare for test system

Mid-life upgrade project

- Hardware is end of life
- Port to new platform
- Methods and techniques from the original project still apply
- Improved error diagnostics and drill -down functionality
- GUI taken out of the core system
- Currently under development

Experiences upgrade project

- Use of Z from original project is still effective
  - Tricks required to make tooling work
  - Steep learning curve due to new development team
  - Formal methods missing in software engineering education
- Formal methods augment and improve existing techniques, especially the combination of
  - Formal specification
  - Module testing
  - Code review
- Experience is difficult to retain organizationally
  - People move on in their career
  - Amount of projects applying formal methods is low

Current status

- Logica
  - Few customers are willing to pay the price of a SIL4 project
    - Required reliability reduced by conventional design techniques
  - Learning curve for formal methods is still steep
  - Cooperating with University of Twente in formal methods research
  - Cooperating with Verum in industrializing formal methods
- Acision
  - Experience with storm surge barrier re-used in Telecom products
  - Formal specification badly needed in telecommunications protocols
    - Internet RFCs and 3GPP specifications lack formality
    - Set back from the more rigorous SDL notation used in ETSI
  - Cooperating with Technical University Eindhoven on formal architecture verification

A look to the future: what do we need most

- Support for the specification and design phase.
  - Majority of the problems are introduced in the specification and design, not the implementation.
  - External systems need to be part of formal specification
- Support for practical methods and tooling that make the use of formal methods simple
  - Notation and tooling need to be integrated in reqs/design tooling to support engineers
  - Promising developments in this area
- Standardize on specific formal methods (best of breed) as part of the mandatory computer science education.
  - Learning how to specify is critical engineering knowledge
  - Even if people have encountered formal methods, there are many proprietary variations (treated as religion)

Questions?

logica Acision
Innovation Assured