Simulation, Analysis, and Validation of Computational Models

— Verification and Validation —

Lecturer: Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

Overview

We started (lect. 6, 2nd part) to learn about systems engineering. Today's lecture about Verification and Validation is a continuation of this topic.

- Testing
- Verification
- Validation
- Risk

Short summary: Validation is similar to verification, except that verification checks before, and validation during actual use.

What is Systems Engineering?

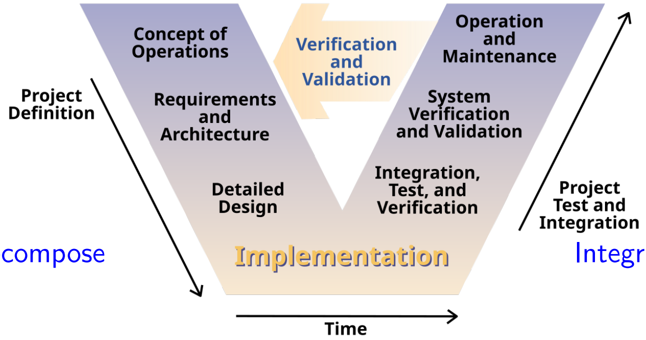
- “Systems Engineering represents a method, but also a mindset which enables the design, construction, operations and testing of a system in a successful way if done correctly.”
- “It isn’t just about the design or the pieces that go into it, it is a process that lasts the entire life of the product.”
- “To exemplify in real life, system engineering is like cooking a delicious cake. The ingredients are either made by you or manufactured by a producer.”
- “Systems engineering is the optimal process for bringing an idea to reality with a combination of disciplines that work in concert — just like all the instruments and vocals in [a] band combine to make beautiful music.”

Haberfellner e.a. (2019) Systems engineering. Springer.
Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 7)

V-model (see lect. 6, 2nd part)

Vision

Operation


Concept of Operations
Requirements and Architecture
Detailed Design
Implementation
Verification and Validation
Operation and Maintenance
System Verification and Validation
Integration, Test, and Verification
Project Definition
Project Test and Integration
Time

Decompose

Integrate

Details

L. Osborne e. a. (2005) Clarus Concept of Operations. FHWA-JPO-05-072, (Redrawn by Slashme).

Testing

Consider a software system:

- Functional testing
- Complexity testing
- Offline vs. online testing

Same applies for models of any real processes:

- Test: Verification
- RW test: Validation

In both cases, final tests are conducted with customers and stakeholders:

- Meets specification and expectations
- Valid for purpose as expected

Testing caveats

- Testing is critical, but expensive: Time, test rig or setup, sensors, data acquisition etc.
- Testing of components: Trust parts and libraries and vendors of components or retest everything?
- Calibration of sensors, procedures, and equipment
- "Testing the test": To what extent do test conditions and benchmarks reflect actual operational usage or RW conditions
- Simulated tests use *dummy* components if real ones are not available, but this may not be representative
- Failures often occur outside any test scenarios

Haberfellner e.a. (2019) Systems engineering. Springer.
Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Verification vs. Validation

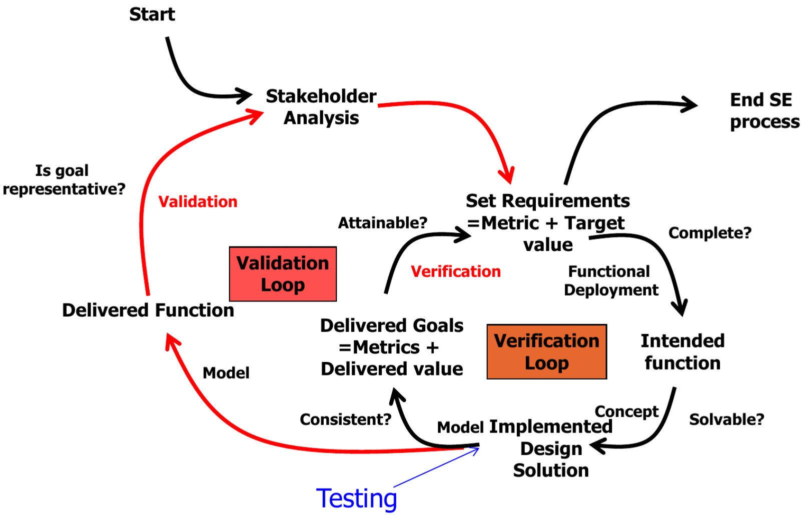
Verification (Tests): Is the function realised correctly?

- Modelling
- Simulations
- Alternative calculations
- Comparison with other proven designs
- Experiments
- Specialist technical reviews
- Mainly on components and subsystems

Validation (Case studies): Is the correct function realised?

- Meets needs and requirements of its intended users in the intended use environment
- In real environment (may also include simulations)
- Full system according to use cases by stakeholder
- Determines the parameters for verification across development cycles

Validation vs. Verification loops



Haberfellner e.a. (2019) Systems engineering. Springer.
Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Volkswagen emissions scandal

Can verification and validation get mixed up in practice?

- US Environmental Protection Agency (EPA) found in 2015 that Volkswagen had intentionally programmed TDI diesel engines ("Clean Diesel") to activate emissions controls only during laboratory emissions testing (verification).
- However, the vehicles emitted up to 40 times more $\mathrm{NO_x}$ in real-world driving (validation) than in testing.
- Volkswagen had chosen already around 2006 to program the Engine Control Unit (ECU) to switch from lower fuel consumption and high $\mathrm{NO_x}$ emissions to low-emission compliant mode when it detected an emissions test.
- Software was deployed in $\approx$ 11 Million cars (2009 – 2015).
- Volkswagen's cost of the scandal $\approx$ 32 Billion USD.

https://en.wikipedia.org/wiki/Volkswagen_emissions_scandal

Chernobyl disaster

- Test to simulate cooling the reactor during an accident in blackout conditions.
- To get proper test results emergency core cooling system was disabled.
- Blackout occurred simultaneously with the rupture of a coolant pipe.
- Test carried out despite an accidental drop in reactor power.
- Design issue: Shutting down the reactor in those conditions results in a power surge.
- Reactor components ruptured, lost coolants, and the resulting steam explosions and meltdown destroyed the containment building.
- This was followed by a reactor core fire that spread radioactive contaminants across Europe.
- Casualties 30 - 50. Evacuation of more than 100,000 people.

https://en.wikipedia.org/wiki/Chernobyl_disaster

Testing large computational systems

- Testing can be the most time consuming design stage: Use two levels of description:
- Timing, power consumption to be checked at low level (easier than testing logical functionality)
- Hardware description language (HDL) to test functional correctness (“functional equivalence”) of all logic paths at high level. Relatively fast and easy to obtain good coverage
- Tools for verification
  - Logic synthesis: To ensure equivalence of high-level logic and circuit-level function
  - Timing tools on circuit level
  - Check of design to ensure that circuits can be realised
- More general: Function vs. structure testing (“opaque box” vs. “transparent box”)

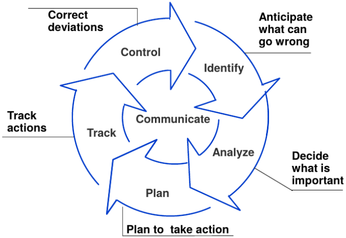
Neil H. E. Weste and David Money Harris (2011) CMOS VLSI Design 4e. Pearson.

Risk

- Risk is defined as the combination of:
  - Probability that a system will experience an (undesired) incident
  - Impact, consequences, or severity if the incident occurred (hazard)
- Includes technical or programmatic sources of problems (e.g. a cost overrun, schedule slippage, safety mishap, health problems, malicious activities, environmental impact, or failure to achieve a needed scientific or technological objective or success criterion)
- Risk management: proactively identifies, analyzes, plans, tracks, controls, communicates, documents risks

Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Risk Management Framework



How to identify risk?

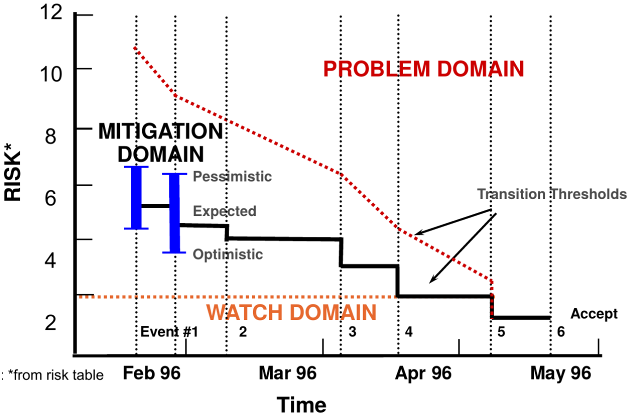
- Brainstorm
- Experience
- Regulations (scores)
- Ethics committee
- Stakeholders

Mitigation domain: Pessimistic, Expected, and Optimistic

Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Risk-reducing design process

Threshold Risk Metric (NASA)



Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Safety

- Component Failure Accidents
  - Single or multiple component failures
  - Usually assume random failure
- Component Interaction Accidents
  - Related to interactive complexity and tight coupling
  - Use of computers and software
  - Role of humans in systems

Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

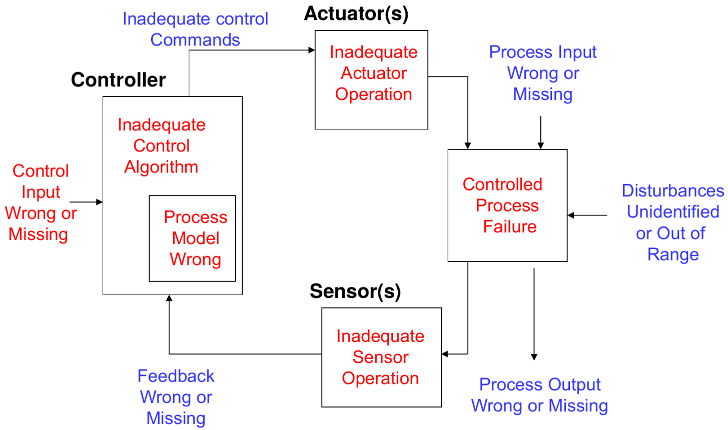
System-theoretic view of safety

- Safety is an emergent system property
- Based on systems theory rather than reliability theory
- Accidents arise from interactions among system components (human, physical, computational) that violate the constraints on safe component behavior and interactions
- Losses are the result of complex processes, not simply chains of failure events
- Expect component interaction accidents
- Most major accidents arise from a slow migration of the entire system toward a state of high-risk

Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Systems Theoretic Process Analysis (STPA)

How to find out about Component Interaction Accidents?



Inadequate control Commands

Actuator(s)

Inadequate Actuator Operation

Process Input Wrong or Missing

Controller

Inadequate Control Algorithm

Process Model Wrong

Control Input Wrong or Missing

Controlled Process Failure

Disturbances Unidentified or Out of Range

Sensor(s)

Inadequate Sensor Operation

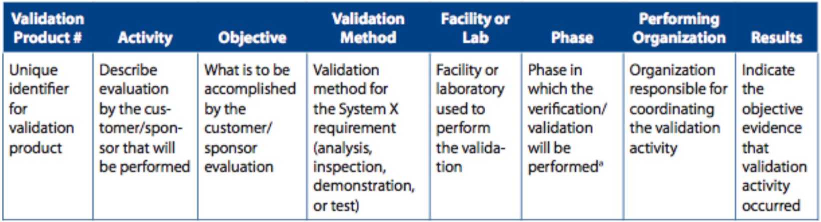
Feedback Wrong or Missing

Process Output Wrong or Missing

Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Validation

Formalised validation starts from verification and continues over the lifetime of the product:



Validation Product #	Activity	Objective	Validation Method	Facility or Lab	Phase	Performing Organization	Results
Unique identifier for validation product	Describe evaluation by the customer/sponsor that will be performed	What is to be accomplished by the customer/ sponsor evaluation	Validation method for the System X requirement (analysis, inspection, demonstration, or test)	Facility or laboratory used to perform the validation	Phase in which the verification/ validation will be performed[a]	Organization responsible for coordinating the validation activity	Indicate the objective evidence that validation activity occurred

[a] Phases: During selecting of service; prior to final selection, prior to full-scale fabrication/delivery, during component functional test, during system functional test, during end-to-end functional test, during operational test, during normal operation

Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Life monitoring

- Operation rules, qualification of operators
- Maintenance cycle and regular checks
- Recording and evaluation of minor failures
- Scenarios to deal with failures and with partial failures
- Life cycle of components
- Replacement systems and spares
- Options for reconfiguration, upgrades

Olivier de Weck (2015) Fundamentals of Systems Engineering (Lect. 9)

Complex Systems at NASA

- System Engineering of Complex Systems is not well understood
- System Engineering of Complex Systems remains challenging
  - System Engineering can produce elegant solutions in some instances
  - System Engineering can produce embarrassing failures in some instances
  - Within NASA, System Engineering is frequently unable to maintain complex system designs within budget, schedule, and performance constraints
- "How do we Fix System Engineering?" (Michael Griffin, 2010)
  - Successful practice in System Engineering is frequently based on the ability of the lead system engineer, rather than on the approach of system engineering in general
  - The rules and properties that govern complex systems are not well defined in order to define system elegance

Quoted from Michael D. Watson (2015) Complex Systems at NASA - Help from Natural Systems

Complex Systems at NASA: Help from Natural Systems (?)

- Complex engineered systems (star ship, submarine, computer)
- Complex Adaptive Systems in nature (cells, ecology, weather)
- Do complex systems follow the similar relationship rules?

Engineered systems	Natural systems
Big data in safe place	Some data in situ
Minimise complexity (parsimony principle)	Complexity can help against adversaries
Stability from rigidity	Stability from adaptivity
Single system (traditionally)	Population
Hierarchy (typically)	Dynamic patterns
Optimal (?)	Viable (filling niche)
Energy intense	Energy efficient
Highly optimised tolerance	Self-organised criticality

HOT (undesirable) and SOC (provides flexibility) are similar and a consequence of the complex interaction (in a fundamentally different situation).

Arguing against Michael D. Watson (2015) Complex Systems at NASA - Help from Natural Systems

Help from Natural Systems (!)

The differences do not imply that we cannot learn from nature, e.g.

- What observables, principles, governance (as studied in cybernetics)
- Safety against attacks, immune system/firewall
- Intra-system communication
- Applications such as
  - multi-agent systems
  - colonisation of other planets
  - marine applications or agriculture, forestry etc.

  may be more suitable for biologically-inspired approaches, i.e. depends on task
- Self-optimisation, -adaptation, -organisation

Learning effective dynamics

- More recently neural network approaches were considered for testing, prediction and novelty detection
- Multi-scale behaviour:
  - Recurrent networks for short term dynamics
  - LSTM for variable medium ranges
  - Feed-forward for long-term predictions by maps
- Versatile, but largely not explainable

(to be discussed later)

see e.g. P.R. Vlachas (2022) Learning and forecasting the effective dynamics of complex systems across scales (Doctoral dissertation, ETH Zurich).

Conclusion

- Verification and validation are the main connection between theory and practice
- We did not discuss today how to deal with any test data
- Tests in complex systems are a complex issue
- Risk and safety management need to include systems thinking

Next topics

We'll continue with additional topics on systems engineering, such as

- Confidence
- Explainability
- Scaling
- Complexity
- Heterogeneity