A systems engineering approach to design of complex systems

Sagar Behere

22 June 2015

Kungliga Tekniska Högskolan, Stockholm

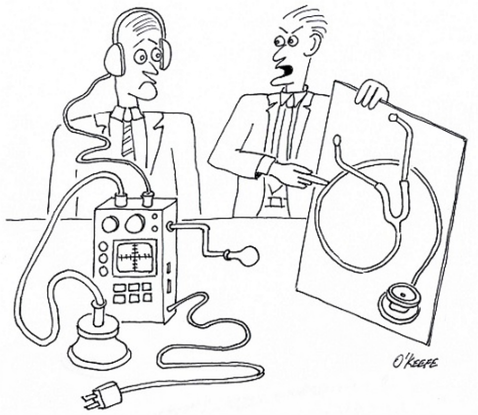
Question

How will you go about developing a quadrocopter?

Contents

1. Systems Engineering
2. Requirements
3. Architecture
4. Testing, Verification and Validation
5. Safety
6. ~~Model based systems engineering~~

What is Systems Engineering?



Interesting design, but this is more of what we had in mind.

A plethora of definitions ;-)

"...an interdisciplinary field of engineering that focuses on how to design and manage complex engineering projects over their life cycles."

[source: Wikipedia]

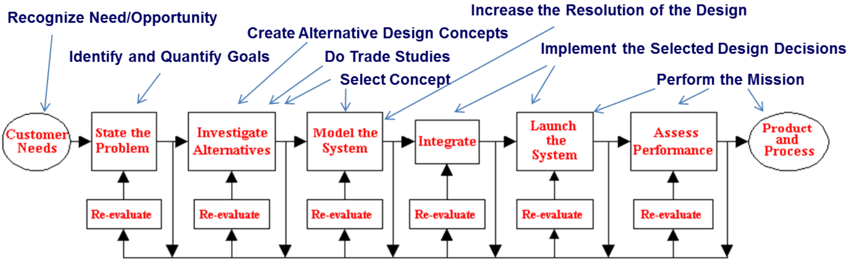
"...a robust approach to the design, creation, and operation of systems."

[source: NASA Systems Engineering Handbook, 1995]

"The Art and Science of creating effective systems..."

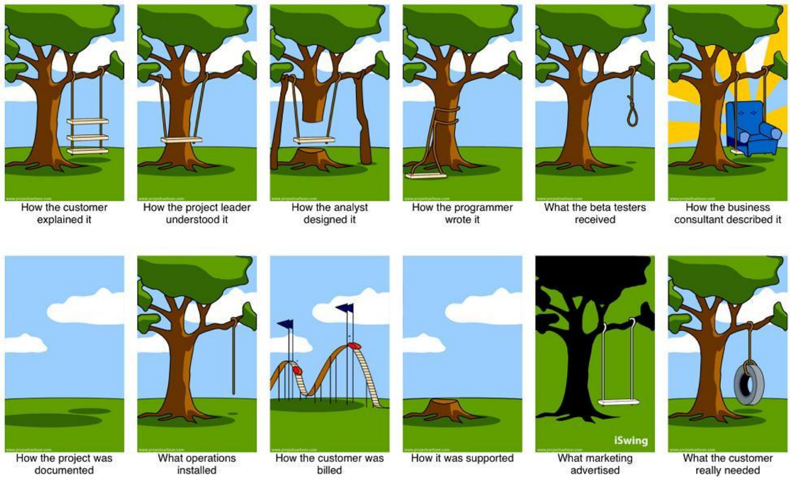
[source: Derek Hitchins, former President of INCOSE]

The systems engineering process

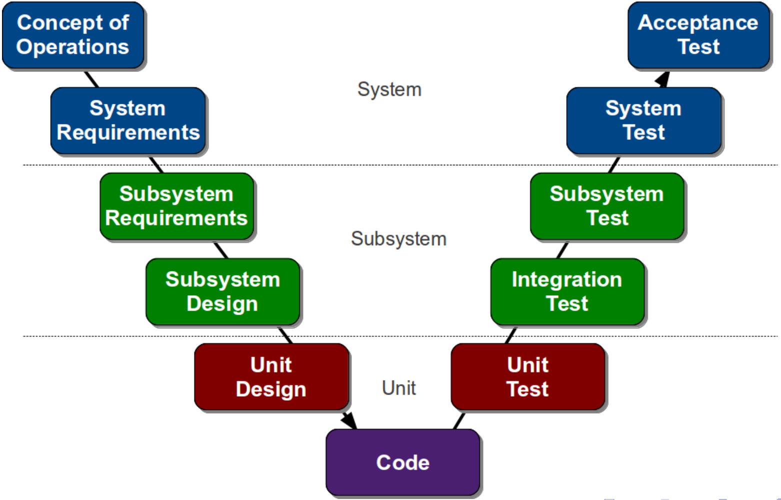


[source: A. T. Bahill and B. Gissing, Re-evaluating systems engineering concepts using systems thinking; (overlaid with the NASA approach in blue)]

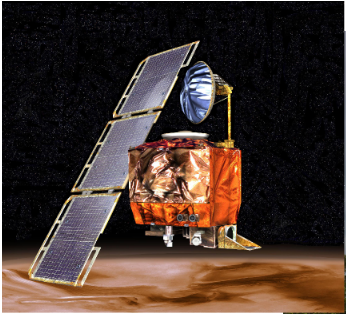
Why is a process needed?

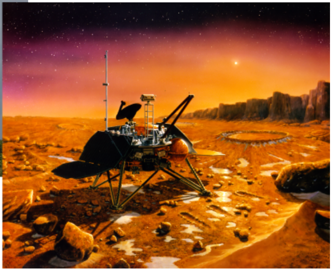


The V model

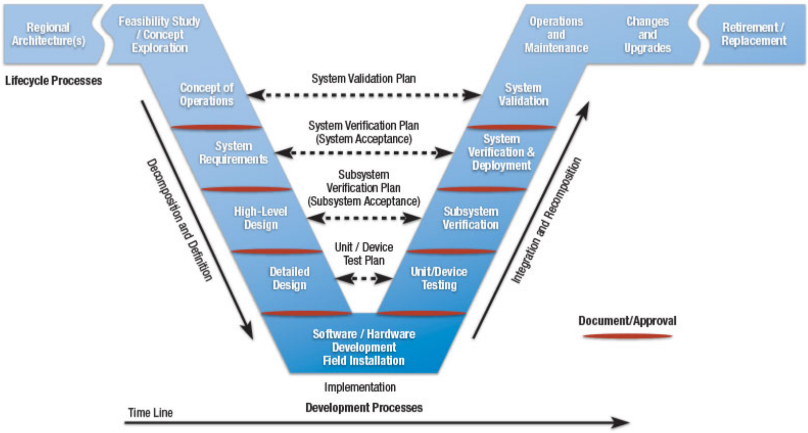

Concept of Operations
System Requirements
Subsystem Requirements
Subsystem Design
Unit Design
Code
Unit Test
Integration Test
Subsystem Test
System Test
Acceptance Test
System
Subsystem
Unit

Systems Engineering 'Epic fail'



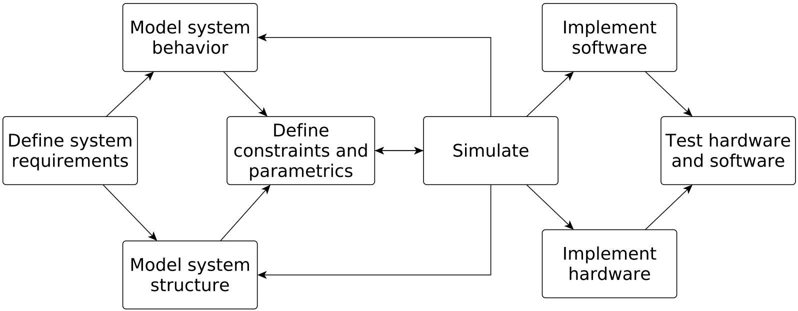


The V model



Regional Architecture(s)
Feasibility Study / Concept Exploration
Operations and Maintenance
Changes and Upgrades
Retirement / Replacement
Lifecycle Processes
Concept of Operations
System Validation Plan
System Validation
System Requirements
System Verification Plan (System Acceptance)
System Verification & Deployment
High-Level Design
Subsystem Verification Plan (Subsystem Acceptance)
Subsystem Verification
Detailed Design
Unit / Device Test Plan
Unit/Device Testing
Software / Hardware Development Field Installation
Decomposition and Definition
Integration and Recomposition
Document/Approval
Implementation
Development Processes
Time Line

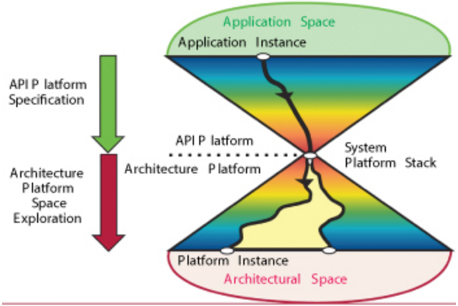
Simplified processes



[source: ICONIX process for embedded systems]

Beyond development processes

- The lifecycle perspective
  - Deployment, operations, management, retirement
- Design approaches
  - Top down, bottom up, inside out, platform based

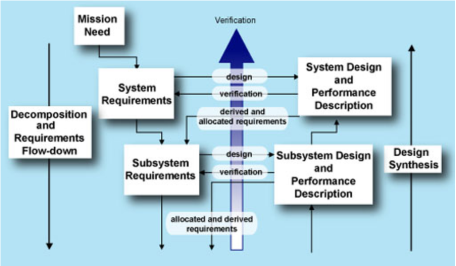


Some resources

- INCOSE
- Embedded Systems Development Using SysML (available online)
  - Short and sweet book
- Systems Engineering with SysML/UML: Modeling, Analysis, Design - Tim Weilkiens
  - OMG Press
- NASA Systems Engineering Handbook (available online)
- Overview of the System Engineering process (available online)
- Tool tutorials

Takeaway: Systems engineering

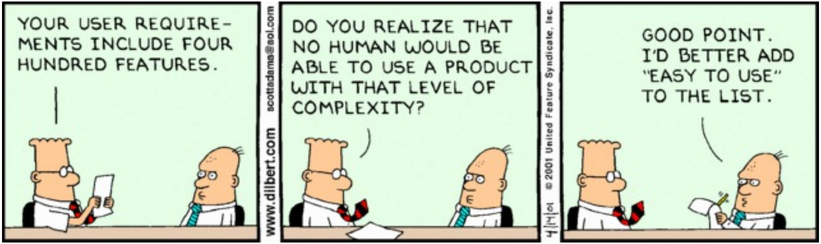
- Consider the product as a whole within its 'Concept of Operation'
- Systematic processes exist to guide you
- There is a community devoted to Systems Engineering
- Software and tools exist to make your task simpler



Contents

1. Systems Engineering ✓
2. Requirements
3. Architecture
4. Testing, Verification and Validation
5. Safety
6. Model based systems engineering

Requirements Engineering



The science and discipline of analyzing, documenting, validating and tracing requirements.

What is a requirement?

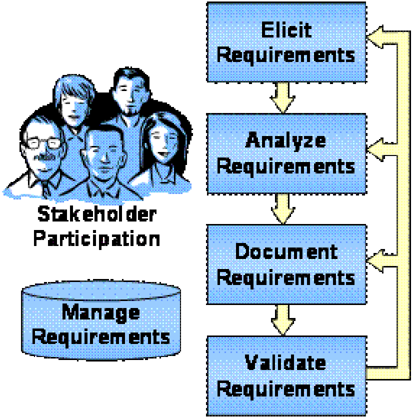
1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in 1 or 2

[source: IEEE-Std-610.12-1990]

Characteristics of good requirements

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable

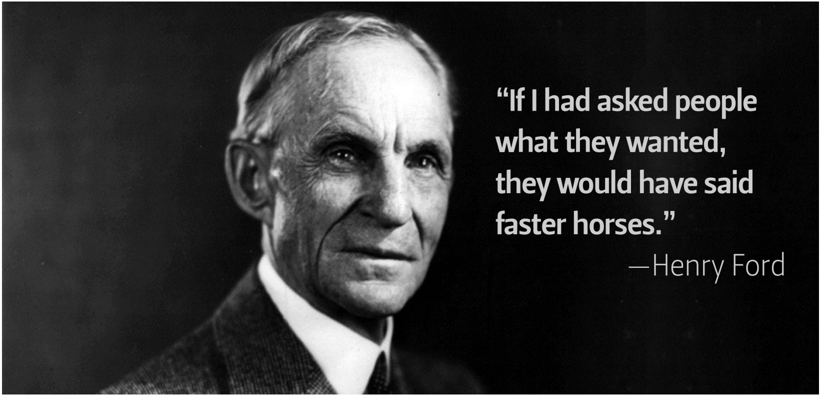
Elements of requirements engineering



Requirements elicitation

- Interviews and questionnaires
- Concept of operation, use cases and models of usage
- Examination of documentation
  - Standards
  - Systems manuals
  - Statement of requirements
- Prototyping
- Conversation and interaction analysis
- Ethnographic studies

...but remember



Requirements analysis - Structuring

- System Behavior $\leftarrow$ User perspective
- Functional
  - Logical and implementation specific
- Extra-functional

Requirements drill-down depends on architecture!
(We will come back to this)

Requirements documentation and validation

- Traditional approach: Documents
- Newer approach: Model based
    - Documents can be auto-generated
- "Each requirement must have an associated test case!"

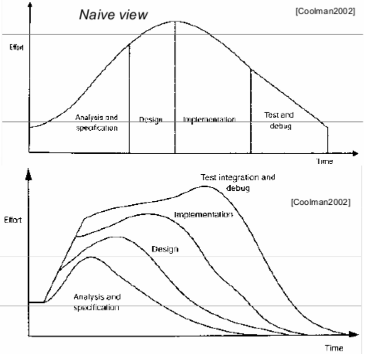
More on this later..

RE effort distribution

University of Paderborn
Software Engineering Group

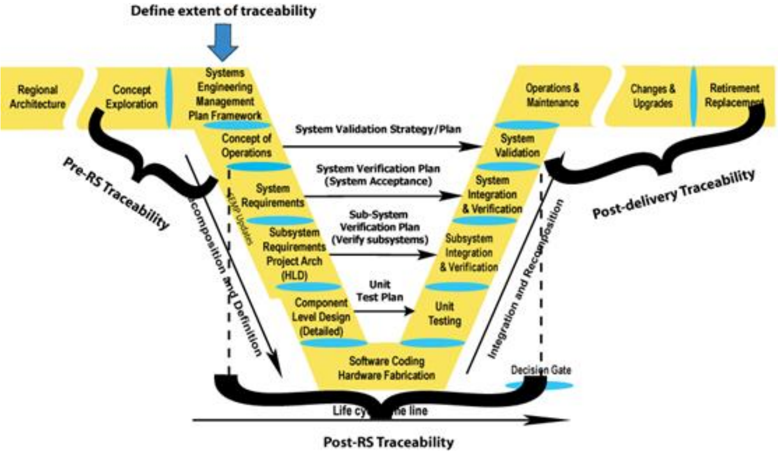
Effort Distribution for Complex Systems

- For complex systems due to allocation and flowdown the requirement engineering **continues** during the design (and implementation) phase



Holger Giese WS05/06 – Software Engineering for Software-Intensive Systems – IV Requirements IV-27

Requirements traceability



Some resources

- Requirements engineering: a good practice guide - Ian Sommerville, Pete Sawyer
  - John Wiley & Sons, 1997
- Requirements Engineering: Fundamentals, Principles, and Techniques - Klaus Pohl
  - Springer 2010
- Tool tutorials

Takeaway: Requirements

"As many problems are caused by improperly chosen requirements as by incorrect implementations."

- Continuous part of product development
- Elicit, analyze, document and verify
- Ensure requirement traceability

But remember: There's more to a good product than merely fulfilling all requirements!

Contents

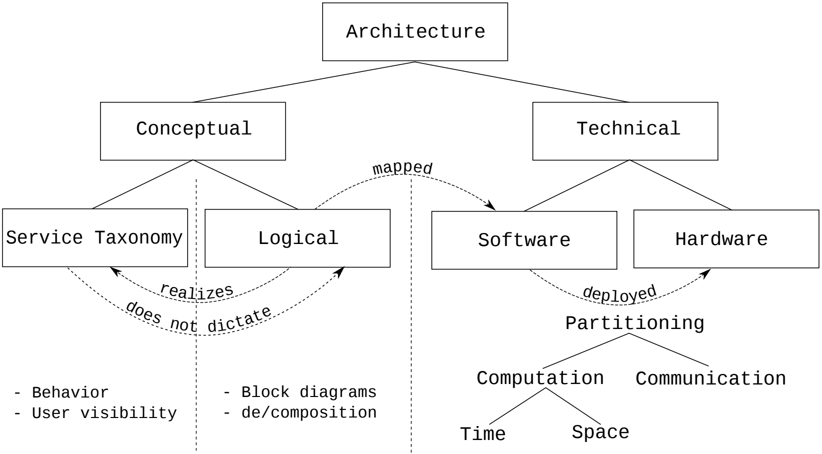
1. Systems Engineering ✓
2. Requirements ✓
3. Architecture
4. Testing, Verification and Validation
5. Safety
6. Model based systems engineering

What is architecture

"..fundamental concepts or properties of a system in its environment, embodied in its elements, relationships and principles of its design and evolution." [ISO42010:2011]

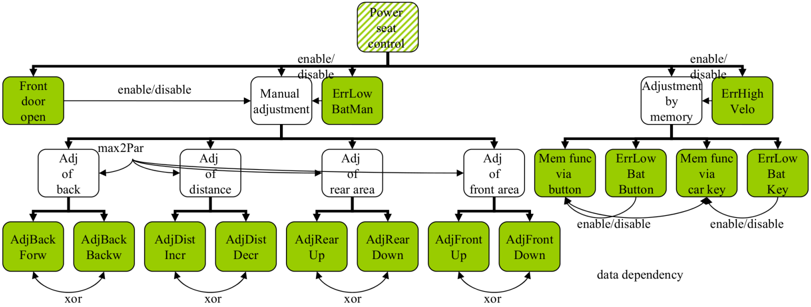
But remember: The map is not the territory!

Architecture types



Conceptual/Service taxonomy

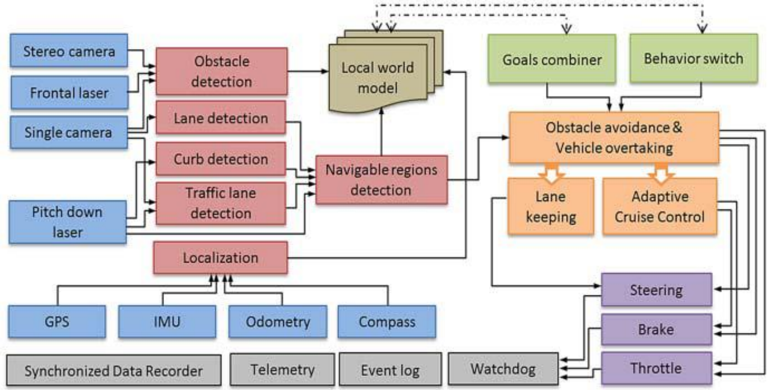
Hierarchy of Services/Features as seen by the **user** of the system.



(source: S. Rittman, A methodology for modeling usage behavior of multi-functional systems)

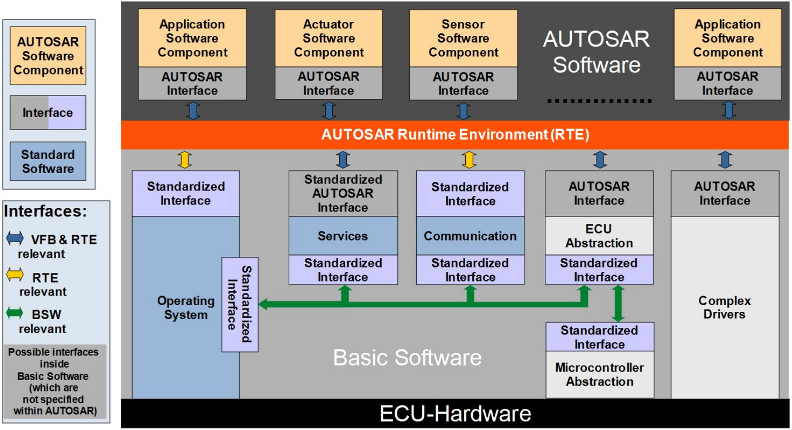
Conceptual/Logical architecture

Block diagram of the system



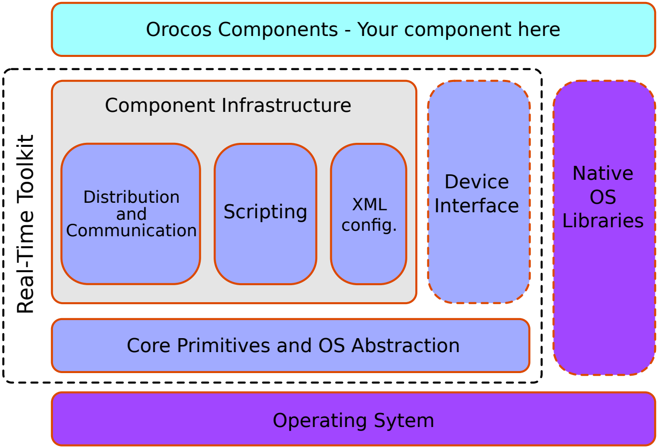
(source: Fernandes L. et al, Intelligent robotic car for autonomous navigation)

Technical/Software



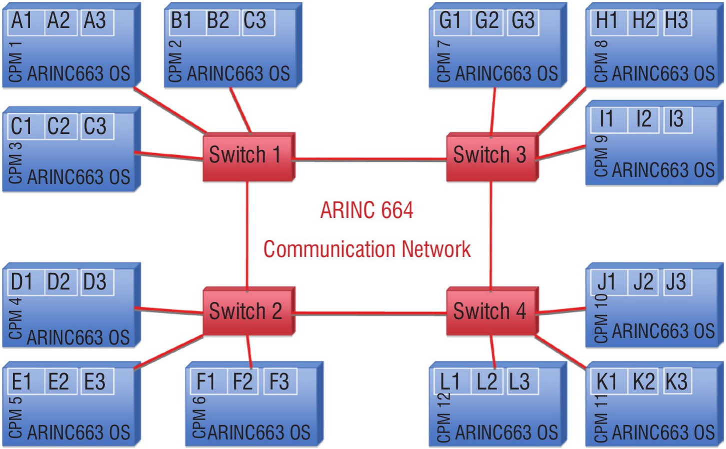
(source: AUTOSAR.org)

Technical/Software



(source: OROCOS.org)

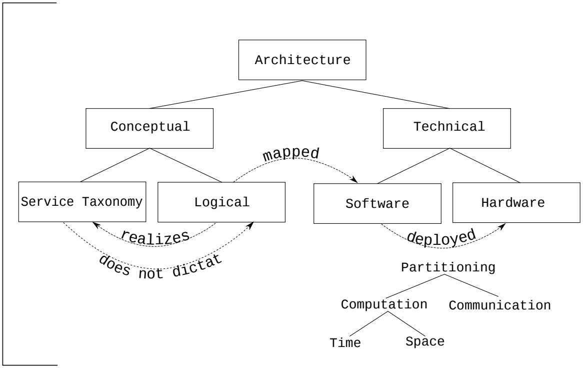
Technical/Hardware



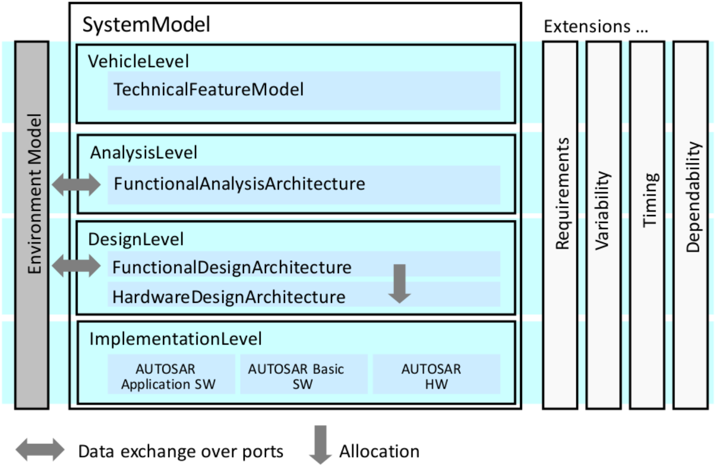
(source: Biber P. et al, New challenges for future avionic architectures)

Extra functional properties

Safety
Reliability
Robustness
Predictability
...



Architecture description languages

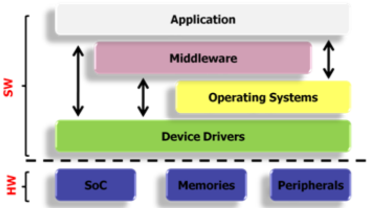


Some resources

- For software - The Architecture Of Open Source Applications (book, readable online)
  - http://aosabook.org
- For architecture description - Model-Based Engineering with AADL (book)
- http://www.aadl.info , http://east-adl.info

Takeaway: Architecture

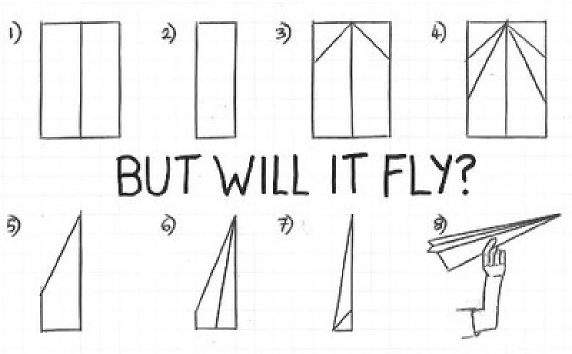
- Architecture is represented with a hierarchy of abstractions
  - Example: Service Taxonomy, logical, software, hardware
- Elements higher in the hierarchy are allocated to (realized by) elements lower in the hierarchy
- An architecture **description** aids in understanding, communication and formal analysis



Contents

1. Systems Engineering ✓
2. Requirements ✓
3. Architecture ✓
4. Testing, Verification and Validation
5. Safety
6. Model based systems engineering

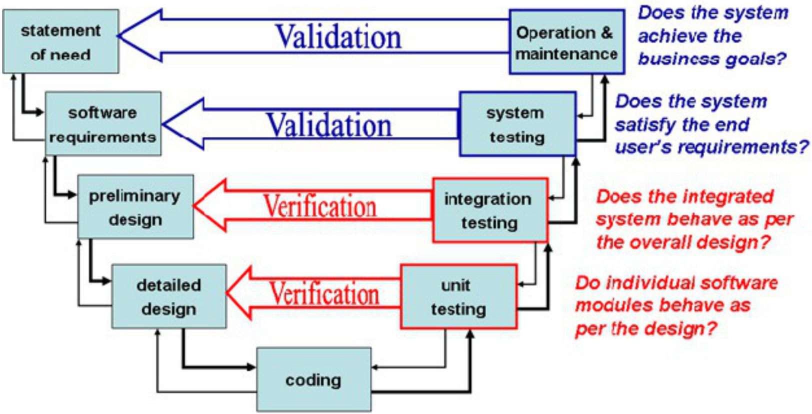
Testing, Verification and Validation



Verification and Validation

- Verification establishes the truth of correspondence between a work product and its specification
  - From Latin **veritas** (=truth)
  - Are we building the product right?
  - Does product meet all documented requirements?
- Validation establishes the fitness of the product for its operational mission a.k.a user needs
  - From Latin **valere** (=to be worth)
  - Are we building the right product?
  - Do specifications correctly describe a system useful for intended purpose?

Verification and Validation



Example

- Requirement: Reverse thrust and spoilers shall not operate mid-air
  - Shall be active only in landing situation
- Domain Properties
  - Wheel pulses are on IFF wheels are turning
  - Wheels are turning IFF the plane is moving on the runway
  - A plane on ground has $>>$ 6.3 tons on each main landing gear strut
- System specification:
  - RT only if S1 = True. Spoilers if S1 || S2
  - (S1) Weight of at least 6.3 tons on each main landing gear strut
  - (S2) Wheels of the plane turning faster than 72 knots

What could go wrong?

(Lufthansa flight LH2904)

V&V techniques

- Simple checks
    - Traceability, well-written requirements
- Prototyping
- Functional test design
- User manual development
- Reviews and inspections
    - Walkthroughs
    - Formal inspections
    - Checklists
- Model-Based V&V

Static and Dynamic analysis

- Static analysis
  - Evaluates static criteria: Properties of the system at rest
  - Looks for **faults**
  - Considers documentation for requirements, specification, analysis of source code etc.
- Dynamic analysis
  - Evaluates dynamic criteria: Properties that only manifest in operating system
  - Looks for **failures**
  - Impiles executing the system, injecting faults, etc.

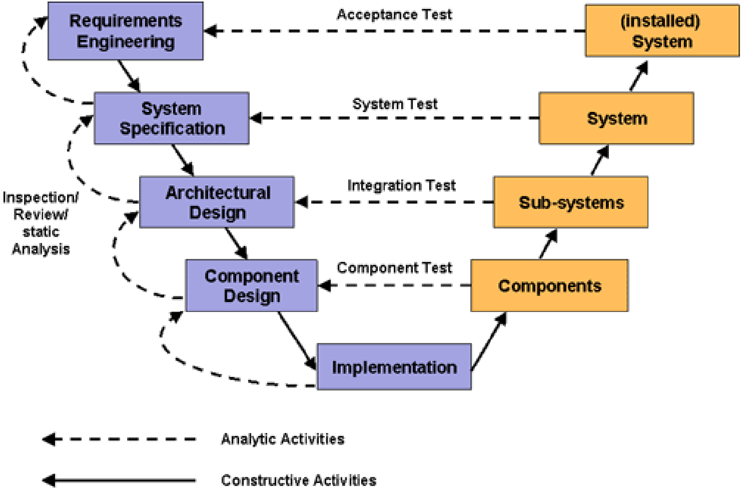
Basic V&V approaches

- Testing (dynamic analysis, dynamic testing)
  - assessing through execution with selected stimuli (test data) on "real" environment
  - Simulation, diagnostics - other forms of testing
    - Hardware-in-the-loop, Model-in-the-loop
- Analysis
  - investigation of properties of a product without running it
  - Code verification, reviews, model checking, etc . – other forms of static analysis

Some validation types

- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Testing


Requirements Engineering
Acceptance Test
(installed) System
System Specification
System Test
System
Architectural Design
Integration Test
Sub-systems
Inspection/ Review/ static Analysis
Component Design
Component Test
Components
Implementation
Analytic Activities
Constructive Activities

Test case design

- Output is test cases, not faults!
- Describes **what** needs to be done for a particular V&V effort, and **how** it is done

Requirements to be controlled
e.g., functions, timing, performance, robustness..

V&V targets -The hardware, software, or functional component/system under test.

V&V procedure
1. Tasks to be performed.
2. The setup of test environment
3. The stimuli (selected test data)
4. The expected outcome

V&V log
- Documentation of test executions, e.g., the actual outcomes of test procedures, time of execution, responsible persons..

Testing processes

- Common observation: testing process boils down to executing the system
    - as many times as deemed necessary (or as often as there is time to)
    - testing with randomly selected inputs
- Neither the generated test cases, nor the stop testing condition are ever reported.
- No proper estimate of the necessary time and resources to run the tests
- As always, proper processess need to be defined!

Example testing process

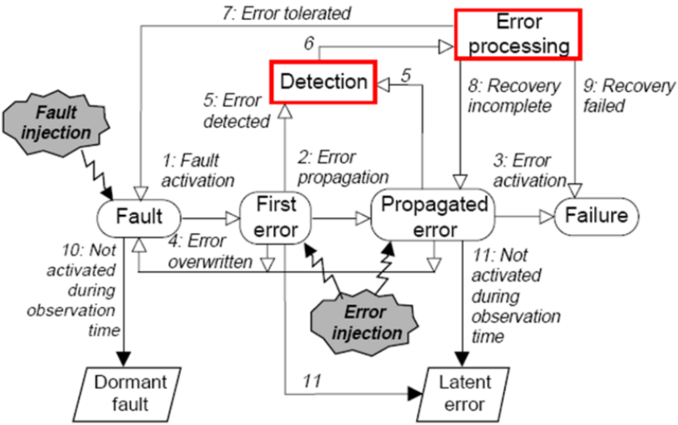
1. Test planning
2. Test design
3. Test case specification
4. Test procedure definition
5. Test procedure execution
6. Analysis of results

[source: ESA software engineering standards PSS-05-0]

Testing techniques

- Black box testing (also called Functional testing)
  - Equivalence classes and input partition testing
  - Boundary value analysis
  - Error guessing
- White box testing (also called Structure based testing)
  - Control flow analysis
  - Data flow analysis
  - Cause consequence diagram
- Other techniques

Fault and error injection



[source: Christmansson, J.; Hiller, M.; Rimen, M. An experimental comparison of fault and error injection]

How much is enough?

- 100% coverage is not feasible for complex systems
- Often V&V is costliest part of system development
  - >50% of development cost in aviation
- Choice of test cases is of utmost importance
- V&V effort commensurate with size and criticality of system
  - Is system safety critical?
  - Is target technology immature or high risk?
  - Will the business tolerate a high risk project?

Some resources

- RTO-TR-IST-027 - Validation, Verification and Certification of Embedded Systems
  - NATO technical report (available online)
- ESA software engineering standards (available online)
- ESA ECSS standards
  - ECSS-E-HB-40A software engineering handbook (available online)
- IEEE Std 1059-1993: IEEE Guide for Software Verification and Validation Plans

Takeaway: Verification and Validation

- V&V thinking should be incorporated throughout the systems engineering process
- A V&V **template** should be developed early in the project
- Acceptable testing coverage needs to be determined
- Test cases need to be designed keeping system requirements in mind (like certification)
- V&V is costly!

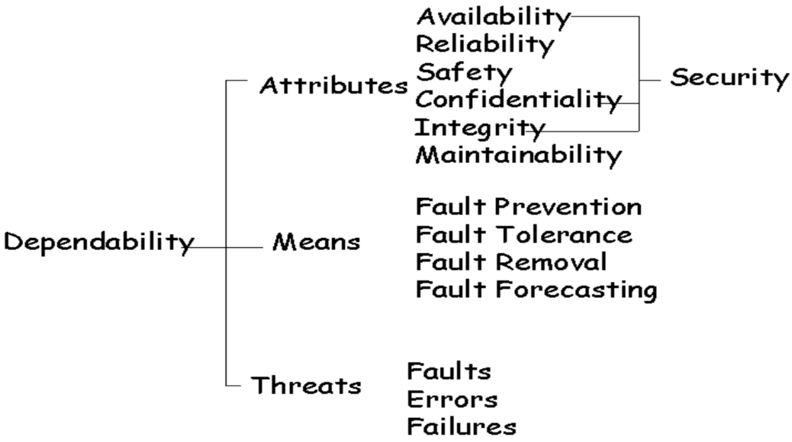
"Testing can be a very effective way to show the presence of faults, but it is hopelessly inadequate for showing their absence" – Dijkstra, 1972

Contents

1. Systems Engineering ✓
2. Requirements ✓
3. Architecture ✓
4. Testing, Verification and Validation ✓
5. Safety
6. Model based systems engineering

Safety

Context



[source: J.C. Laprie]

Definitions

Safety Freedom from unacceptable risk (to life, property, ...)

Risk An expression of the future impact of an undesired event in terms of event **likelihood** and event **severity**
$Risk = Probability * Consequence$

Hazard Present condition, event, or circumstance that could lead to or contribute to an unplanned or undesired event

Critical systems

- Safety critical
  - Failure may injure or kill people, damage the environment
  - Example: nuclear and chemical plants, aircraft
- Business critical
  - Failure may cause severe financial loss
  - Example: information system. Customer information should not be lost
- Mission critical
  - Failure may cause a mission to fail
  - Large values potentially wasted
  - Example: Space probe. Large sums of money, many years of waiting

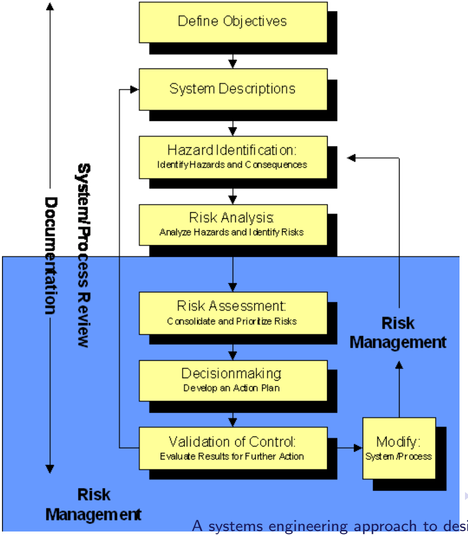
Scope of safety engineering

Functional safety Absence of unreasonable risk due to hazards caused by malfunctioning behavior of the system
- Correct functioning of system in response to inputs

System safety Absence of unacceptable risk due to:
- Errors related to Functional Safety
- Hazardous materials
- Hazardous environments
- Hazards related to energy sources

Safety is a system level property/concern!

System safety process



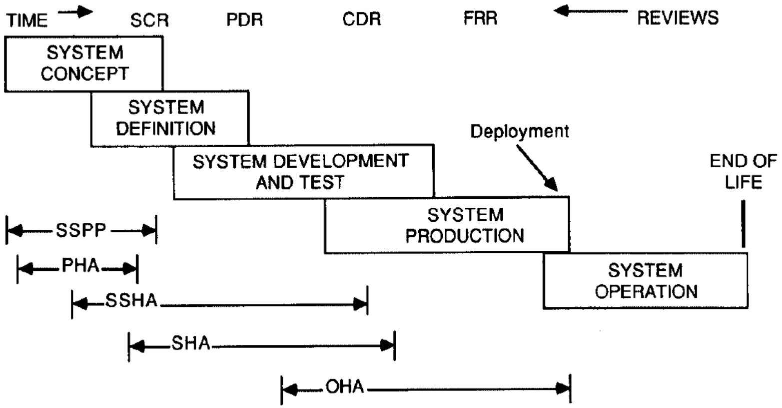
System safety techniques

- Preliminary hazard analysis
- System hazard analysis
- Subsystem hazard analysis
- Operating and support hazard analysis
- Fault hazard analysis
- Failure Mode and Effects Analysis (FMEA)
- Fault Tree Analysis (FTA)
- Software hazard analysis
- Sneak circuit analysis
- Simultaneous Timed Events Plotting Analysis (STEP)
- Hazard totem pole
- Management Oversight and Risk Tree (MORT)

System safety products

- System Safety Program Plan (SSPP)
- Preliminary Hazard Analysis (PHA)
- Subsystem Hazard Analysis (SSHA)
- System Hazard Analysis (SHA)
- Operating Hazard Analysis (OHA)

System safety products in lifecycle



Exemplary numbers

- 10^{-6} per hour - "ultra-reliable" [source: Parnas]
  - 10^6 hours $\approx$ 114 years
- 10^{-9} , 10^{-7} failures per hour (or flight) [source: Leveson]
  - Mishap is not expected to happen during the lifetime of the whole fleet (of a certain airplane)

[sources:

1. Courtois and Parnas, " Documentation for Safety Critical Software", IEEE 1993
2. Leveson, "Software Safety: Why, What, and How", ACM Computing Surveys, 1986
3. Leveson, Safeware , Addison-Wesley, 1995]

Risk assessment

Exact method differs for each standard...

RISK ASSESSMENT MATRIX						
		Probability				
Severity		Frequent A	Likely B	Occasional C	Seldom D	Unlikely E
Catastrophic	I	E	E	H	H	M
Critical	II	E	H	H	M	L
Marginal	III	H	M	M	L	L
Negligible	IV	M	L	L	L	L
E – Extremely High		H – High		M – Moderate		L – Low

Risk management

- **Reject** - Risk outweighs benefits
- **Avoid** - Go around, do it differently
- **Delay** - Maybe it'll go away?
- **Transfer** - Pass on to someone else (insurance?)
- **Spread** - dilute the impact
- **Compensate** - Design parallel and redundant systems

Reduce - Decrease probability

Safety Integrity Level (SIL)

Automotive example

CONTROLLABILITY:
Ability to **avoid a specific harm** or damage through the timely reactions of the persons involved

SEVERITY:
Estimate of the extent of harm to one or more individuals that can occur in a potentially hazardous situation

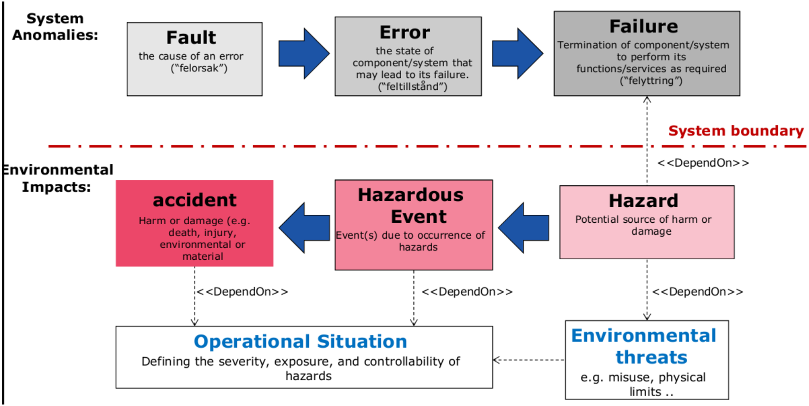
EXPOSURE:
State of being in an operational situation can be hazardous if coincident with the failure mode under analysis

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

SIL numbers - IEC/EN 61508

Probability of failure		
Safety Integrity Level (SIL)	**Mode of operation – on demand** (average probability of failure to perform its design function upon demand)	**Mode of operation – continous** (probability of dangerous failure per hour)
4	$\geq 10^{-5}$ to $< 10^{-4}$	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-4}$ to $< 10^{-3}$	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-3}$ to $< 10^{-2}$	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-2}$ to $< 10^{-1}$	$\geq 10^{-6}$ to $< 10^{-5}$

Accident causality



Fault categorization

- Endogenous - arise from within the system itself
  - Incorrectly functioning subsystems
  - Feature Interaction between correctly functioning subsystems
- Exogenous - arise from physical/logical/human environment
  - Uncertainties - inadequate perception
  - Contingencies - uncontrollable, unforeseen, ...

(source: Baudin et. al, Independent safety systems for autonomy)

Alternatively,

- Systematic - mistakes in development, maintainence, reuse
- Operational - unintended system usage

Fail safe and Fail operational

- Fail safe
  - Safe state can be reached upon system failure
  - Example: An automatic landing system is **fail safe** if, in the event of a failure, there is no significant out-of-trim condition or deviation of flight path or attitude - but the landing is not completed automatically.
- Fail operational
  - No safe state can be reached, minimum level of service expected
  - Example: An automatic landing system is **fail operational** if, in the event of a failure, the approach, flare and landing can be completed by the remaining part of the automatic system.

Example: CAT IIIB Autoland

Testing, verification and validation

Two main approaches

- Show that a fault cannot occur
- Show that if a fault occurs, it is not dangerous

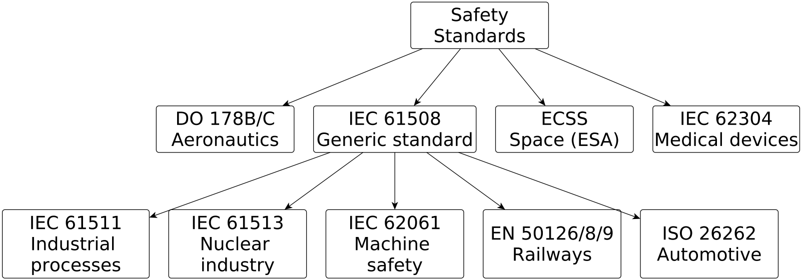
BUT

- Testing, V&V can show consistency only within the requirements specified
- Requirements build on assumptions
- What about **failures of imagination?**

Challenges to functional safety

- Incorrect specifications of the system, hardware or software
- Omissions in the safety requirements specification (e.g. failure to develop all relevant safety functions during different modes of operation)
- Random hardware failure mechanisms
- Systematic hardware failure mechanisms
- Software errors
- Common cause failures
- Human error
- Environmental influences (e.g. electromagnetic, temperature, mechanical phenomena)
- Supply system voltage disturbances (e.g. loss of supply, reduced voltages, re-connection of supply).

Functional safety standards



Some resources

- Nancy Leveson
- "System Safety for the 21st Century" - Richard A. Stephans
- Ariane 5 Flight 501 Failure - Full report (available online)
- Air crash investigations (TV series)

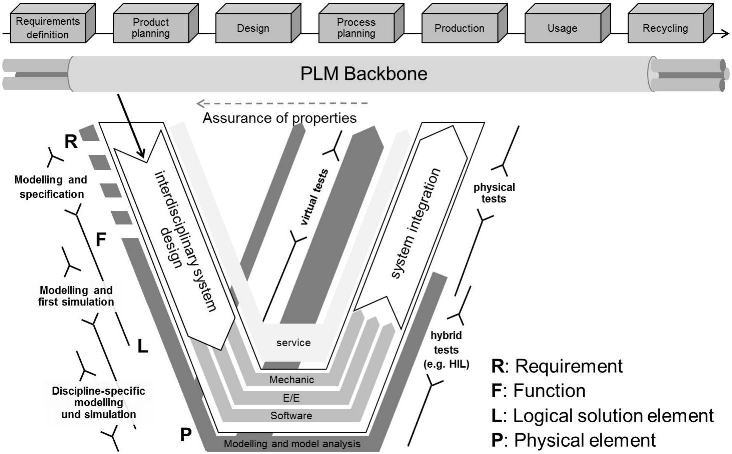
Takeaway: Safety

- Safety is freedom from unacceptable risk
- Can be thought of in terms of System Safety and Functional Safety
- Many techniques to assess and analyse system and functional safety
- Critical systems need to be designed to applicable SILs
- Many safety and certification standards exist
    - Know what is applicable to your product,

Contents

1. Systems Engineering ✓
2. Requirements ✓
3. Architecture ✓
4. Testing, Verification and Validation ✓
5. Safety ✓
6. Model based systems engineering

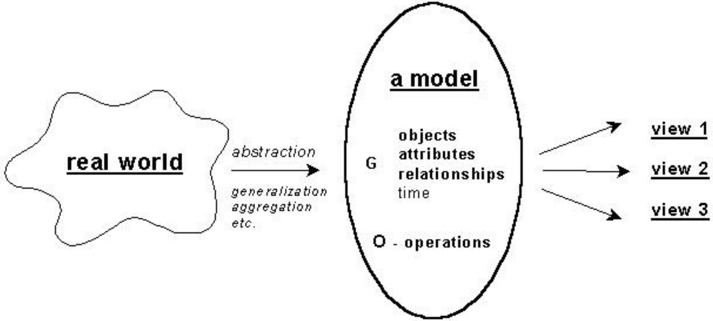
Model Based Systems Engineering



R: Requirement
F: Function
L: Logical solution element
P: Physical element

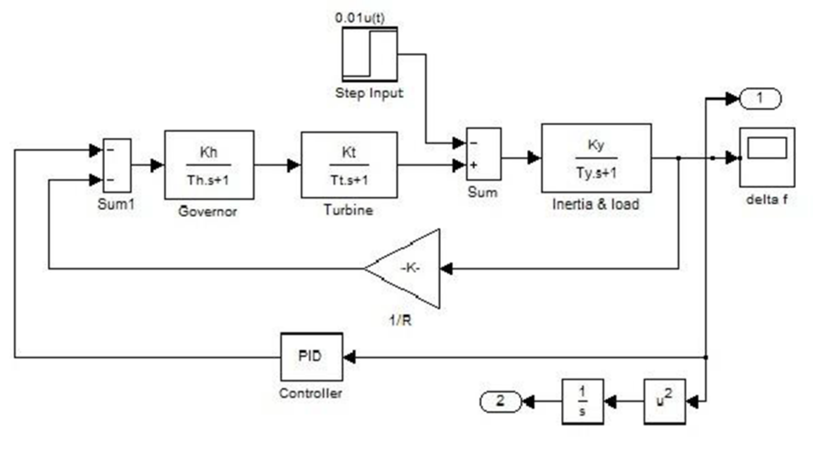
What is a model?

A human construct to help us better understand real world systems.



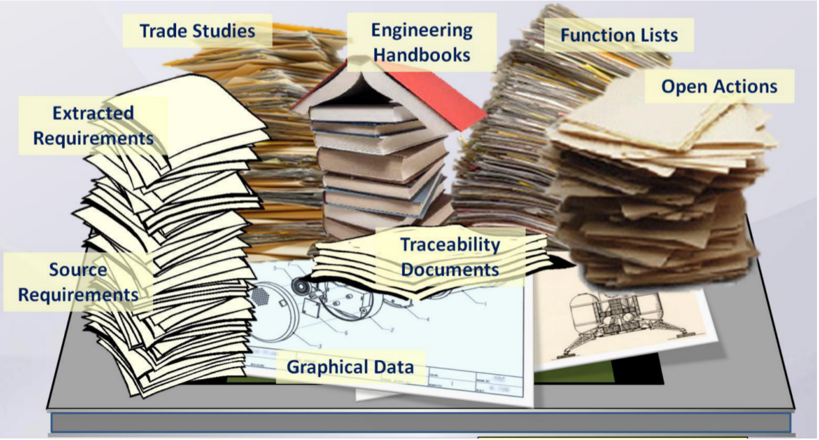
Models as abstractions

Models as domain specific constructs

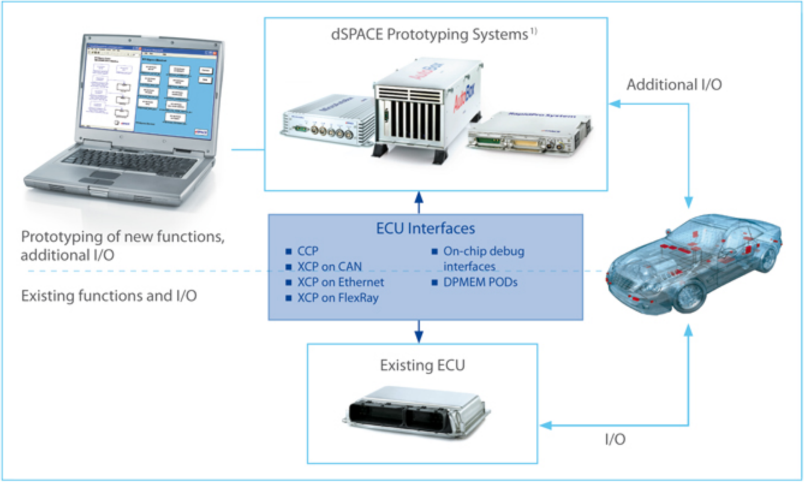


Models as digital convenience?!?

Every change affects something else :P



Models for rapid prototyping



Model based systems engineering

Model based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life-cycle phases.

"INCOSE Systems Engineering Vision" 2020 INCOSE-TP-2004-004-02 September, 2007

Essential components of MBSE

- A declared Metamodel/language
  - Structure and semantics
  - Textual/Graphical
  - Explicit, context-free language for communication
  - Problem, solution and management dimensions
- A process or methodology
- Defined mappings/projections
  - "Fit for purpose" views
  - Documentation and design artifacts
  - Other work products

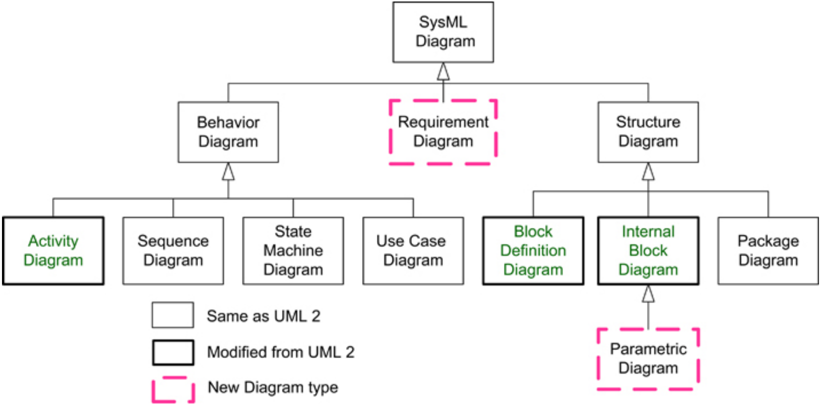
[source: Model-Based Systems Engineering, Zane Scott, Vitech Corporation]

- Good software tools ← This is often ignored!

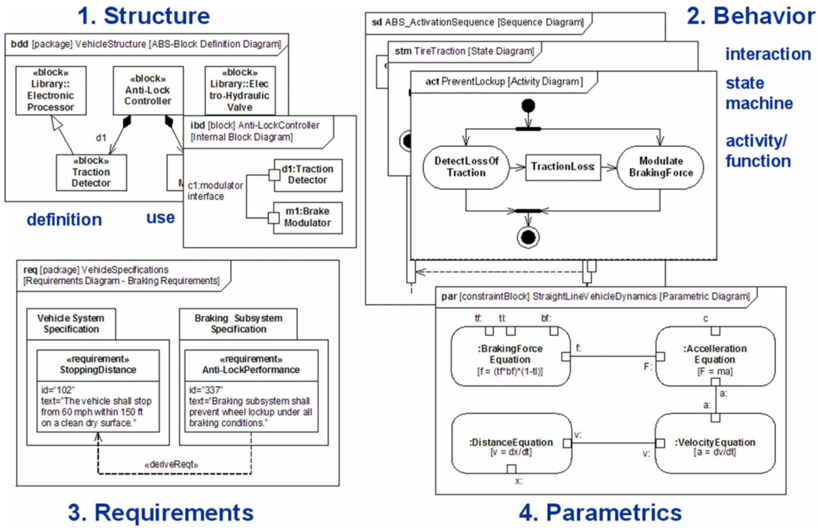
MBSE can be applied to

- Requirements
  - Analysis, traceability, test case generation, document generation...
- Architecture
  - Description, design space exploration, structure, behavior...
- Testing
  - Model in the loop, automation, optimization, parameterization...
- Validation and Verification
  - Coverage, property assuarance (safety, reachability, deadlock,...)
- Other things: Thermal, Mechanics, Assemblies, Fluid dynamics,...

SysML: Diagram types

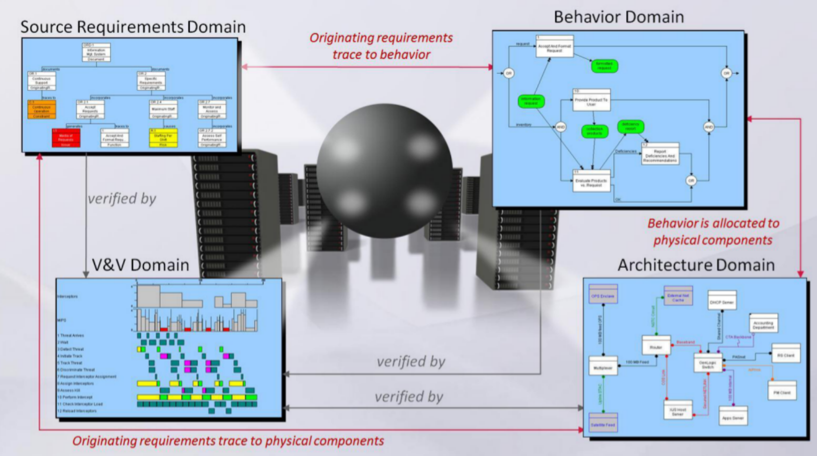


SysML: The four pillars



MBSE's golden dream

One (integrated) model + Toolchain integration



[image source: Model-Based Systems Engineering, Zane Scott, Vitech Corporation]

Some resources

- A Practical Guide to SysML, 2nd edition - Friedenthal et al
- OMG SysML tutorials (available online)
- Model Based Systems Engineering (available online)
  - A 116 slide presentation by Zane Scott, Vitech corporation

Takeaway: MBSE

- Models
  - Are limited representations of a system or process
  - Can be migrated into cohesive, unambiguous representation of a system
- In model based systems engineering
  - The 'model' is the system specification; conversely the system specification is the model
  - Visualizations are derived from the model, and the model is enriched through addition to the models

[source: Model-Based Systems Engineering, Zane Scott, Vitech Corporation]

So again...

How will you go about developing a quadrocopter?

Questions?

behere@kth.se