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."


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

[source: Derek Hitchins, former President of INCOSE]
The systems engineering process

Recognize Need/Opportunity

Identify and Quantify Goals

Create Alternative Design Concepts

Do Trade Studies

Select Concept

Increase the Resolution of the Design

Implement the Selected Design Decisions

Perform the Mission

Customer Needs

State the Problem

Investigate Alternatives

Model the System

Integrate

Launch the System

Assess Performance

Product and Process

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

Re-evaluate

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

Why is a process needed?

A systems engineering approach to design of complex systems
Systems Engineering

The V model

- Concept of Operations
- System Requirements
- Subsystem Requirements
- Subsystem Design
- Unit Design
- Code

- System Test
- Integration Test
- Unit Test
- Acceptance Test

A systems engineering approach to design of complex systems
A systems engineering approach to design of complex systems
A systems engineering approach to design of complex systems
Systems Engineering

Simplified processes

A systems engineering approach to design of complex systems

[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
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

- Elicit Requirements
- Analyze Requirements
- Document Requirements
- Validate Requirements
- Manage Requirements
- Stakeholder Participation
Requirements

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

“If I had asked people what they wanted, they would have said faster horses.”

—Henry Ford
Requirements

Requirements analysis - Structuring

- System Behavior ← 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..
Requirements

RE effort distribution

Effort Distribution for Complex Systems

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

Contents

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

A systems engineering approach to design of complex systems
"..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!
A systems engineering approach to design of complex systems

Architecture types

- Conceptual
  - Service Taxonomy
    - Behavior
    - User visibility
  - Logical
    - Block diagrams
    - de/composition

- Technical
  - Software
  - Hardware

Architecture

Partitioning
- Computation
  - Time
- Communication
  - Space
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)
A systems engineering approach to design of complex systems

(source: AUTOSAR.org)
A systems engineering approach to design of complex systems
A systems engineering approach to design of complex systems

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

Architecture

Extra functional properties

Service Taxonomy
Logical

Conceptual

Technical

Software
Hardware

Partitioning
Computation
Communication
Time
Space

Architecture

realizes
mapped
deployed

does not dictat
Architecture

Architecture description languages

SystemModel

- VehicleLevel
  - TechnicalFeatureModel

- AnalysisLevel
  - FunctionalAnalysisArchitecture

- DesignLevel
  - FunctionalDesignArchitecture
  - HardwareDesignArchitecture

- ImplementationLevel
  - AUTOSAR Application SW
  - AUTOSAR Basic SW
  - AUTOSAR HW

Environment Model

Data exchange over ports

Extensions ...

Requirements

Variability

Timing

Dependability

Allocation

A systems engineering approach to design of complex systems
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)
Architecture

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
A systems engineering approach to design of complex systems
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?
Testing, Verification and Validation

Verification and Validation

A systems engineering approach to design of complex systems
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 $\geq 6.3$ tons on each main landing gear strut

- System specification:
  - RT only if $S_1 = True$. Spoilers if $S_1 \parallel S_2$
  - ($S_1$) Weight of at least 6.3 tons on each main landing gear strut
  - ($S_2$) Wheels of the plane turning faster than 72 knots

What could go wrong?
(Lufthansa flight LH2904)
Testing, Verification and Validation

**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
Testing, Verification and Validation

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, Verification and Validation

Testing

A systems engineering approach to design of complex systems
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 procedure**
  1. Tasks to be performed.
  2. The setup of test environment
  3. The stimuli (selected test data)
  4. The expected outcome

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

- **V&V log**
  - Documentation of test executions, e.g., the actual outcomes of test procedures, time of execution, responsible persons..
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 processes need to be defined!
Testing, Verification and Validation

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, Verification and Validation

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
Testing, Verification and Validation

Fault and error injection

[source: Christmansson, J.; Hiller, M.; Rimen, M. An experimental comparison of fault and error injection]
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)
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
1. Systems Engineering ✓
2. Requirements ✓
3. Architecture ✓
4. Testing, Verification and Validation ✓
5. Safety
6. Model based systems engineering
Safety
Safety

Context

**Attributes**
- Availability
- Reliability
- Safety
- Confidentiality
- Integrity
- Maintainability

**Security**
- Fault Prevention
- Fault Tolerance
- Fault Removal
- Fault Forecasting

**Dependability**
- Means

**Threats**
- Faults
- Errors
- Failures

[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**

\[ \text{Risk} = \text{Probability} \times \text{Consequence} \]

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

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
Safety

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!
Safety

System safety process

1. Define Objectives
2. System Descriptions
3. Hazard Identification:
   - Identify Hazards and Consequences
4. Risk Analysis:
   - Analyze Hazards and Identify Risks
5. Risk Assessment:
   - Consolidate and Prioritize Risks
6. Decisionmaking:
   - Develop an Action Plan
7. Validation of Control:
   - Evaluate Results for Further Action

Risk Management

KTH

A systems engineering approach to design of complex systems
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

A systems engineering approach to design of complex systems
Safety

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:

3. Leveson, Safeware, Addison-Wesley, 1995]
Risk assessment

Exact method differs for each standard...

<table>
<thead>
<tr>
<th>Severity</th>
<th>Frequent</th>
<th>Likely</th>
<th>Occasional</th>
<th>Seldom</th>
<th>Unlikely</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catastrophic</td>
<td>I</td>
<td>E</td>
<td>E</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>Critical</td>
<td>II</td>
<td>E</td>
<td>H</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>Marginal</td>
<td>III</td>
<td>H</td>
<td>M</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>Negligible</td>
<td>IV</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

E – Extremely High
H – High
M – Moderate
L – Low
Safety

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
## SIL numbers - IEC/EN 61508

<table>
<thead>
<tr>
<th>Safety Integrity Level (SIL)</th>
<th>Mode of operation – on demand (average probability of failure to perform its design function upon demand)</th>
<th>Mode of operation – continuous (probability of dangerous failure per hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>( \geq 10^{-5} \text{ to } &lt; 10^{-4} )</td>
<td>( \geq 10^{-9} \text{ to } &lt; 10^{-8} )</td>
</tr>
<tr>
<td>3</td>
<td>( \geq 10^{-4} \text{ to } &lt; 10^{-3} )</td>
<td>( \geq 10^{-8} \text{ to } &lt; 10^{-7} )</td>
</tr>
<tr>
<td>2</td>
<td>( \geq 10^{-3} \text{ to } &lt; 10^{-2} )</td>
<td>( \geq 10^{-7} \text{ to } &lt; 10^{-6} )</td>
</tr>
<tr>
<td>1</td>
<td>( \geq 10^{-2} \text{ to } &lt; 10^{-1} )</td>
<td>( \geq 10^{-6} \text{ to } &lt; 10^{-5} )</td>
</tr>
</tbody>
</table>
Safety

Accident causality

System Anomalies:

- **Fault**: the cause of an error ("felorsak")
- **Error**: the state of component/system that may lead to its failure ("feltilstånd")
- **Failure**: Termination of component/system to perform its functions/services as required ("felyttring")

Environmental Impacts:

- **accident**: Harm or damage (e.g. death, injury, environmental or material
- **Hazardous Event**: Event(s) due to occurrence of hazards
- **Hazard**: Potential source of harm or damage
- **Operational Situation**: Defining the severity, exposure, and controllability of hazards
- **Environmental threats**: e.g. misuse, physical limits...

A systems engineering approach to design of complex systems
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.
Safety

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

- DO 178B/C Aeronautics
- IEC 61508 Generic standard
- ECSS Space (ESA)
- IEC 62304 Medical devices

- IEC 61511 Industrial processes
- IEC 61513 Nuclear industry
- IEC 62061 Machine safety
- EN 50126/8/9 Railways
- ISO 26262 Automotive
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)
Safety

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

- Requirements definition
- Product planning
- Design
- Process planning
- Production
- Usage
- Recycling

PLM Backbone

Assurance of properties

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

A systems engineering approach to design of complex systems
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

Prototyping of new functions, additional I/O
Existing functions and I/O

ECU Interfaces
- CCP
- XCP on CAN
- XCP on Ethernet
- XCP on FlexRay
- On-chip debug interfaces
- DPMEM PODs

dSPACE Prototyping Systems

Additional I/O
I/O

A systems engineering approach to design of complex systems
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 Metamodelling/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 assurance (safety, reachability, deadlock, ...)

Other things: Thermal, Mechanics, Assemblies, Fluid dynamics,...
MBSE

SysML: Diagram types

A systems engineering approach to design of complex systems
SysML: The four pillars

1. Structure

2. Behavior
- interaction
- state machine
- activity/function

3. Requirements

4. Parametrics

A systems engineering approach to design of complex systems
MBSE

MBSE’s golden dream

One (integrated) model + Toolchain integration

A systems engineering approach to design of complex systems
Some resources

- OMG SysML tutorials (available online)
- Model Based Systems Engineering (available online)
  - A 116 slide presentation by Zane Scott, Vitech corporation
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]
Wrap up

So again...

How will you go about developing a quadrocopter?
Wrap up

Questions?

behere@kth.se