GoedelWorks and The ASIL project

Eric Verhulst

www.altreonic.com
Some history

- R&D project of Open License Society:
  - Metamodel for systems engineering: “systems grammar”
  - OpenSpecs and OpenCookBook prototype web portal

- **EVOLVE** ITEA project
  - *Evo*olutionary *V*alidation, *V*erification and *C*ertification

- **ASIL**: Flanders Drive project on developing a common safety engineering methodology for automotive and related domains

- Currently commercialised and redeveloped by Altreonic under **GoedelWorks** by Altreonic
Refinement approach

- Refinement by adding structure and properties
- Avoids overlapping in concepts

Diagram:
- Domain
  - Entities
  - Entities and Interactions

- Metamodel
  - Structure & Architecture

- Model
  - System / Process

- Instance
Meta-levels for different users and different application domains

User levels
- M4: Mathematician
- M3: Expert
- M2: Engineer
- M1: User

Abstract meta-levels
- M4: element (of set) meta-meta-type
- M3: Metatypes declarations (inheritance of the M4 element meta-meta-type)
- M2: types declarations (inheritance of M3 meta-types with domain specific attributes)
- M1: Instances of M2 types with concrete values of attributes

RTOS domain
- M4: Kernel and libraries
- M3: Virtual machine executing M2 methods for M1 data
- M2: domain specific declarations (types, grammar, methods)
- M1: Data
The different views on a system

**View 1:**
- System = Processes + Architecture
- or: the "right" System = "how" + "what"

**View 2:**
- A process is a meta-system
- Has to be developed as well

In practice different views correspond to complementary domains:
- Process, Engineering, Modeling, Simulation, Testing, Software, Hardware, Safety, ...
## Systems engineering with just 16 meta-concepts

<table>
<thead>
<tr>
<th>System</th>
<th>Sub-systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Sub-Project</td>
</tr>
<tr>
<td>Process</td>
<td>Sub-Process</td>
</tr>
<tr>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>Requirement</td>
<td>Sub-Requirement</td>
</tr>
<tr>
<td>Specification</td>
<td>Sub-Specification</td>
</tr>
<tr>
<td>Resource</td>
<td></td>
</tr>
<tr>
<td>Work Package</td>
<td>Development, Verification, Test, Validation Task</td>
</tr>
<tr>
<td>Work Package Flow</td>
<td>Work Package</td>
</tr>
<tr>
<td>Work Product</td>
<td>Process type (&quot;evidence&quot;) or development (&quot;Model&quot;)</td>
</tr>
<tr>
<td>Model</td>
<td>Sub-Models</td>
</tr>
<tr>
<td>Entity</td>
<td>Sub-Entities</td>
</tr>
<tr>
<td>Change Request</td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td></td>
</tr>
</tbody>
</table>
Relationships

- **Dependency links:**
  - E.g. a SPC depends on REQ (n)
  - etc.

- **Precedence links:**
  - A WP precedes a WPT (n)
  - etc.

- **Structural links:**
  - A WP is composed of Tasks (n)
  - A Model is composed of Entities (n)
  - etc.
State Transitions in a Process

During the life-time of a Project/Process entities go through states:
- Defined => In Work => Frozen For Approval => Approved

Dependency and structural relationships create a partial order for Approval
- REF=>REQ=>SPC // RES // Tasks =>WP=>WPT (MOD)

A Project is a collection of Processes producing Work Products. Not one V-model but 100’s.

Overall Process follows from respecting states
- WorkProducts morphe (Resource at input is always result of previous Project)
Some differences

- Explicit difference between Requirements and Specifications
- Distinction Process (how) and Project (what)
- Verification = verifying the work done
- Testing = verifying the system meets specifications
- Validation = verifying it meets requirements (includes integration)
- Process/Project is not seen as flow but as a collection of steps producing WorkProducts
- System = Implementation model
- Safety case is seen as Specification-Fault case
- Domain agnostic
Trustworthiness as goal

Trustworthy system

<table>
<thead>
<tr>
<th>Safety</th>
<th>Security</th>
<th>Usability</th>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>no physical fault can cause harm</td>
<td>no injected fault can cause harm</td>
<td>no interface fault can cause harm</td>
<td>no personal data loss can cause harm</td>
</tr>
</tbody>
</table>

Specification has subtypes:
Normal Case, Test Case, Fault Case
Safety and Security case are subtypes of Fault Case
Application and validation using a Safety Engineering process

- **Input:** ASIL project of Flanders Drive
  - **Automotive Safety Integrity Level**
- **Goal:** develop common safety engineering process based on existing standards:
  - Automotive: off-highway, on-highway
  - Machinery
  - IEC 61508, IEC 62061, ISO DIS 26262, ISO 13849, ISO DIS 25119 and ISO 15998
- **Partners:**
  - Altreonc, DANA, EIA, Flanders Drive, Punch Powertrain, Triphase, TüV Nord
Process followed

- Acquiring general understanding of Safety and Systems Engineering standards.
- Development of ASIL process flow:
  - Dissecting standards in semi-atomic statements
  - Tagging according to activity domain
- Development of ASIL V-model with 3 Process domains:
  - Organisational Processes ("safety culture")
  - Supporting Processes
  - Safety and Engineering Development Processes.
- Completion
  - Identification of Work Products and RACI Roles
  - Development of templates for Work Products (.doc or .xls)
  - Development of Guidelines (e.g. HARA)
  - Development of Glossary
ASIL V-model

- Organisational
- Safety and Engineering/Development
- Supporting
ASIL Results

- Effort: approx. 21000 personhours (over 3 years.)
- Semi-atomic process requirements extracted: 3800
- Work products defined: 98 => templates
- Types of roles identified: 17 => HR responsibility
- Guidelines developed: 34 => templates
- ASIL process flow has 355 steps
  - Organisational processes identified: 19
  - Supporting processes identified: 75
  - Safety and Engineering processes identified: 261
- Work is not finished! (validation using use cases + organisation specific mapping) + iterative!
### ASIL import (1)

<table>
<thead>
<tr>
<th>GoedelWorks</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Process</td>
</tr>
<tr>
<td>Flow</td>
<td>Flow</td>
</tr>
<tr>
<td>Work Package</td>
<td>Step with descriptive text</td>
</tr>
<tr>
<td>Tasks (DEV, VET, TST, VAT)</td>
<td>Not defined</td>
</tr>
<tr>
<td>Project</td>
<td>Not defined</td>
</tr>
<tr>
<td>Model/Entity</td>
<td>Not defined</td>
</tr>
<tr>
<td>Reference</td>
<td>Standards’ requirements attached to Step</td>
</tr>
<tr>
<td>Requirement</td>
<td>Not defined, Step description</td>
</tr>
<tr>
<td>Specification</td>
<td>Not defined</td>
</tr>
<tr>
<td>Resource</td>
<td>Roles, Work Product template, Guidelines</td>
</tr>
<tr>
<td>Work Product</td>
<td>Work Product (input and output of Step)</td>
</tr>
<tr>
<td>Change Request</td>
<td>Not defined, but Change Management Step</td>
</tr>
<tr>
<td>Issue</td>
<td>Not defined, but Change Management Step</td>
</tr>
<tr>
<td>State</td>
<td>Not defined</td>
</tr>
<tr>
<td>Relationships</td>
<td>Net defined, except as WPT input and Roles</td>
</tr>
</tbody>
</table>
ASIL import (2)

V-model respected by following order:
- Steps become Work Packages
- Dependencies and structural relationships inserted but left empty
- State: most often “In Work” upon creation.

Benefits from import:
- All Process (and Project) Entities user-editable
- Project entities and Process entities can be linked
- Organisation specific instance of Processes can be created and new processes added
- Dependency analysis and reporting
Example project (1)

ASl imported reference
Example project (2)

Example of state verification (Approval)
Example project (3)

Dependency graph (Process)
Example project (4)

Example: shift-by-wire example
Example project (5)

- Generated precedence graph
Main lessons learned:
- Bridging different domains: semantic differences
- Safety engineering standards are subsets of systems engineering
- Certification requires “evidence” (artifacts)

Major problems:
- Find a common language
- Find a clean language: orthogonality
- Usability aspects prime requirement for tool
  - Difficult in a web based environment
- Standards’ license terms!
Conclusion

- Systems engineering process can be formalised using a common metamodel
- Booklet available from Altreonic website

Challenges
- Integration of different domains
  - Concepts, Architectural design, WorkFlow
  - System Engineering processes ("standards") are heuristic

Progress through formalisation
- Reduction of design space give reliability
- Modular architecture and unified semantics essential for incremental/evolutionary verification/validation/certification
- Automated support is feasible

- Work will continue in OPENCOSS FP7 project
  - (cover avionics, railway, automotive)
  - Focus on re-use of certification evidence