

# Ontology-guided requirements and Safety analysis

Tor Stålhane, NTNU

Inah Omoronyia, NTNU

Frank Reichenbach, ABB

# Goal

This work is part of the CESAR project.

The goal is to reduce the cost of:

- Verification – reduced by analysis performed based on a domain ontology.
- Safety analysis in the early phases – reduced by partly filled-in HazOp forms
- Certification – reduced by having the necessary items available in a database so that part of the safety case can be built automatically

# Process improvement

We will achieve our goals by improving the development process by introducing:

- Boilerplates – a set of templates for stating and formulating requirements
- A domain ontology – a domain vocabulary plus a set of rules for relationships between vocabulary items.
- Predefined HazOp forms linked to each requirement

# Boilerplates - 1

Boilerplates consists of reserved words, a structure and placeholders.

Two example:

- **The** <system> **shall be able to** <action> **to** <entity>
- **If** <operational condition> **the** <system> **shall** <action>

# Boilerplates - 2

Placeholders also have definitions, e.g.:

- <action> = <action> <entity>  
e.g. <open> <the tank>
- <operational condition> = <entity> (equal to  
| different from | greater than | less than |  
included in) <entity>  
e.g. <boiler pressure> **greater than** <max  
pressure>

# Boilerplates – examples

Two example requirements from a pilot study:

- **The** <controller> **shall be able to** [<open> <the tank>] **to** <the sewer>
- **If** [<boiler pressure> **greater than** <max pressure>] **the** <controller> **shall** [<sound> <alarm>]

# Ontologies

We use the domain ontology to

- Improve communication between domain expert, requirements engineer and software developers
- Derive additional information about the concepts used in the requirement
- Check for completeness, consistency and correctness

# Ontology – ACC example

Boilerplate

**If** <operational condition> **the** <system>  
**shall** <action>

ACC requirement:

**If** [<engage> <break>] **the** <ACC system>  
**shall** [<turn on> <the brake light>]

# GNLQ knowledge base view


Domain knowledge-base review







Domain Concepts

- Electric Park Brake Assist
- Positive Acceleration
- Wheel Speed Sensor
- Steady State System
- Unsteady System State
- preset time gap
- Preceding Vehicle
- Primary Target Selection
- ACC System
- ACC Type 2b
- Curve Road
- State
- Curve Radius Capability
- System State
- Potential Collision Threat
- Congested Traffic
- Speedometer Cable
- Left Direction
- ACC Type 1a
- Brake Light**
- Brake System Failure
- Controller

Definition

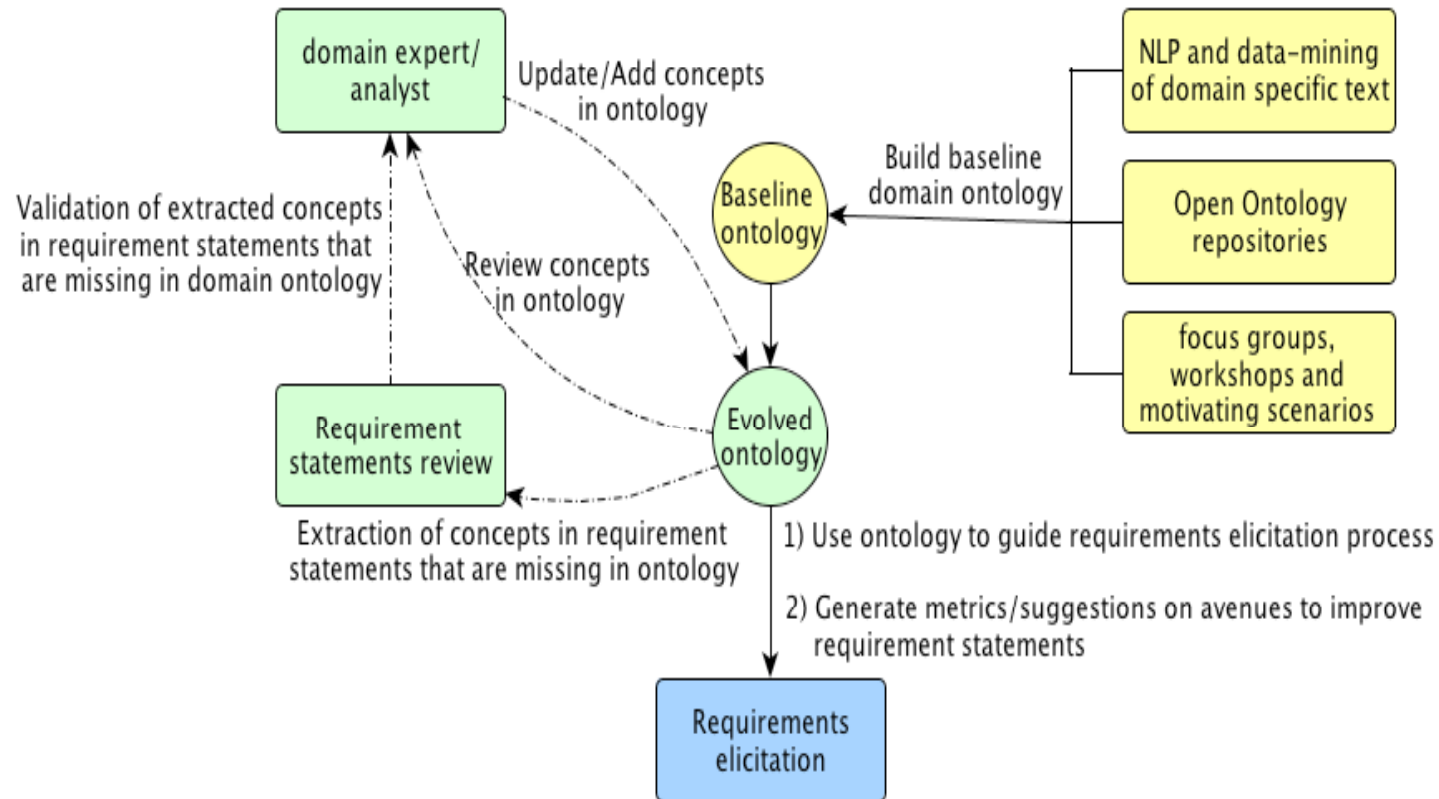
Red light indicator on the rear of a motor vehicle that signals when the brakes are applied to slow or stop the vehicle.

Refine 

!	Relations		
1	has ->State		
2	provides ->Visual Alert		
3	connected to ->Brake		

!	Failure Modes	
	Stuck	
	->Brake Light remains OFF or ON when Brakes is released by Driver	
	Omission	
	->Brake Light remains OFF when Brakes is applied by Driver	
	Commission	
	->Head Light turn OFF or ON when Brakes is applied by Driver	

# An evolving model – 1



# An evolving model - 2

As we insert more requirements, our knowledge of the system grows. On the next slide we see:

1. The currently inserted requirements
2. The definition of an item – the communication bus
3. The domain knowledge for a break

Statement Reviews

1 ACC System

- #37 The controller shall generate control signals to the brake and engine control systems of the ego-vehicle to maintain a preset headway distance between the ego-vehicle and a target immediately before it.
- #38 The collision warning system shall alert drivers with an audio warning when objects are in their path.
- #39 The stop-and-go function shall be disabled when driving through residential areas
- #41 The controller component shall be capable of sending control signals to the vehicle braking system, to the electric park and to the ego-vehicle engine control system to increase and reduce the speed of the ego-vehicle.
- #41 Therefore, the controller shall be connected to the communication bus implemented in the ego-vehicle.
- #41 When ACC is active, vehicle speed shall be controlled automatically maintain time gap to a forward vehicle, or to maintain the set speed is lower.

Completeness Inconsistency Opacity Ambiguity Noise Update-knowledge-base

Commit

!	Knowledge base concepts	Add	Invalid
1	communication bus	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	objects	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	path	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	target	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	stop-and-go function	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Domain knowledge-base review

Domain Concepts

- Visual\_Alert
- Brake
- Clutch\_Pedal
- On\_State
- Controller
- Desired\_Headway\_Distance
- Driveshaft

Definition

3

A brake is a device that decelerates a moving object such as a machine or vehicle by converting its kinetic energy into another form of energy, or a device which prevents an object from accelerating.

Define Relations Failure modes

!	Relation		
1	leads_to -ACC_Deactivation	<input type="checkbox"/>	<input type="checkbox"/>
2	requires_a -Brake_Light	<input type="checkbox"/>	<input type="checkbox"/>

Omission Commission Stuck

2

communication bus

A device that transfers control, timing, and data signals between switching processor subsystems.

Comments:

relations: interacts\_via

Concepts: Controller

Constraints: SomeValuesFrom

cancel Select

- Traffic
- Speed
- \_gap
- id
- or
- al
- ollision
- ollision\_Threat
- me\_gap
- Area
- ate

# Using the ontology

The ontology contains information on the actions that are relevant for a component. A domain expert can add information on the component's high level failure modes.

Based on this we can establish chains like:

Action => Element => Failure modes

E.g.

[<open> <the tank>] => valve => {s.a open, s.a. closed, ...}

# Early safety analysis

We have decided to use a simple form of HazOp in the early phases.

- One HazOp form per requirement
- The guidewords “Omission”, “Commission” and “Stuck”

Based on information available in the boilerplate and the ontology we can insert the failure condition. The rest must be inserted by a domain safety expert.

# Early safety analysis - example

BP61: The <operator > shall be able to <open the tank> to <the sewer>				
<b>Deviation (Guideword)</b>	<b>Failure condition</b>	<b>Effect of failure</b>	<b>Type</b>	<b>Remark</b>
Omission	<b>Not able to</b> <open the tank> to <the sewer>			
Commission	<open> also some other valve			
Stuck	<drainage valve> <b>s.a. on</b>			
	<drainage valve> <b>s.a. off</b>			

# Early safety analysis – example

## Two boilerplates combined

BP63, BP64: <b>If</b> [<boiler pressure> <b>greater than</b> <max pressure>] <b>the</b> <controller> <b>shall</b> [<sound> <alarm>]				
<b>Deviation (Guideword)</b>	<b>Failure condition</b>	<b>Effect of failure</b>	<b>Type</b>	<b>Remark</b>
Omission	<b>Not able to</b> <sound an alarm>			
	<b>Not able to</b> <detect too high pressure>			
Commission	Other alarm are also activated			
Stuck	<alarm > <b>s.a. off</b>			
	<pressure indicator> <b>s.a. low</b>			

# Benefits

By linking a partially filled in HazOp table to each requirement we obtain the following benefits:

- Each requirement *will* get a HazOp form with all available information inserted.
- We can automatically check whether all HazOp forms are completed.
- The finished HazOp for each requirement will be available in a database and can later be used in the final safety case for certification

# Current status

We have developed a tool for entering and analyzing requirements based on boilerplates and a domain ontology.

The tool is currently being tested out in four large European companies – among them ABB Norway.

The feedback we have received so far is promising.