Validating Industrial Requirements with a Contract-Based Approach

Matthias Bernaerts, **Bentley James Oakes**, Ken Vanherpen, Bjorn Aelvoet, Hans Vangheluwe, and Joachim Denil

Matthias.Bernaerts@student.uantwerpen.be,
{Bentley.Oakes, Ken.Vanherpen, Hans.Vangheluwe,
Joachim.Denil}@uantwerpen.be,
Bjorn.Aelvoet@dana.com

University of Antwerp

September 15, 2019

Table of Contents

1. Overview

2. Contract-based Approach

3. Contract Language

4. Contract Verification

5. Outcome and Conclusion

Outline

1 Overview

2 Contract-based Approach

3 Contract Language

4 Contract Verification

5 Outcome and Conclusion

Context

Automotive components are:

- Highly complicated cyber-physical systems
- Can be safety-critical

Example: **Electronic Differential Lock**

- Allow/prevent wheels from rotating at different speeds
  - Different speed for cornering
  - Same speed for traction assist
- Example Requirement: `The system must close the EDL after receiving a locking command from the driver.`

`dana.com/light-vehicles/products/driveline/axles/high-efficiency-advantek`

ASET Project

Automated and Simulation-based Functional Safety Engineering Methodology (ASET) Project

Goal: optimize design processes of functional safety components in the context of ISO 26262

- Need to **verify** component correctness
- **Lower time/cost** if performed early on in design process
- ISO26262 standard mandates **recording traceability** throughout development

Outline

1 Overview

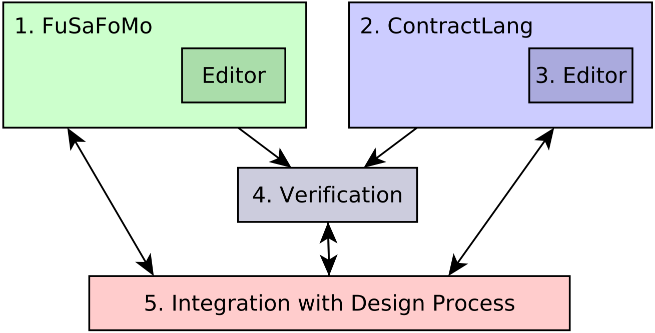
2 Contract-based Approach

3 Contract Language

4 Contract Verification

5 Outcome and Conclusion

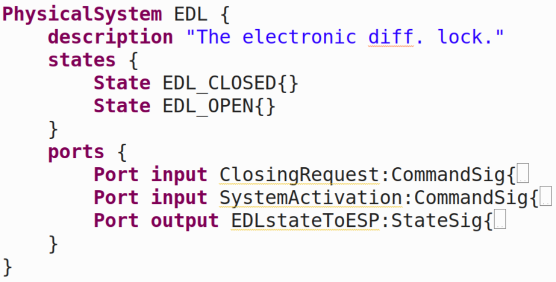
Contract-based Approach



1. Architectural language for functional safety components
2. Targeted contract language for functional safety
3. Tooling to formalize requirements in contract language
4. Tooling to verify contracts on system
5. Integration of contract approach in design process

Architectural Language

- aSET project also includes definition of system architecture model
  - Functional Safety Formal Model (FuSaFoMo)
- Supports the functional safety artefacts of ISO 26262, like *elements*, *items*, etc.
- Similar to aspects of SysML, can define interfaces to components



```
PhysicalSystem EDL {
    description "The electronic diff. lock."
    states {
        State EDL_CLOSED{}
        State EDL_OPEN{}
    }
    ports {
        Port input ClosingRequest:CommandSig{...
        Port input SystemActivation:CommandSig{...
        Port output EDLstateToESP:StateSig{...
    }
}
```

- Contract language can (optionally) refer to these ports and connections in contracts
- Have to enable parallel development of system architecture and contracts

Outline

1 Overview

2 Contract-based Approach

3 Contract Language

4 Contract Verification

5 Outcome and Conclusion

Contract Language

- Focused on providing a domain-specific language (DSL) for functional safety
- Under development, so syntax and semantics not finalized

Intention:

- Include relevant functional safety concepts
  - Elements from signal processing, probability
- Automatically map contracts to temporal logic for verification
- Provide easy-to-use editor for safety engineer

Statement Language

First component of contracts - *Statement Language*:

- Contains mathematical/logical operators on signals
  - Sets, averages, min/max, derivatives
- Example: “Event `driver_lock` := Port `driverCommands_closingRequest` == True”
- Example: “Event `unsafe_temp` := Port `EDL_temperature` >= 100”

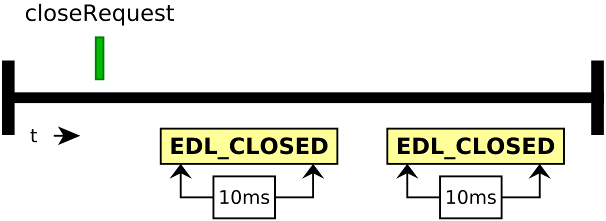
Major challenge:

- What operators are most appropriate for functional safety?
  - Mainly signal processing? - Pulses, rise, fall. . .

Scopes and Patterns

Second component of contracts - *Scopes and Patterns*:

- From Autili *et al.*
- Structured property specification patterns defining when the statements hold or not
- Example: After `closeRequest`, once `EDL_CLOSED` becomes satisfied, *it remains so for at least 10 ms*



- Few constructs - Five scopes, 16 patterns
- Seems to have sufficient expressiveness

M. Autili, L. Grunske, M. Lumpe, P. Pelliccione, and A. Tang,
"Aligning qualitative, real-time, and probabilistic property specification patterns using a structured English grammar", IEEE Transactions on Software Engineering, vol. 41, no. 7, pp. 620-638, 2015.

Contract Example

```
Contract FR07{
        longname "Response to driver locking command"
        description "The system must close the EDL after
                    receiving a locking command from the driver."

        statements{
            Event driver_lock :=
                Port driverCommands_closingRequest == True,
            Property close_EDL :=
                EDL_STATE in Set{State EDLphysicalSyst_CloseEDL}
        }
        scope Globally
        pattern ResponsePattern:
            if driver_lock has-occurred,then-in-response
                close_EDL eventually-occurs
        generate-STL
    }
```

- Contracts defined in editor created as XText plugin for Eclipse
    - Offers autocomplete, warnings/errors
- Low barrier to usage for partners

Outline

1 Overview

2 Contract-based Approach

3 Contract Language

4 Contract Verification

5 Outcome and Conclusion

Mapping to Signal Temporal Logic

- Autili *et al.* also provide a direct mapping to temporal logic

Example:

- ResponsePattern: if *P* has occurred, then in response *S* eventually holds (optionally within some amount of time).
- ResponsePattern: $\square(P\rightarrow \Diamond[t1,t2]S)$
  - always(P implies eventually[between t1 and t2 time units](S))
- Temporal logic can become very complicated
  - Response Pattern which is valid between two events Q and R: $\square((Q\wedge\square[0,t1]\neg R\wedge\Diamond[t1,\infty)R)\rightarrow(P\rightarrow(\neg R U[t1,t2](S\wedge\neg R)))UR)$
- Goal: shield functional safety engineer from this complexity
  - Editor automatically produces STL formulas when contracts are saved

Verifying Signal Temporal Logic

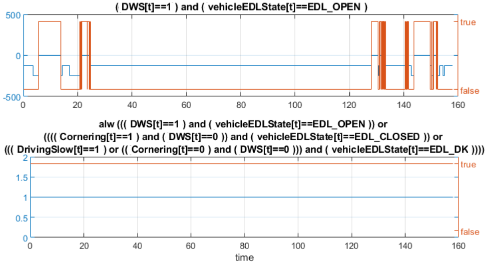
- Using the Breach toolbox for STL verification
    - Within the Matlab framework
- Proves STL formulas against simulation traces

A. Donzé, “Breach, a toolbox for verification and parameter synthesis of hybrid systems,” in International Conference on Computer Aided Verification. Springer, 2010, pp. 167–170. Available: https://github.com/decyphir/breach

Verification Example

Example contract regarding the state of the EDL:

```
Event vehicle_EDL_State :=
    EDLVehicleState from-table
        case DWS==True
            : Set{EDL_State OPENED}
        case and{ DWS==False, Cornering==True}
            : Set{EDL_State CLOSED}
        case DrivingSlow==True
            : Set{EDL_State UNKNOWN}
        case and{ DWS==False, Cornering==False}
            : Set{EDL_State UNKNOWN}
```


(DWS[t]==1) and (vehicleEDLState[t]==EDL_OPEN)

alw (((DWS[t]==1) and (vehicleEDLState[t]==EDL_OPEN)) or
((((Cornering[t]==1) and (DWS[t]==0)) and (vehicleEDLState[t]==EDL_CLOSED)) or
(((DrivingSlow[t]==1) or ((Cornering[t]==0) and (DWS[t]==0))) and (vehicleEDLState[t]==EDL_DK))))

- Provides boolean satisfaction, and quantitative satisfaction throughout trace
- Details satisfaction for each sub-formula
- Manual verification for now, automation to follow

Outline

1 Overview

2 Contract-based Approach

3 Contract Language

4 Contract Verification

5 Outcome and Conclusion

Outcome

Outcome 1: Ambiguous and conflicting requirements detected

```
Contract FR12{
        longname "Report to ESP"
        description " During normal operation and within
        t_SYSTEM_RESPONSE, the system must report to the ESP that
                             ...
        scope Before enter_error_state
        pattern Recurrence:
            ReportState occurs-repeatedly every 10 ms
    }
```

Informal text *"within"* suggests a *ReponsePattern*, but this is a *RecurrencePattern*

Outcome 2: Uptake by DANA

> *"Safety functions" [...] need to be simple (as in non-complex) in nature, and formalization helps in attaining the (process) requirements for the higher safety integrity levels. Dana is planning on incorporating the contract-based requirements technology as part of the current day-to-day practices for validating safety-critical systems."*

Conclusion

ASET project is in progress:

- Improving design process for safety-critical automotive components
- Developing a domain-specific contract language (DSL) for functional safety
- Providing tooling for editing contracts, mapping to temporal logic, and contract verification
- Integrating contracts into design process (and day-to-day practices)

Future work:

- Enhance contracts with relevant operators
- Automate verification and bring results back to editor
- Examine static verification of contracts
  - Employ OCRA tool for component consistency

Questions?