On Integration of MBSE and Simulation for Evaluation of Complex, Quantitative, Temporal Key Performance Indicators

Lukas Macha, University of Bath, lm2292@bath.ac.uk & Peter Ebner, AVL List GmbH & Julian Padget, University of Bath & Chris Brace, University of Bath

Categorisation

- Accessibility: PRACTITIONER
- Application: RESEARCH
- Topics: MBSE, Simulation, Key Performance Indicators

Abstract

Engineering of today's complex automotive systems relies on heterogeneous, model-based development toolchains to compute various performance measures to demonstrate product quality and support decision making. Despite advances in System Modelling and simulation, their integration often hits a bottleneck due to cumbersome, expensive and error-prone process of manual transformation of information regarding verification criteria from authoritative System Models into domain specific simulation toolchains. To overcome these obstacles, we introduce a novel methodology that facilitates integration of System Models, built using System Modelling Language (SysML), with simulation, to enable evaluation of quantitative Key Performance Indicators (KPIs) at every system life cycle stage. This is demonstrated in a five-step process using an automotive Adaptive Cruise Control (ACC) application. First, we develop a comprehensive SysML model to analyse the system in its intended context. Second, we extract the KPIs from system requirements written in natural language and formalize them in Signal Temporal Logic (STL) to define constraint parameters. Third, we capture the atomic propositions of STL KPIs to establish traceability between constraint parameters, system properties and interfaces. Fourth, we establish correspondence between the System Architecture and domain-specific models. Fifth, simulation capabilities are leveraged to evaluate temporal, quantitative KPIs, providing insight into the system performance in achieving the desired task. This novel integration eliminates the need for manual transformation of information from System Models to simulation toolchains, reduces opportunity for error, and enhances scalability to streamline the development process of automotive systems using Model Based Systems Engineering (MBSE).

Introduction

Modern automotive systems have evolved into Cyber-Physical Systems (CPS), where discrete-time electrical and electronic system elements interact directly with mechanical, continuous-time elements [Zheng et al 2017]. Whilst these systems can fulfil ever-increasing stakeholder requirements and ever-more stringent emissions regulations, they make the complexity of today's CPSs unmanageable using traditional Document-Intensive Systems Engineering processes potentially resulting in project

overruns and costly delays [Wasmer et al, 2011]. To successfully design and verify such CPSs, one needs artifacts such as the system context, structure, and requirements, describing the desirable properties of the system behaviour [Deshmukh & Sankaranarayanan, 2019; Sierla et al, 2013], that can be further decomposed into lower-level system specifications for individual subsystems and components. MBSE allows management of the complexity of such systems through thorough modelling of system requirements, structure, behaviour and parametrics in a single-source-of truth System Model, [Walden et al 2015]. Such a model can then be used and reused throughout the Model-Based Vehicle Development Process to authorise work packages as well as dictate simulation and testing activities to complete for system verification purposes.

Despite the latest advancements in MBSE, the difficulties of running dynamic system simulations and extracting parametric KPIs, or importing results back, often remains the bottleneck to using the system model beyond the information capture phase and the promised potential of MBSE has not yet been fully realised [Henderson & Salado, 2021]. This also holds true also for simulation and testing activities. Domain specific modelling tools are instead used to develop simulation models that serve as a virtual prototype of the system of interest [Puntigam et al, 2021]. These virtual prototypes provide means to analyse system behaviour first, to simulate the system under various inputs and observe how the system reacts and second, to observe any hidden variables, that might be difficult or expensive to measure. It is through these high-fidelity, physics-based simulation models, where one can, at various phases of the system development process uncover undesirable system trajectories or erroneous states. Generated results are however often inspected manually or analysed using manually programmed property testers [Bartocci et al, 2018]. Whilst accepted as common practice, the diversity of stakeholders in the Vehicle Development Process (VDP), can lead to ambiguity in the requirement and parametric capture process and these inconsistencies can result in misinterpretations leading to design teams developing and testing to wrong specifications.

To address these shortcomings, we present a methodological approach, that integrates MBSE and Simulation. This allows automated transformation of parametrics from the System Model into formal behavioural specifications that facilitate effective communication, eliminate ambiguity, and provide an artifact, that is directly transformed into property checkers and monitoring programs. These formal behavioural specifications provide qualitative and quantitative measures of system performance to decide whether the system satisfies its requirement, directly dictate what system properties need to be verified and authorise resource allocation for simulation and testing activities.

Methodology

First, we develop a solution agnostic System Model describing the Problem Domain where the Stakeholder Needs are captured and refined. The modelling continues describing the Use Case Scenario, capturing the individual system functions to satisfy stakeholder needs and allocating those functions to individual Functional Blocks. The System Model allows specification of dependencies and interactions between system functions resulting in common interfaces and influencing (noise and control) factors.

The System Context with interface and item flow definitions is shown in Figure 1. The two Functional Blocks, as shown, are Driver and Vehicle, along with their Information Interfaces. The Driver Interface provides the Setpoint Velocity, Accelerator Pedal Command, and Brake Pedal Command signals. The

Vehicle Information interface conveys the Actual Velocity state and Acceleration signals and noise factors from the Environment are also captured.

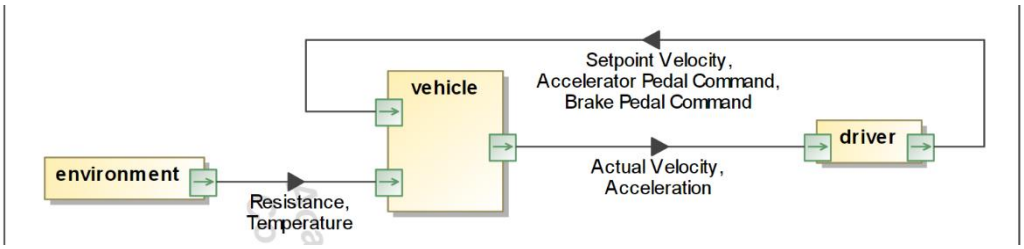

Figure 1. Internal Block Definition Diagram (classifiers hidden for clarity)

Upon specifying system interfaces, parameters and behaviours, the Parametric Diagram is used to model the KPIs associated with the Cruise Control functionality. A subset of 4 requirements was selected to demonstrate the approach. These are:

1) REQ 1: The maximum difference between the *actual speed* and the *desired speed* shall be less than or equal to *max desired velocity delta* of **1.5 km/h**.
2) REQ 2: The *mean absolute velocity delta* between the *actual speed* and the *desired speed* shall be less than or equal to **0.5 km/h**.
3) REQ 3: Upon change of the *desired velocity setpoint* the *acceleration* shall not be greater than **1.5 m/s²** and *deceleration* shall not be less than **-1.5 m/s²**.
4) REQ 4: The *maximum vehicle velocity overshoot* shall not be greater than **0.2 km/h**.

These four requirements are temporal and require mathematical formula to be evaluated, there are:

1) $abs(v_{desired} - v_{actual}) \leq v_{\max Delta}$
2) $mean(abs(v_{desired} - v_{actual})) \leq v_{meanDelta}$
3) $a_{min} \leq a \leq a_{max}$
4) $(v_{actual} - v_{desired}) \leq v_{overshoot}$

Where: setpoint velocity $v_{desired}$, actual velocity v_{actual}, max desired velocity delta $v_{\max Delta}$, mean absolute velocity delta $v_{meanDelta}$, maximum acceleration a_{max}, maximum deceleration a_{min}. Traditionally these expressions would be programmed manually in the result postprocessing phase after running a simulation or physical experiment. This can however be prone to errors due to misinterpretation, typos, or out-of-date acceptance measures, potentially leading to design teams developing to wrong specifications. To avoid this from happening, formal behavioural specifications were introduced for the design and verification of CPSs with the rationale that they facilitate effective communication, eliminate ambiguity, and provide artifacts that can be directly transformed into property checkers or monitoring programs to qualitatively and quantitatively decide whether the system satisfies its requirements, [Donze et al, 2013]. To leverage such formal behavioural description, we capture these KPIs as constraints in Parametric Diagrams and through an automatic transformation extract these constraints from the Parametric Diagrams and transform them into STL formulas. First, the *"Absolute delta setpoint velocity"* constraint is modelled to identify the relevant continuous-time signals associated with the computation of the absolute value. This establishes the traceability between the system interfaces and the constraint parameters. The same is done for *"Maximum acceleration"* and *"Maximum deceleration"* constraints. Second the Atomic Propositions KPI 1, KPI 2, KPI 3 and KPI 4 are modelled as shown in *"Max absolute delta to setpoint velocity", "Mean absolute*

delta to setpoint", "Maximum acceleration" and *"Maximum deceleration"* constraint blocks. These allow modelling of the basic comparison between the signals of interest and thresholds (acceptance criteria) that can be changed throughout the VDP. KPI 3 and KPI 4 are combined using the logical AND operator denoting that both constraints shall be satisfied at any one time. Following similar logic one can construct and trace more complex STL formulas and combine Atomic Propositions with logical and temporal operators. A Parametric Diagram for REQs 1, 2 and 3 is shown in Figure 2:

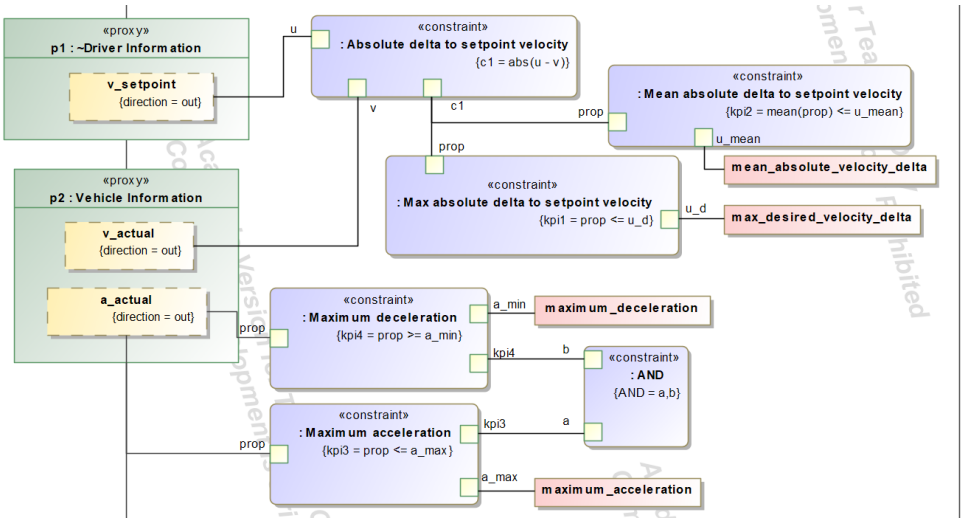

Figure 2. KPIs Captured Using Parametric Diagram

Results

The transformation process from graphical constraint representation in the Parametric Diagram into the Signal Temporal Logic formulas is depicted in Figure 3. First, the SysML model is exported using the UML 2.5 XMI standard providing an XML document that can be processed with available programming languages and libraries. The document is iterated through, and every packaged element of stereotype Constraint Block is extracted. For every extracted constraint, its owned attributes are enumerated, and it is identified whether they are traced to interfaces (inputs/outputs) or to system properties. Identified I/O attributes are then monitored at every timestep and constant (system properties) are assigned as params of the STL formulas. Every constraint, unless modelled otherwise, is automatically assumed to be safety property, and therefore prepended with the *"always"* temporal operator indicating that the property must globally hold true.

Each constraint from the Parametric Diagram is first transformed into an atomic STL proposition with Constraints Parameters mapped onto either the system interfaces or parameters associated with the Functional Block itself. This differentiation in the transformation process is important as it allows assigning which parameters in the STL formula are continuous-time, time-series variables that need to be monitored at every time-step and which variables are constant parameters associated with the Functional Block itself. Upon extraction of individual constraints as single Atomic Propositions each is checked for links to other atomic propositions in its formula or if it is linked to a logical or temporal operator and if so, the constraints are combined accordingly.

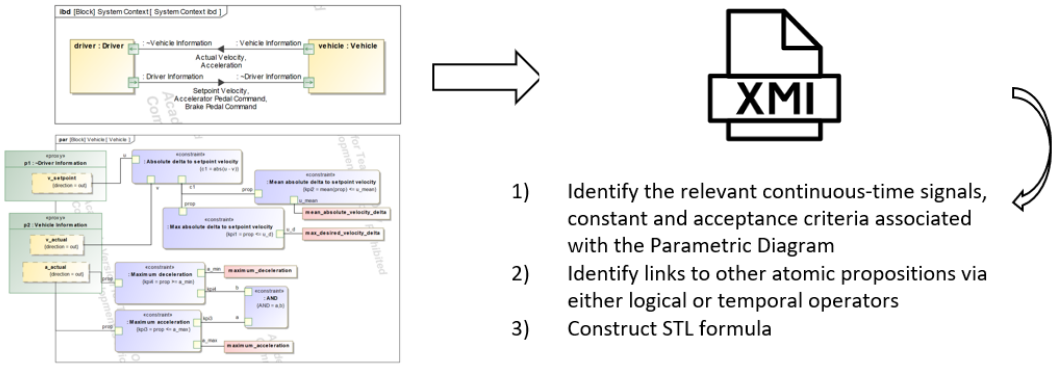

Figure 3. KPI transformation process

This formalised KPI modelling approach facilitates automatic transformation of KPIs modelled using the Parametric Diagram into formal specification language of STL that is consequently used for KPI monitoring and evaluation. The automatically transformed KPIs rewritten in the STL are shown below.

$$\boldsymbol{\phi_1 = always_{[0,END]}(abs(u[t] - v[t]) \leq u_d)} \qquad \boldsymbol{\phi_3 = always_{[0,END]}(a[t] \leq a_{max} \wedge a[t] \geq a_{min})}$$

$$\boldsymbol{\phi_2 = always_{[0,END]}(mean(\phi_1) \leq u_{mean})} \qquad \boldsymbol{\phi_4 = always_{[0,END]}(v_m[t] \leq u[t])}$$

Where ϕ_i is the i_{th} KPI, $always_{[START,\,END]}$denotes a safety requirement that must be satisfied between time $START$ and END, and $\wedge$ is the logical AND operator, meaning that both atomic propositions $a[t] \leq a_{max}$, $a[t] \geq a_{min}$ must be satisfied.

To evaluate these temporal system properties, we combine MBSE and Simulation. The expressiveness and customizability of SysML offers various methods of constructing dynamic models that can in some cases, thanks to developing standards such as Foundational UML, [omg.org 2021], a subset of UML that focuses on executable models, be simulated and properties computed during this dynamic model evaluation. Often, however, to thoroughly analyse the system under development requires solving governing differential equations that can only be approximated within desired tolerances. These equations are difficult, if not impossible, to express in the System Model using SysML. We thus take advantage of the capabilities of simulation tools for the dynamic system behaviour, obtain time-series results, and monitor these to check whether they meet the requirements specified in the System Model.

Conclusion

This paper introduces a process of going from natural language requirements to their STL representation captured using SysML Constraints to obtain exact mathematical expression for evaluation of KPIs, removing possible ambiguities and achieving traceability between the individual parameters of the constraint itself and its corresponding system elements. This enables an automated, scalable and interpretable approach to monitoring properties of time-series data obtained from simulation and testing activities and offers an alternative to tedious manual inspection of simulation traces or ad-hoc programmed property testers. As this approach only requires access to the system

interfaces, the actual system performance can be verified even for black-box models that protect intellectual property. This provides a powerful technique for analysing the system behaviour of complex automotive systems where white-box model exchange is often undesirable. Leveraging the capabilities of simulation tools can therefore enhance current potential of MBSE for evaluation of KPIs. Whilst this paper presents a method that brings MBSE and Simulation a step closer, there are limitations in its adoption due to the modelling efforts and expertise required. This can however be offset by introduction of reusable libraries of standardised (either company or project-specific), parameterizable constraints, simplifying the modelling step and reducing the potential modelling overhead.

Acknowledgement

Lukas Macha is supported by a scholarship from the EPSRC Centre for Doctoral Training in Advanced Automotive Propulsion Systems (AAPS), under the project EP/S023364/1

References

[Walden et al 2015] Walden DD, Roedler GJ, Forsberg KJ, Hamelin RD, Shortell TM. (2015). '*INCOSE Systems Engineering Handbook – a guide for system life cycle processes and activities*. 4th ed. San Diego: WILEY.

[Puntigam et al 2021] Puntigam W., et al. (2021). '*Integrated and Open Development Platform for the Automotive Industry*'. In *Systems Engineering for Automotive Powertrains*. Springer.

[Donze et al 2013] *Donze, A., et. al (2013*). *'*Efficient Robust Monitoring for STL'. *In: CAV 2013*. LNCS.

[Henderson & Salado 2021] *Henderson K, Salado A., (2021*). *'*Value and benefits of model-based systems engineering (MBSE): Evidence from the literature'. *In: CAV 2013*. LNCS.

[Deshmukh & Sankaranarayanan 2019] *Deshmukh, J. & Sankaranarayanan S. (2019*). *'*Formal Techniques for Verification and Testing of Cyber-Physical Systems'. *In: Design Automation of Cyber-Physical Systems 2019*.

[Bartocci et al 2018] *Bartocci E, et. al (2018*). *'*Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications'. *In: Lectures on Runtime Verification*. LNCS.

[Wasmer et al 2011] *Wasmer A., Staub G., Vroom, R. W. (2011*). *'An industry approach to shared, cross-organisational engineering change handling – The road towards standards for product data processing*'. *In: Computer Aided Design*. Elsevier.

[Zheng et al 2017] *Zheng X, Julien Ch, Kim M, Khurshid S., (2017*). *'Perceptions on the State of the Art in Verification and Validation in Cyber-Physical Systems*'. *In: IEEE Systems Journal*.

[Sierla et al 2013] *Sierla S, et al (2013*). *'Common cause failure analysis of cyber-physical systems situated in constructed environments. In: Research in Engineering Design*. Springer

[omg.org 2017] Omg.org. (2017). '*OMG System Modelling Language Specification Version 1.5'*. [online] Available at: http://www.omg.org/spec/SysML/1.5/ [Accessed 02 May. 2023].

[omg.org 2021] Omg.org. (2021). '*FUML Semantics of a Foundational Subset for Executable UML Models'*. [online] Available at: https://www.omg.org/spec/FUML [Accessed 05 April. 2023].