J.Natn.Sci.Foundation Sri Lanka 2019 47 (4): 393 - 408
DOI: http://dx.doi.org/10.4038/jnsfsr.v47i4.9678

RESEARCH ARTICLE

Integrating runtime validation and hardware-in-the-loop (HiL) testing with V & V in complex hybrid systems

S.D. Dewasurendra[1*], A.C. Vidanapathirana[2] and S.G. Abeyratne[3]
[1] *Department of Computer Engineering, Faculty of Engineering, University of Peradeniya, Peradeniya.*
[2] *Industrial Service Bureau (ISB), Kurunegala.*
[3] *Department of Electrical and Electronic Engineering, Faculty of Engineering, University of Peradeniya, Peradeniya.*

Submitted: 03 February 2018; Revised: 04 July 2019; Accepted: 26 July 2019

Abstract: This study addresses the problem of assuring provably safe and correct behaviour of safety-critical complex hybrid systems (CHS) throughout their life-cycles when physical system dynamics tend to change due to natural causes. Model-based development methods are needed that integrate formal specification, verification, implementation level testing and runtime validation. Scalability limitations of available algorithms/methods dictate a modular approach, but this poses conflicting issues of compositionality, false-transitivity, soundness and completeness. In this paper a compositional solution approach based on the decomposition and hybridisation of statechart models (into hybrid automata - *HA*) is demonstrated. Specifically, a compositional formal verification methodology developed earlier for discrete event dynamic systems (DEDS) was elevated to hybrid dynamics, successfully overcoming the risk of false transitivity common to direct abstraction methods of continuous state-space. It was then used to develop a scalable strategy to generate sound and complete (relative to coverage criteria) tests, and configure corresponding compositional HiL tests for different abstraction and functional levels. The same decomposition was used to derive compositional runtime validation tests based on discrete invariants and differential invariants. The study formally proves that (1) the proposed HA based formal verification method is compositional, sound and complete relative to first-order logic of differential equations, (2) the modular tests are compositional and (3) the HA based test generation method is compositional, sound and complete relative to first-order logic of differential equations. To reduce complexity compositionality is rendered parallel than sequential to perform the simpler tasks concurrently. The claims have been validated experimentally on a full scale experimental rig.

Keywords: Complex hybrid systems, compositional testing, compositional verification, hardware-in-the-loop testing, runtime validation, V&V – testing integration.

INTRODUCTION

The underlying problem addressed is assurance of provably safe and correct behaviour according to requirement specifications, in real-scale non-terminating (repeatedly looping) safety-critical complex hybrid systems (CHS) throughout their life-cycles when physical system dynamics and the environment tend to change due to natural causes. These systems have interacting continuous-variable dynamics (CVDS; position, velocity, friction, etc., expressed by differential equations or difference equations based on the model of time used: discrete time or continuous time) and discrete-event dynamics (DEDS; on/off, collision, etc., expressed by transition systems, formal languages or abstract algebras); hence called hybrid systems.

A model-based solution (the dominant paradigm (Alur, 2011)) requires expressing the requirement specifications in a formal model where the correctness and safety properties could be verified, enabling a faithful translation of the verified model into a control programme to be coded into software/hardware.

Formal verification consists of ascertaining the continued conformance of system behaviour to

* Corresponding author (devapriyad@pdn.ac.lk; https://orcid.org/0000-0002-4280-4037)

specifications. Methods employed in verifying draw heavily from methods for DEDS and CVDS: for DEDS, classically, model checking (for temporal and first-order logic specifications) on transition systems and theorem proving on formal language models and process algebras have been extensively discussed (Aceto *et al*., 2007). CVDS specifications were classically limited to overshoot, rise-time, etc. (Belta *et al*., 2017). Verification of nontrivial behavioural specifications (based on temporal and differential dynamic logics) using model checking has now been made possible by providing them with transition structures through the abstraction of the state-space into affine partitions (based on observational equivalence relations -bisimilarity-). This becomes states of timed or hybrid automata with discrete transitions defined for affine boundaries (Belta, 2017; Sloth & Wisnieski, 2011), *albeit* with a risk of over-approximation and false-transitivity.

Hence, hybrid automata (HA) (Henzinger, 1996) are dominant among formal models of hybrid systems.

The core problem in model checking methods is image computation (Platzer & Clarke, 2007). For DEDS, symbolic computations help in abstracting the reachability space (the image) on a transition system, particularly for first-order logic specifications which require quantifier elimination. HA may have specifications with quantification over reals (first-order logic on reals), which would require real quantifier elimination (RQE) for model checking: RQE is impossible when modalities are present; some differential equations do not support RQE; even otherwise, complexity of RQE in real-closed fields is doubly exponential in the number of quantifier alternations. In numerical or approximation approaches, approximation errors can cause unsoundness in model checking. Platzer (2010) proposes methods based on differential induction for differential invariants to address this situation. Local invariants are found in a fixed-point algorithm for each continuous evolution $D_{\wedge}H$, with differential equation system D and evolution domain H. The local invariants are then composed into global invariants using closure properties.

However, these verification approaches are designed for hybrid systems with centralised sequential control, a limitation arising from models of computation (MoC) used here: (e.g. hybrid automata). An approach suitable for distributed systems (with, for instance, networked control, as in automobiles or aircrafts) needs MoC with richer semantics: permitting parallel composition of sequential components, thus concurrent computation. Our MoC of choice for DEDS has been statecharts (Harel, 1987) for these reasons and it was extended to capture CVDS.

Some currently used correct-by-design model-based development approaches for reactive systems, notably the B-method and Event-B-method (Abrial, 1996; Moreira, 2015) too prescribe conducting formal verification and validation (V&V) throughout the design process: a progressive refinement process with a theorem proving system for proof obligations at each refinement stage. The premise is that detailed models cannot be built at the beginning of a development process. Event-B has been extended to handle hybrid dynamics by adding simple differential dynamic logic (dL) operations and defining a refinement calculus for differential events (Liu *et al*., 2014).

However, without compositional strategies for complex systems, development processes can quickly become computationally intractable and unsound/ incomplete: the case with current decomposition and refinement strategies (theoretical or commercial) (Abrial, 1996; 2010), including the Event-B method. Disappointingly, guaranteeing completeness in compositional proofs is difficult (Namjoshi *et al*., 2010).

In Platzer (2010) a decomposition of hybrid systems into a sequential composition of hybrid programme (HP: their preferred MoC) components is presented; HP being extensions of conventional discrete programmes (Harel, 1979), the programmes of subsystems can be composed using logical operators in a compositional manner. However, this approach does not support parallel composition of the sequential components. Further, hybrid textual programmes lack the familiar structure and expressive power of graphical HA.

Chaochen *et al*. (2005) realised compositionality of parallel components in hybrid systems by introducing continuous statements into process algebraic models, CSP (Hoare, 1985) to be precise, defining hybrid CSP. However, verification methods for hybrid CSP are yet to be developed.

In this paper we demonstrate our compositional approach to formal verification and test design, which supports parallel composition of sequential HA: statecharts (Harel, 1987) modelling discrete-event dynamics of a hybrid system are decomposed into a set of automata (language generators) communicating through port-structures using a decomposition we introduced previously (Dewasurendra, 2006; Vidanapathirana *et al*.,

2011) based on Drusinsky and Harel (1989). Differential equations/constraints are embedded into automata states representing real actions in the physical system. As will be seen in our development, such states are few in numbers in these modules and ports. The generated architecture is unique, permitting efficient compositional verification of hybrid dynamics.

In addition to design verification, testing a CHS implementation is mandatory: for instance, ISO 26262-6 requires back-to-back testing of a model and an implementation derived from the model. Testing needs to be sound and complete, while avoiding redundant coverage. Sound and exhaustive tests are generated in (Tretmans, 2008) for a given specification in the limited context of an ioco (input-output conformance) labelled transition system. In complex systems, modularity needs to be exploited, but there is not much reported on compositionality in test design: (Bijl *et al*., 2003; Daca *et al*., 2014) are exceptions, treating ioco labelled transition systems. These are still open problems for more general configurations.

In Ferrante *et al*. (2016) formal models of the system (simulink blocks and stateflow) are translated to NuSMV programmes, which are then run against negations of CTL formulas representing individual test objectives to generate counter-examples that become tests covering the required test criteria. The approach is not compositional, nor does it support continuous dynamics, and may not scale up. Time partition testing by Bringmann and Kramer (2006), targets continuous behaviour testing, extendable to hybrid components using stream-processing functions of Müller and Scholz (1997). However, their tests do not derive from or relate to formal verification models.

Finally, variations in physical system dynamics and environment not captured in the modelling phase require runtime validation to ensure continued safety.

Hence, test design and runtime validation need to be contemplated from early stages of modelling for formal verification and validation (V&V) (ARTEMIS, 2005; Namjoshi & Trefler, 2010; Nielsen, 2014; CRYSTAL, 2016) as required by standards such as EN50128, DO-178C, IEC 61508, IEC 60880 and ISO 26262.

Formal validation of model-based development at instrumentation level is still poorly supported (Bringmann & Kramer, 2008). De Matos (2015) and Nielsen (2014) help comparing a strict formal model-based development and a combination of formal and traditional unit-test based methods.

(Malik & Roop, 2015) discuss code generation from method-B. We do not cover formal verification of automated code generators in this paper.

We address the issues raised above through compositional integration of these functions on statechart models. This paper provides details on, the design, setup and results of hardware-in-the-Loop (HiL) tests, and runtime validation carried out on a fully functional prototype CHS.

Specific contributions of the paper are the following:

(a) Elevating compositional formal verification from DEDS to hybrid dynamics by converting simpler automata modules from our statechart decomposition into HA: the graphical models facilitating collaborative development. Kim and Lee (2003) and Malik and Roop (2019) discuss code generation from HA. We overcome over-approximation and risk of false transitivity common to direct abstraction methods of continuous state-space.

(b) Use of the decomposition from (a) to configure compositional HiL tests and MiL tests (reported earlier) for testing at different abstraction and functional levels of the CHS.

(c) A scalable strategy to generate sound and complete (relative to coverage criteria) tests for CHS using (a), reducing the possibility of error introduction.

(d) Deriving compositional runtime validation tests from specifications in (a).

Some implementation - level faults of the integrated system could be identified and corrected in this approach.

A fully functional passenger elevator was built with safety features as CHS test-bed. The elevator controller, designed and implemented first on PLC was easily migrated to FPGA hardware, demonstrating the flexibility of our approach. Control implementation is distributed: the discrete-event control, modularly in an FPGA and the continuous-variable control of motors as Simulink blocks on a PC.

METHODOLOGY, RESULTS AND DISCUSSION

Compositional modelling of hybrid dynamics

In hybrid dynamical systems (Alur *et al*., 1995; Henzinger, 1996; Chaochen *et al*., 2005), states change

instantaneously (possibly discontinuously) for discrete transitions and following differential equations, subject to defined restrictions resulting from physical circumstances or the interaction of continuous dynamics with discrete control. Continuous-variables pose the greatest challenge in verifying hybrid systems. The state spaces become infinite with infinite time actions.

We demonstrate how hybrid automata (HA) (Henzinger, 1996) can model hybrid dynamical systems, through a simple example.

The HA in Figure 1(a) models hybrid dynamics of a moving object. When it is in acceleration mode the node ACCLN is active. The continuous acceleration of

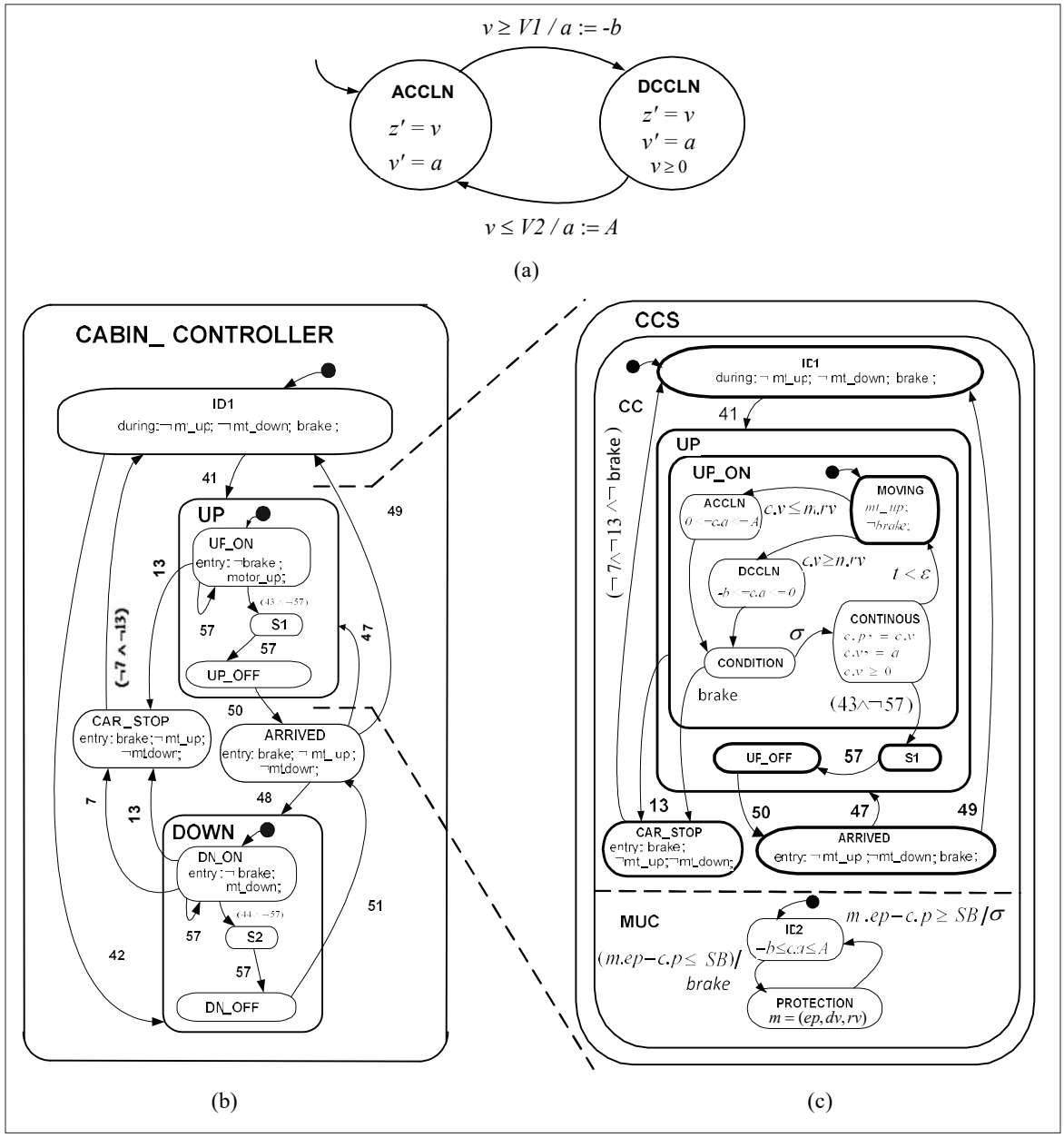

Figure 1: Hybrid automata and hybridisation of statecharts.
(a) Hybrid automaton for controlled linear motion of an object; (b) cabin controller with discrete-event dynamics; (c) hybrid augmentation of statechart state 'UP' of the cabin controller.

the object is described by the set of differential equations $z' = v$ and $v' = a$, where z is the distance travelled. When it attains a speed of $v = V1$ the ACCLN is deactivated and node DCCLN becomes active. It starts to decelerate with $a = -b, b > 0$. When the speed falls below $v = V2$ the state DCCLN is deactivated and ACCLN becomes active again.

Behavioural specifications on the system can now be defined on this model: e.g., the Differential Dynamic Logic. (dL) Platzer, (2010) formula [object] $(v \geq 0)$ expresses that the object always has a positive velocity. In general, $[\alpha]\Phi$ specifies that property 'Φ' holds in all future executions of hybrid system 'α'. A control objective could be to determine corresponding parameter constraints that guarantee the validity of such a formula and a verification objective could ascertain the satisfaction of those constraints in implementations.

An HA, α, however, cannot be decomposed into sub graphs, α_i such that the formula $[\alpha_1]\Phi_1 \wedge [\alpha_2]\Phi_2$ becomes equivalent to $[\alpha]\Phi$, because of the loose connecting edges between the sub graphs α_i (Platzer, 2010). For instance, the automaton in Figure 1(a) cannot simply be verified by proving [ACCLN] $\Phi \wedge$ [DCCLN] Φ, since the trigger events between the nodes cannot be accommodated in the decomposed components. Hence, HA are not directly compositional.

Statechart with hybrid augmentation for compositional verification of elevator control

In our previous studies (Dewasurendra, 2006; 2013; 2017; Dewasurendra *et al*., 2011; Vidanapathirana *et al*., 2011; 2013), we gradually developed a compositional formal verification strategy for complex reactive systems starting from statechart - based specifications for DEDS. Our statechart decomposition results in sequential components (finite state automata) that accept parallel composition when the plant possesses concurrently evolving subsystems (AND states in a statechart). Previously we concerned more about the compositional verifying of the correctness of decomposition and using the decomposition for distributed supervisory control implementation. More importantly, this decomposition potentially reduces the complexity of the image computation problem in model checking.

The elevator-cabin controller module developed in this process is shown in Figure 1(b). The language generators (LG) developed in this process for the states, 'Cabin Controller' and 'UP', and the port-structure between them are given in Figures 2 a, b and c, respectively. The port-structure has been developed to perform a controllability check on interaction specification between two hierarchically adjacent states (Dewasurendra, 2006) and to impose local supervisory control. In this paper we augment states representing real actions of the formal language generators and corresponding port-structures with continuous-variable dynamics to build HA. Figure 1(c), explains what would be obtained if instead, the continuous-variable dynamics were included in the statechart itself. In developing the hybrid control we closely follow the strategy used in the cooperation protocols of European Train Control System (ETCS) (Platzer, 2010).

Elevator-cabin movements consist of discrete-event dynamics, continuous variable dynamics (differential or difference equations), and their hybrid interactions. The elevator cabin control system, (CCS), was designed for the hybrid control and runtime validation of the cabin. The CCS consists of two concurrent controllers as shown in Figure 1(c): the movement update controller, (MUC), and the cabin controller, (CC). The MUC gives movement updates (MU) to CC based on the current state of the cabin dynamics. Cabin is allowed to move within its current MU, which can be updated dynamically by the MUC. Hence CC needs to regulate the movement of the cabin such that it always remains within its MU given by the MUC. The expected cabin speed profile is predetermined, specifying comfort levels of passengers inside: smooth acceleration, constant speed and deceleration before the cabin stops at a requested floor.

To illustrate our approach, we expanded the discrete sub states of 'UP' state of Figure 1(b) as HA in Figure 1(c) and introduced a control interface to represent the MUC to obtain the HA - based statechart state CCS (Cabin-Control-System) in Figure 1 (c). The discrete events of Figure 1(b,c) are described in Table 1.

MU is a vector $m = (ep, dv, rv)$, a dL constraint as follows: beyond the latest updated end point $m.ep$ along the hoist way, the cabin is not permitted by the MUC to have a velocity greater than the desired velocity, $m.dv$. The cabin should try not to exceed the recommended velocity $m.rv$ during the move, while short periods of slightly higher values are allowed. Figure 2(d) shows an example of a possible cabin speed profile in conjunction with the current value of m that changes over time due to the control actions of CC. The moving vectors are updated as m_i.ep, m_i.dv, m_i.rv, for periods i = 1, 2,., n. The computation of these constraints are explained later.

Movement-update control MUC

Given the required speed profile of the cabin along the hoist way, the MUC provides timely information to the CC to regulate the velocity of the cabin by comparing the actual and recommended speed values (see output transitions from the state, MOVING, in Figure 1(c)).

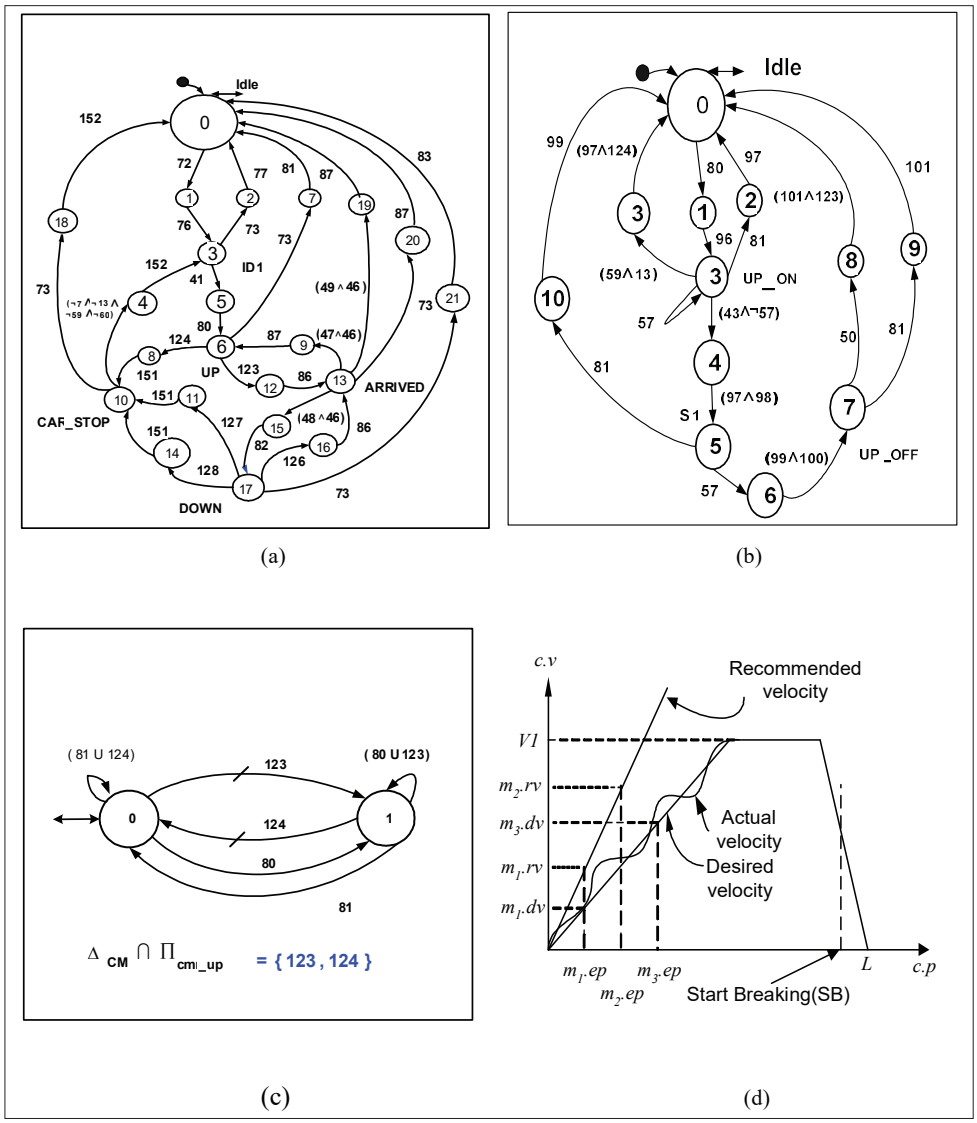

Figure 2: Communicating language generators and cabin speed regulation.
(a) Cabin controller state language generator; (b) UP state language generator; (c) port-structure developed to represent communication between cabin controller and UP states and (d) CCS moving permission update pattern

Getting back to the construction proposed in this paper, we start off from language generators in Figures. 2(a) and (b). The states representing real actions in the physical system are, UP, DOWN, ARRIVED, CAR_STOP and UP_ON. These are the states to be augmented by embedding HA: we embed the UP_ON hybrid automaton from the statechart in Figure 1(c) in the UP_ON state of the language generator in Figure 2(b). For states in Figure 2(b) corresponding to basic states of Figure 1(c), the differential equations from the latter are embedded in the former.

The next step is to embed the HA, MUC. This has to go into a state of the port-structure in Figure 2(c). Since the conditions specified for continuous-variables in MUC are valid in state '0' of the port-structure, we embed MUC in state '0'.

The collapsed versions (in which embedded hybrid-automata/differential equations are removed from their states) of these hybrid embedded language generators and port structures were used to perform discrete-event based compositional verification of the control (Dewasurendra, 2006; Vidanapathirana *et al*., 2011, 2013). Distributed verification of continuous variable dynamics carried out using embedded HA is described next. In latter verification we closely follow the work of Platzer in ETCS (Platzer, 2010) using dL.

Differential dynamic logics for verification of continuous-variable dynamics

A proof calculus for Differential Dynamic Logic (dL) (first order dynamic logic for reals) and its temporal extension, developed by Harel (1979) and Platzer (2010) uses discrete/ differential induction on differential invariants/variants for compositional verification of HDS. The proof of calculus is complete relative to handling differential equations. To achieve scalability, the invariants/variants are compositionally computed in proof loops: dL is closed under logical operators (Chaochen *et al*., 2005). However, the use of sequential HP to express system dynamics limits this to sequential composition. We elevate this to parallel composition.

In dL the model formula [α]ϕ expresses that all states reachable by the hybrid system, α satisfy the dL formula Φ and ⟨α⟩ϕ that at least one state reachable by α satisfies ϕ. Typically, in first order dL, where quantifiers over reals are allowed, $\exists p[\alpha]\langle\beta\rangle\emptyset$ says that there is a choice of parameter p (denoted $\exists p$) such that, for all possible (future) behaviours of system α (denoted [α]) there is some reaction of system β (denoted $\langle\beta\rangle$) that ensures $\emptyset$.

As a simple demonstration of compositionality achieved through the use of logical operators, $\exists p([\alpha]\emptyset \wedge [\beta]\varphi)$ says that there is a choice of parameter p that makes both $[\alpha]\emptyset$ and $[\beta]\varphi$ true, simultaneously.

For elevating their sequential composition to admit parallel composition using parallel HA, consider systems α and β above as resulting from the decomposition of an AND state of a statechart model: this permits concurrent evaluation of $\exists p[\alpha]\emptyset$ and $\exists p[\beta]\emptyset$. In addition, when model-checking for these properties, the granularity of decomposition products of the statechart can be kept as fine as desired, thus making real quantifier elimination manageable in practice: cf. classical state-space abstraction of CVDS into convex polytopes. This extends our compositional formal verification methodology from DEDS to CVDS.

Differential dynamic logic for runtime validation

Still using the hybrid augmented statechart in Figure 1(c) the current state, $c = (p, v, a)$ is given by the current position *p*, speed *v* and acceleration *a*, which is updated in real-time by the interaction of parallel statechart states, CC, and MUC controlling and monitoring cabin movement along the hoist way. In order to meet the speed specifications, *m*, posed on the cabin [Figure 2(d)], we need to determine sufficient conditions that force $c.v$ to always respect its current MU, satisfying,

$$c.p \geq m.ep \rightarrow c.v \leq m.dv \quad \text{...(c1)}$$

The requirement (c1) expresses that the cabin velocity $c.v$ does not exceed the desired speed limit $m.dv$ after passing the latest updated end point (i.e., $c.p \geq m.ep$). The $m.ep$ and $m.dv$ values are determined for the current MU by MUC based on state information received from CC. For example, in tall buildings, a high speed cabin needs to reduce the velocity while arriving at designated points close to the destination. Our model captures this by updating the speed component $m.dv$ of MU, m appropriately to achieve the predetermined speed profile. Unlike collision with other trains in the case of high speed train control in ETCS, here the principle issues concern handling the sudden requests for embarkation, debarkation, emergency stoppages, between-level arrests of movement due to failure, malfunctioning of doors and evacuation of a running lift cabin. Depending on the current state of the cabin, the requests are either accommodated or ignored. If accommodated, then a new MU which can satisfy all the pending accommodated requests is computed by MUC and corresponding information shared with CC.

Coming back to the operational mode depicted in Figure 1(c), when user requests are made for upward movement, the CC initiates the relevant upward movement under the current MU. In parallel, the MUC starts its execution. With the user request for upward movement (up_req = on) the following state transitions take place at the CC super-state:

$$IDLE \rightarrow UP \rightarrow UP\_ON$$

Depending on the value of $c.v - m.rv$ the controller enters either ACCLN or DCCLN. Upon arriving at the state CONDITION, brakes have to be applied if $m.ep - c.p \leq SB$. The point SB (start braking) is the point at which brake has to be applied, indicated by the action, 'brake' to ensure correct alignment of the cabin at a given floor. In operation, solving a set of differential equations (Perko, 2006) is done using real-time clocks with appropriate time steps Δt. Necessary actual position information to derive the $c.v$ and $c.a$ is supplied by the encoder attached to the cabin motor.

The cabin motor control is supervised by the hybrid controller to stay within the performance specification, ensuring stress free and accurate cabin control.

Whenever the runtime system parameters deviate due to un-modelled factors or physical system degradation, MUC provides timely feed-back to CC to do necessary regulation; hence,

$$CCS \equiv (MUC \cup CC)^*$$

MUC and CC being independently distributed hybrid components running in parallel. '$\cup$' represents nondeterministic choice and '*', repetition. We now make the following claim:

Proposition 1. Our HA - based formal verification method is compositional, sound and complete relative to first-order logic of differential equations.

Proof:
Our construction derives from the semantics-preserving decomposition strategy for statecharts and compositional verification of DEDS (Dewasurendra, 2006; Vidanapathirana *et al*, 2011; Vidanapathirana, 2019).

HA are embedded in a subset of automata representing real actions of this decomposition [see Figure 2(b)].

Whereas the discrete-event dynamics are represented by collapsing the embedded HA, the continuous variable dynamics are represented by the interaction of HA distributed among DEDS states.

Since the HA can be translated unambiguously to HP, Platzer's first-order logic of differential equations (FOD) analysis can be directly applied to continuous-variable dynamics embedded in HA to provide a verification complete, relative to FOD (Theorem 2.3 of Platzer, 2010).

Given that dL calculus is sound (Theorem 2.1 of Platzer, 2010), we now have a sound and complete verification system that is compositional and still based on HA.

QED

This lays the foundation for us to develop the remaining results in this paper.

Getting back to determining sufficient conditions that force $c.v$ to always respect (c1), whereas the actual control of components (e.g., motors) is based on detailed models in the verification and parameter discovery process for runtime validation, it is difficult to use highly detailed plant models, and hence, following the approach in Platzer (2010), we approximate plant dynamics by a ranged choice for effective cabin acceleration between its lower and upper bounds, -b and A, respectively: cf. Figure 1(c).

We find constraints that ensure the safety of the system in operation (discrete and continuous invariants) using iterative refinement process of Platzer (2010).

This process is explained on language generator for the state UP [Figure 2(b)].

The formula F_D is found as a discrete invariant of $\varphi \rightarrow [\alpha^*]\emptyset$, where, safety specifications are required, $\emptyset = \neg DOWN$ to be true in all executions, α, of the state UP if it is true in the initial state $\varphi =$ UP. Idle. Here α is the coding of the Kripke structure derived from Figure 2(b).

Using discrete induction to find F_D:
F_D is a discrete invariant of $\varphi \rightarrow [\alpha^*]\emptyset$ if the following formulas are valid:
1. $\varphi \rightarrow F_D$ (initialisation) and
2. $F_D \rightarrow [\alpha]F_D$ (induction step).
F_D is sufficiently strong if $F_D \rightarrow \emptyset$ is valid.

F_D can be obtained as $\omega_r \geq 0$, where ω_r is the actual motor speed in cabin upward motion.

The state '3' (UP_ON) in Figure 2(b) corresponds to the state UP_ON in Figure 1(c), and hence the hybrid automaton in the latter gets embedded in the former. Let us consider the state 'CONTINUOUS' in Figure 1(c).

The formula F_{Dif} is found as a continuous invariant of $\varphi \rightarrow [D \wedge H]\emptyset$, when, for instance, we want to give the safety specification as $\emptyset =$ encoding of $(c.p \geq m.ep \rightarrow c.v \leq m.dv)$, if the following formulas are valid:

1. $\varphi \wedge H \rightarrow F_{Dif}$ (initialisation) and
2. $F_{Dif} \rightarrow [D \wedge H] F_{Dif'}$ (induction step).

As in the discrete case, F_{Dif} is sufficiently strong if $F_{Dif} \rightarrow \emptyset$ is valid.

Here, $H = in(CONTINUOUS)$ defines the domain of continuous variation and D is the set of differential equations,

$$c.p' = c.v; c.v' = a; c.v \geq 0.$$

The differential saturation algorithm of Platzer *et al*. (2009) is used to progressively refine the domain, H, by differential invariants, F_{Dif}, until $H := H \wedge F_{Dif}$ becomes itself an invariant strong enough (reaches a fixed point, $F_{Dif} := H \wedge F_{Dif}$) to imply $\emptyset$.

During runtime, the relevant parameter values are fitted into the invariants, F_D and F_{Dif}, to perform an automatic validation of the verification proof.

Compositionality of modular X-in-the-loop (XiL) testing

Our architecture permits both function - based decomposition (subsystems in plant control: cabin movements, cabin door movements, plant inputs, displays, etc.) and abstraction - based decomposition (Vidanapathirana *et al*., 2013), thus facilitating modular integration testing in addition to function - based (XiL) testing performed separately, if compositionality can be proven. We now demonstrate the compositionality and soundness of modular tests, and that they are complete relative to coverage objectives.

A. Modularity and compositionality of tests

Observation 1: Modular HiL tests were done for an FPGA, which carried the complete controller for elevator operations: interfacing necessary for the tests was done by selecting only the relevant input/output ports of the module concerned, as different from testing an isolated control module (ECU) implemented on dedicated hardware.

Observation 2: Real-time simulation was done by selecting the inputs/outputs for the relevant module on a complete plant simulator; not for a separate module.

Now we make the claim on compositionality of our modular tests.

Proposition 2: The control modules implemented on the system controller can be tested compositionally.

Proof:
Event communication protocols between adjacent FSA in the semantics preserving translation of statecharts to a set of communicating FSA have been modelled as port automata in Dewasurendra (1986);

Each FSA composed of prioritised synchronous composition with its respective port structures constitute a component in the decomposition. They can in turn be composed to retrieve the original statechart. Hence, the components are compositional in their semantics.

We have formally verified the system controller of our target system using a compositional modular verification methodology developed for the MoC used (Dewasurendra, 2006, 2013; Vidanapathirana, 2019).

Hence observations 1 and 2 above permit to conclude that the tests are in fact compositional under the same assumptions made in specifying and verifying the system control model using the MoC of choice.

QED

B. Soundness and completeness of tests

Proposition 3: Our HA - based test generation method is compositional, sound and complete relative to first-order logic of differential equations.

Proof:
By Proposition 1 in the previous section, our HA based formal verification method is compositional, sound and complete relative to first-order logic of differential equations.

Case 1: Tests for discrete dynamics

Following Ferrante *et al*. (2016), we run NuSMV (Cimatti *et al*., 2015) programmes of language generators resulting from our statechart decomposition (after collapsing embedded HA) against negations of CTL (Harel, 1979) formulas representing individual test objectives (criteria) to generate counter-examples that become tests covering required test criteria, thereby overcoming the limitations of directly translating statecharts as input for algorithms 1 and 2 of Ferrante *et al*., (2016).

Tests are sound because the verification based on these models is sound.

Since the decomposition products can be brought down as close as desired to basic states, the counter-examples are guaranteed to be found in finite runs of NuSMV.

Hence, the tests are guaranteed to be complete by coverage criteria.

Case 2: Tests for continuous dynamics

In this case we generate tests by translating the HA we extract from language generators [e.g., Figure 2(b)] into the input language HYDI (an extension of the language of NuSMV) of the model checker HYCOMP (Cimatti *et al*., 2015). The rest of the process is parallel to that of Case 1 and the proof follows the same arguments therein.

QED

In order to conduct CHS research, a prototype passenger elevator with 150 Kg payload was built inside the laboratory (Dewasurendra *et al*., 2011). Essential operational and safety mechanisms were integrated to the elevator, with sensors and actuators that are invoked in different operational modes and a number of interacting sub systems operating in parallel to make it sufficiently complex, yet simple enough to serve as a test-bed.

We will now give details of configuring hardware-in the-loop tests (Dufor *et al*., 2005) for integration with V&V on this test-bed.

Configuring hardware-in-the-loop (HiL) tests for integration with V&V

Elevator controller modelled using statecharts in Matlab/Simulink/Stateflow was converted to equivalent HDL code Verilog and implemented in Xilinx Spartan 3AN FPGA (ECU) (Vidanapathirana *et al*., 2011) with 3.6 V DC i/o interface circuits. For HiL tests the ECU was wired to receive user inputs and dSPACE hardware signals: interface circuits were used between ECU and dSPACE hardware platforms to achieve electrical isolation and voltage level shifting.

HiL testing (Dufor *et al*., 2005) requires physical connections between the ECU and the simulated plant to ensure that communication with the ECU is the same as in real system. The decomposition of Simulink/ Stateflow control specification of the system done for compositional formal verification (Vidanapathirana *et al*., 2011) helped in configuring compositional HiL tests: to compile build and simulate in real-time only a part of the system model at a time and isolating the inputs and outputs corresponding to the module being tested (the cabin module) from the complete system controller (implemented on the ECU).

The passenger cabin is driven by a three-phase induction-motor. Dynamic models of the cabin motor, the inverter based speed/torque control system and the floor sensor operation in the elevator were implemented in Matlab/Simullink with real-time workshop using a fixed time step and uploaded to the TMS 320F DSP processor - based dSPACE platform as the plant model.

Matlab/Simulink/dSPACE and Controldesk software running on a PC were connected to the dSPACE hardware for online monitoring and regulation of real-time simulations.

A. Test generation using NuSMV

The tests for required test coverage criteria were generated using the process described earlier as follows:

Case 1: Test for discrete dynamics:

Consider state coverage criterion with respect to the 'UP' language generator [Figure 3(b)] for discrete dynamics: for each state 'S_i' of 'UP', ∃ at least one run covering S_i. Let S_i = <UP_OFF>, which corresponds to responding to a stop request made while the cabin is moving up.

The Kripke structure (which is used as input format to model-checker NuSMV) corresponding to 'UP' has a run covering 'S_i' if

(i) the state 'S_i' is reachable from the initial state, 'UP_ Idle' and
(ii) ∃ a state 'S_j' which is reachable from 'S_i', at which 'stable' is satisfied.

This is expressed in the CTL formula,
$EF(reachable(S_i) \wedge EF\ stable)$.

We proceed now to translate 'UP' to a NuSMV programme and then run it against $(\neg EF(reachable(S_i) \wedge EF\ stable))$.

Then a counter example is generated, which serves as a sound test sequence for the coverage criterion considered.

Since we only convert to NuSMV programmes, individual language generators (finite automata) resulting from our decomposition, which are several degrees of magnitude smaller in size and complexity than the parent statechart, the counter examples would always be found

in a finite run, hence our method of test generation is complete (by the coverage criteria).

The resulting test sequence is then in format (<present state>, <input>, (<next state>, <output signal>)), given as:
(UP_Idle_at_Floor1, U1, UP, mt_UP) (UP, _, UP_ON, _) (UP_ON, '43', S1, _) (S1, sensor_f2, UP_OFF, '57')) (UP_OFF, ARRIVED, ¬mt_UP) (ARRIVED, UP_Idle_at_Floor2, _).

Case 2: Test for continuous dynamics

The development is similar to the discrete case, except that UP language generator augmented with continuous dynamics to make it a HA, as explained earlier.

We then translate the resulting 'UP_{hybrid}' to an HYDI programme and run it against the negation of c1, (), to generate a counter example that serves as a test sequence.

While running the tests, plant control parameters inside dSPACE real-time simulation were examined using dSPACE Control desk software running in PC. The real-time simulation was controlled through online adjustment of motor parameters using the same.

A fixed time step was required for the real-time simulation on dSPACE hardware platform. Overrun situations for the processor could arise when the allowed time gap is not adequate for the execution of a complete programme cycle and updating the data registers. Larger time steps may not clearly describe the plant dynamics. We selected 1 ms time step for the plant simulations after trying 0.2, 0.4, 0.6, 0.8, 1.0, 2.0, 3.0 and 4.0 ms. This choice had to be compatible with the permitted reaction latency, ϵ, in Figure 1(c).

Induction-motor dynamic model

To analyse variable speed drives, a dynamic motor model was developed using 'space vector' analysis (Ogbuka 2009; Shah *et al*., 2012). For fast dynamic response, precise speed regulation and good dynamic performance, field oriented control was used for the cabin (induction) motor (Ogbuka, 2009; Vidanapathirana *et al*., 2013).

Details of Matlab/Simulink model for IM vector control drive, speed controller design and current controller design, are given in Vidanapathirana (2019).

When the cabin moves along the elevator shaft, respective floor sensors give control signals to the controller.

The motor shaft rotational angle, θ_f, required to travel a linear distance between two adjacent floors for the prototype elevator, was computed as 10,085 deg.

The speed reference input to the controller for a linear distance between two adjacent floors is shown in Figure 3(c) [cf. moving permission update pattern of Figure 2(d)].

For speed regulation, maximum motor speed was set to 100 rad/s to achieve a linear speed for the cabin; acceleration and deceleration were set at 8.9 rad/sec^2.

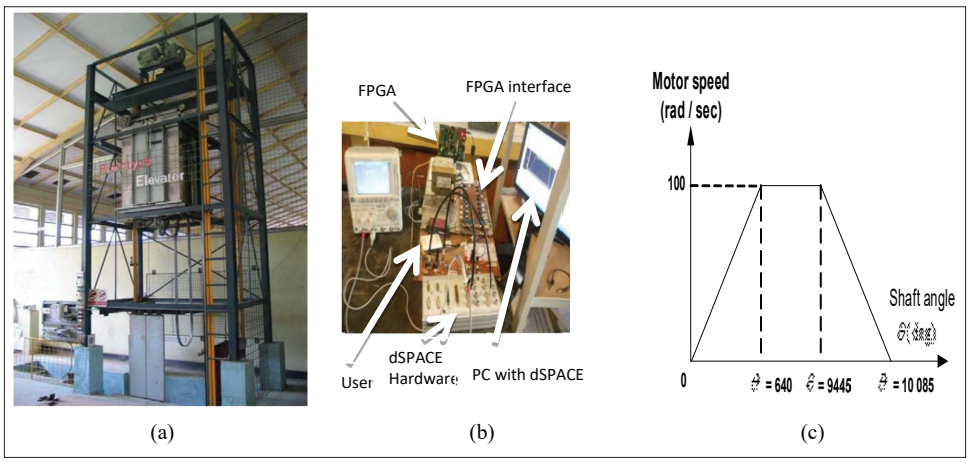

Figure 3: Prototype elevator HiL test setup.
(a) Prototype passenger elevator; (b) HiL test arrangement; (c) speed reference

HiL test results

During testing, an arbitrary load torque was applied to the motor during cabin upward movement. Reference inputs were the passenger car speed, acceleration and deceleration. Variation of key system parameters during the test are shown in Figures 4 (a) to (f).

The test results revealed that the actual speed followed the reference speed closely; so did the load torque and the generated torque as shown in Figures 4(e) and (f).

In comparison to HiL tests, for MiL tests (Vidanapathirana *et al*., 2013) both ECU and plant models were implemented in Matlab/Simulink: simulation parameters could be selected with a greater degree of liberty and time step could be either variable or fixed, variable time steps giving a faster simulation.

Rectifying implementation level faults

Formal verification confirmed logical conformity to specifications, *HiL* tests validated the verification, but the observed behaviour in the controlled elevator plant at the implementation level differed sharply from specifications.

Avoiding a detailed examination of the installed software/hardware logic, focus shifted quickly to electrical/electronic installation.

An EMI problem was discovered: FPGA inputs rated at 3.6 V DC are sensitive even for electrical interferences as small as 0.5 ~ 1 V DC. The 24 V DC power lines and 3.3 V DC FPGA control lines were in close proximity. When the 24 V DC relays used to step-down the sensor voltage to 3.6 V DC and activate FPGA input ports were switched on and off, electrical interference caused spurious outputs from the FPGA, producing erratic behaviour.

The 24 V DC relays were replaced by an optical trigger circuit with electrical isolation. FPGA control signal wiring was rearranged by minimising the effect from the power lines. FPGA control wiring was replaced by shielded fieldbus cables with adequate grounding to overcome electrical interference. This corrected the EMI problem.

This would have been discovered in power-HiL tests. This also justifies runtime validation for monitoring crucial system properties (closed-loop stability, robustness margins, or underlying theoretical assumptions), violation of which could compromise safety of operation in CHS.

CONCLUSION

A compositional model-based strategy for designing and maintaining correct-by-design controllers for CHS was developed and demonstrated. The complexity of the design, formal verification, test design, testing and runtime validation has been reduced to levels usable in industrial practice by providing a problem decomposition that permits the full exploitation of parallelism inherent in the physical system, thus enabling concurrent computation. Soundness of the strategy is maintained by using consistent formal models at each stage and completeness of solutions relative to the specifications is guaranteed up to the levels of abstraction used.

Table 1: Discrete events description

Event No.	Event	Description
7	governor	Limit switch –governor activated
13	motor_overload	motor_overload
41	up_req	Upward movement request
42	down_req	Downward movement request
43	up_immed	Stop request at immediate upper floor
44	down_immed	Stop request at immediate down floor
47	more_up_req	There are requests which need further car upward movements
48	more_down_req	There are requests which need further car downward movements
49	no_req	The car is stopped and there are no further requests made
50	en(UP_OFF)	Entering the UP_OFF state of the statechart
57	t_pf	While car is moving up it is stopped at a given floor due to a stop request

With minimal additional work, system parameters for degraded performance could be computed based on an online simulation of the currently configured system as in (Vinczea *et al*., 2006; Pugi, 2007), or using the runtime validation system as an online monitoring system (Davydov and Keyno 2016).

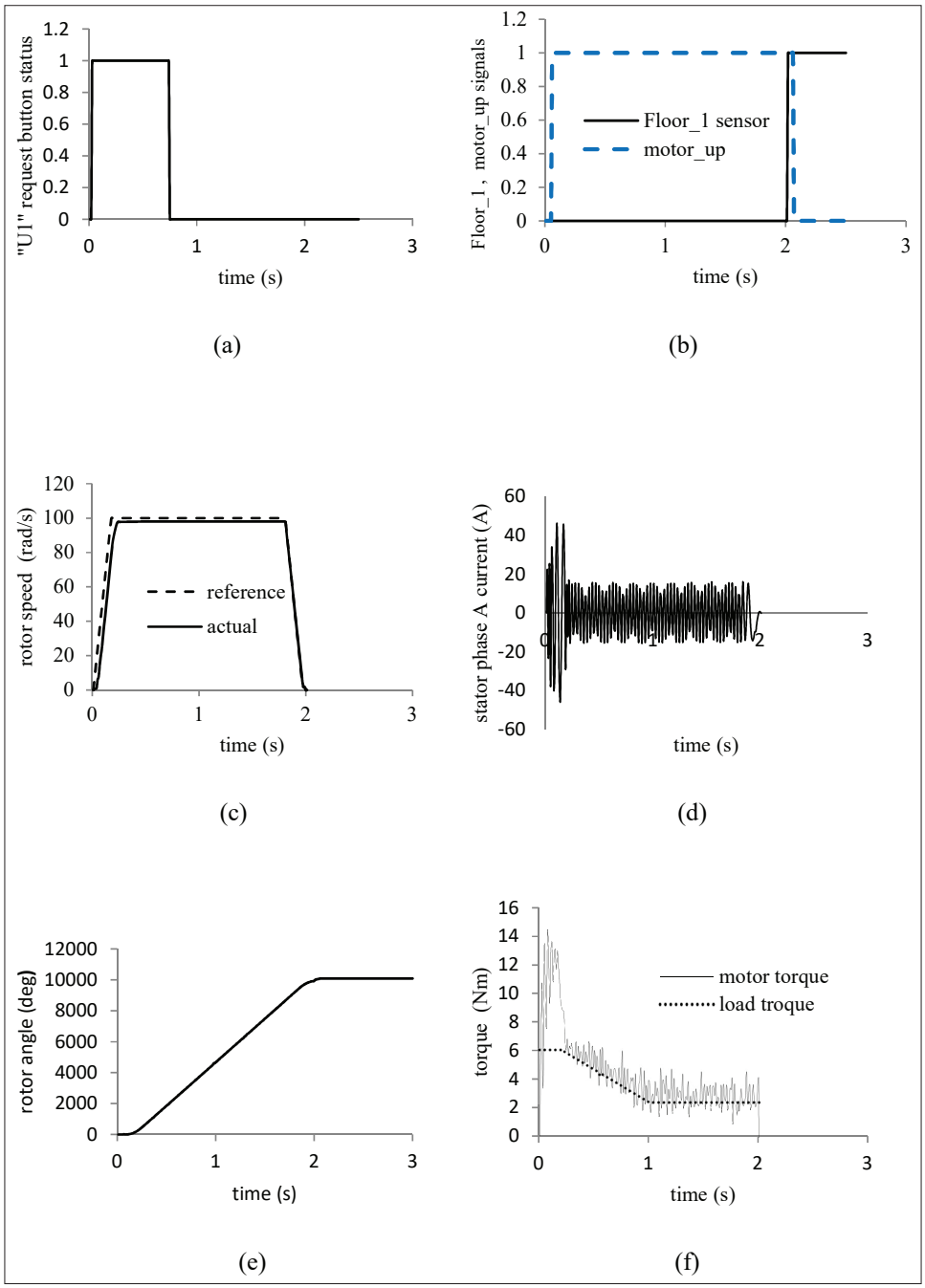

Figure 4: HiL test results.
(a) U1 passenger request; (b) floor_1 and motor_up signals; (c) reference and actual cabin speeds; (d) stator phase A current; (e) rotor angle; (f) motor generated and applied load torque.

Acknowledgement

The authors acknowledge the University of Peradeniya for the Research Grant (RG/2006/32/E), the staff of the Department of Electrical and Electronic Engineering, sub-warden of the Ramanathan Hall of Residence and the colleagues from the CECB and Schneider Electric Co. Ltd.

REFERENCES

Abrial J.-R. (2010). *Modelling in Event-B. System and Software Design*. Cambridge University Press, Cambridge, UK. DOI: https://doi.org/10.1017/CBO9781139195881

Aceto L., Inglfsdttir A., Larsen K.G. & Srba J. (2007). *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, Cambridge, UK. DOI: https://doi.org/10.1017/CBO9780511814105

Alur R. (2011). Formal verification of hybrid systems. *Proceedings of the 9th ACM international Conference on Embedded Software (EMSOFT' 11)*, Taipei, Taiwan, 9–14 October, pp. 273-278. DOI: https://doi.org/10.1145/2038642.2038685

Alur R., Courcoubetis C., Halbwachs N., Henzinger T., Ho P.H., Nicollin X., Olivero A., Sifakis J., Yovine S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science* **138**:3 – 34 DOI: https://doi.org/10.1016/0304-3975(94)00202-t

ARTEMIS Project MBAT (2005). Combined Model-based Analysis and Testing of Embedded Systems. Available at *http://www.mbat-artemis.eu/home/*, Accessed 15 January 2018.

Belta C., Boyan Y. & Ebru A.G. (2017). *Formal Methods for Discrete-Time Dynamical Systems*, Springer, USA. DOI: https://doi.org/10.1007/978-3-319-50763-7.

Bijl V., Rensink M.A. & Tretmans J. (2003). *Compositional Testing With Ioco*. LNCS Book Series, volume 3395, pp. 86–100. Springer, Germany. DOI: https://doi.org/10.1007/978-3-540-24617-6_7.

Bringmann E. Kramer A. (2008). Model-based testing of automotive systems. *Proceedings of the 2008 International Conference on Software Testing, Verification and Validation (ICST'08),*Lillehammer, Norway, 9–11 April, pp. 485–493. DOI: https://doi.org/10.1109/ICST.2008.45.

Bringmann E. & Kramer A. (2006). Systematic testing of the continuous behaviour of automotive systems. *Proceedings of the 3rd International Workshop on Software Engineering for Automotive Systems (SEAS '06)* (Compiled by ACM), Shanghai, China, 23–23 May, pp. 13–20. DOI: https://doi.org/10.1145/1138474.1138479.

Chaochen Z., Ji W. & Ravn A.P. (2005). A formal description of hybrid systems. In: *Hybrid Sytem III, Lecture Notes in Computer Science, LNCS 1066* (eds. R. Alur, T.A. Henzinger & E.D. Sontag) pp. 511–530. Springer-Verlag, Germany. DOI: https://doi.org/10.1007/BFb0020972.

Cimatti A., Griggio A., Mover S. & Tonetta S. (2015). HyComp: an SMT-based model checker for hybrid systems. In: *Tools and Algorithms for the Construction and Analysis of Systems*, *Lecture Notes in Computer Science*, *LNCS 9035* (eds. C. Baier & C. Tinelli), pp. 52–67. Springer, Berlin, Germany. DOI: https://doi.org/10.1007/978-3-662-46681-0_4.

CRYSTAL (2016). CRiticalsYSTem engineering Acceleration. Available at *http://www.crystal-artemis.eu/*, Accessed 20 January 2018.

Daca P., Henzinger T.A., Krenn W. & Nickovic D. (2014). Compositional specifications for ioco testing. *Proceedings of the Seventh IEEE International Conference on Software Testing, Verification and Validation (ICST '14)*Washington DC, USA, 31 March – 04 April, pp. 373–382. DOI: https://doi.org/10.1109/ICST.2014.50.

Davydov Y. & Keyno M. (2016). Longitudinal dynamics in connected trains. *Procedia Engineering* **165**: 1490–1495. DOI: https://doi.org/10.1016/j.proeng.2016.11.884.

Dewasurendra S.D. (2013). Formal verification strategy for statechart based design of reconfigurable control of high integrity reactive systems. *"ENGINEER", Journal of the Institution of Engineers, Sri Lanka* **46**(2):13–33. DOI: https://doi.org/10.4038/engineer.v46i2.6907.

Dewasurendra S.D. (2017). Fully Abstract Compositional Fix-point Semantics for Statecharts to Model Safety-critical Complex Reactive Systems. (Submitted to *IEEE Trns. in SE*).

Dewasurendra S.D., Abayaratne S.G. & Vidanapathirana A.C. (2011). Development of an elevator prototype for multi-disciplinary research in complex reactive systems. *Annual Transactions*, volume 1– part B. The Institution of Engineers, Sri Lanka,.

Dewasurendra S.D., Abayaratne S.G. & Vidanapathirana A.C. (2011). Develpoment of an elevator prototype for multi-disciplinary research in complex reactive systems. *Proceedings of the 2011 Peradeniya University Research Sessions* (*PURSE '11).* Peradeniya, Sri Lanka.

Dewasurendra S.D. (2006). Statecharts for reconfigurable control of complex reactive systems: a new formal verification methodology. *Proceedings of the First International Conference on Industrial and Information Systems (ICIIS 2006),* Peradeniya, Sri Lanka, 8–11 August. Faculty of Engineering, University of Peradeniya, Sri Lanka. DOI: https://doi.org/10.1109/ICIIS.2006.365736.

Drusinsky D. & Harel D. (1989). Using statecharts for hardware description and synthesis. *IEEE Transactions on Computer-Aided Design* **8**(7): DOI: https://doi.org/10.1109/43.31537.

Dufor C., Aboyrida S. & Belanger J. (2005). Hardware in-the-loop simulation of power drives with RT-lab. *Proceedings of the International Conference on Power Electronics and Drives Systems (PEDS 2005),* volume 2, 28 November–01 December, pp. 1646–1651 DOI: https://doi.org/10.1109/PEDS.2005.1619952.

Ferrante O., Marazza M. & Ferrari A. (2016). Formal specs verifier ATG: a tool for model-based generation of high coverage test suites. *Proceedings of The 8th European*

Congress, Embedded Real-time Software and Systems (ERTS²2016), Toulouse, France, 27–29 January.

Harel D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming* **8**(3): 231–275. DOI: https://doi.org/10.1016/0167-6423(87)90035-9

Harel D. (1979). *First-order Dynamic Logic*. *Lecture Notes in Computer Science LNCS 68*, Springer, New York, USA. DOI: https://doi.org/10.1007/3-540-09237-4.

Henzinger T.A. (1996a). The theory of hybrid automata. *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, New Brunswick, NJ, USA, 27–30 July, pp. 278–292. DOI: https://doi.org/10.1109/LICS.1996.561342.

Hoare C.A.R. (1985). *Communicating Sequential Processes*. Prentice-Hall International, USA.

Kim J. & Lee I. (2003). Modular code generation from hybrid automata based on data dependency. *Proceedings of the 9th IEEE Real-Time and Embedded Technology Applications Symposium*. Toronto, Ontario, Canada.

Liu J. & Liu J. (2014). A formal framework for hybrid event B. *Electronic Notes in Theoretical Computer Science (ENTCS)* **309**(C): 3–12. DOI: https://doi.org/10.1016/j.entcs.2014.12.002.

Malik A. & Roop P. (2015). A unified framework for modeling and implementation of hybrid systems with synchronous controllers. *ArXiv 2015*. Available at *https://arxiv.org.* Accessed 20 January 2018.

Malik A. & Roop P. (2019). System and method for emulating hybrid systems. *US Patent Application Publication, Pub No: US 2019/0179988 A1*, Accessed 21 July 2019.

Moreira A.M., Hentz C., Deharbe D., de Matos E.C.B., Neto J.B.S. & de Medeiros Jr V. (2015). Verifying code generation tools for the B-method using tests: a case study. *Proceedings of the International Conference on Tests and Proofs (TAP'15)*. L'Aquila, Italy, 22–24 July. Also in LNCS, volume 9154. Springer, New York, pp 76–91. DOI: https://doi.org/10.1007/978-3-319-21215-9_5.

Müller O. & Scholz P. (1997). Functional specification of real-time and hybrid systems. *Proceedings of Hybrid and Real-Time Systems* (HART '97), Grenoble, France. Also in LNCS,volume 1201. Springer Verlag. DOI: https://doi.org/10.1007/BFb0014732.

Namjoshi K.S. & Trefler R.J. (2010). On the completeness of compositional reasoning methods. *ACM Transactionss on Computational Logic* **11**(3):16:1–16:22. DOI: https://doi.org/10.1145/1740582.1740584.

Nielsen B. (2014). Towards a method for combined model-based testing and analysis. *Proceedings of the 2ndInternational Conference on Model-Driven Engineering and Software Development*, Lisbon, Portugal, 07–09 January, pp. 609–618.

Ogbuka C.U. (2009). Dynamic modelling and simulation of a asynchronous motor driving a mechanical load. *The Pacific Journal of Science and Technology* **10**(2): 77–82.

Perko L. (2006). *Differential Equations and Dynamical Systems*, 3rd edition. Springer-Verlag, USA.

Platzer A. (2010). *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, Germany.

Platzer A. & Clarke E.M. (2007). The image computation problem in hybrid system model checking. *LNCS 4416*, pp 473–486, 2007. Springer-Verlag, Germany. DOI: https://doi.org/10.1007/978-3-540-71493-4_37.

Platzer A. & Clarke E.M. (2009). Computing differential invariants of hybrid systems as fixedpoints. *Proceeding of the 20th International Conference on Computer Aided Verification (CAV 2008)*, Princeton, USA, 7–14 July. DOI: https://doi.org/10.1007/s10703-009-0079-8.

Pugi L., Fioravanti D. & Rindi A. (2007). Modelling the longitudinal dynamics of long freight trains during the braking phase, *Proceedings of the 12th IFToMM World Congress*, Besançon, France, 17–21 June, pp. 602–608.

Shah S., Rashid A. & Bhatti M.K.L. (2012). D-Q Modelling of 3-phase Induction-motor using Matlab/Simulink. *Canadian Journal on Electrical and Electronics Engineering* **3**(5):

Sloth C. & Wisniewski R. (2011). Verification of continuous dynamical systems by timed automata. *Formal Methods in System Design* **39**(1): 47–82. DOI: https://doi.org/10.1007/s10703-011-0118-0.

Tretmans J. (2008). Model based testing with labelled transition systems. In: *Formal Methods and Testing* (eds. R. Hierons, J. Bowen & M. Harman), pp. 1–38. Springer-Verlag, Germany. DOI: https://doi.org/10.1007/978-3-540-78917-8_1.

Vidanapathirana A.C. (2019). Correct-by-design methodology for the control of safety-critical complex reactive systems. *PhD Thesis*, University of Peradeniya, Sri Lanka.

Vidanapathirana A.C., Dewaurendra S.D. & Abeyaratne S.G. (2013). Model in the loop testing of complex reactive systems. *Proceedings of the 7th International Conference on Industrial and Information Systems (ICIIS)*, Kandy, Sri Lanka. DOI: https://doi.org/10.1109/ICIInfS.2013.6731950.

Vidanapathirana A.C., Dewaurendra S.D. & Abeyaratne S.G. (2011). Statechart based modelling and controller implementation of complex reactive systems, *Proceedings of The 6th International Conference on Industrial and Information Systems (ICIIS)*, 16–19 August, Kandy, Sri Lanka. DOI: https://doi.org/10.1109/ICIINFS.2011.6038120.

Vinczea B. & Tarnaib G (2006). Development and analysis of train brake curve calculation methods with complex simulation. *Advances in Electrical and Electronic Engineering (AEEE)* **5**(1-2): 174–177.