# Event-B Development of Modelling Human Intervention Request in Self-Driving Vehicle Systems

Fahad Alotaibi$^{1,2[0000-0001-8545-907X]}$, Thai Son Hoang$^{2[0000-0003-4095-0732]}$, Asieh Salehi Fathabadi$^{2[0000-0002-0508-3066]}$, and Michael Butler$^{2[0000-0003-4642-5373]}$

[1] College of Applied Computer Sciences (CACS), King Saud University, Riyadh, 11543, Saudi Arabia
[2] School of Electronics and Computer Science (ECS), University of Southampton, Southampton SO17 1BJ, U.K.
`{f.a.alotaibi, t.s.hoang, a.salehi-fathabadi, m.j.butler}@soton.ac.uk`

**Abstract.** In the design of autonomous systems, seamless integration with human operators is crucial, particularly when humans are considered as a fail-safe for intervening in hazardous situations. This study presents an Event-B intervention timing pattern designed to include human drivers' responses when they act as fallback mechanisms in Self-Driving Vehicle (SDV) systems. The proposed pattern outlines specific timings for driver interventions following alerts from SDVs, offering a clear set of expectations and conditions for human drivers during these critical takeover instances. The usability of this pattern is demonstrated through a case study, highlighting its importance for situations that require interventions. Ultimately, it sheds light on the operational aspects of SDVs, ensuring a safe and orderly transition from automated to manual control.

**Keywords:** SDV · Driver Interventions · Event-B · Timing Patterns

## 1 Introduction

Autonomy has advanced the modern way of cooperation between humans and machines. With the rapid development of advanced tools and techniques, the interactions between humans and autonomous machines are becoming increasingly complex [20]. Several classification approaches for defining autonomy have been developed in different domains [19]. The International Society of Automotive Engineers (SAE) [17] classified the autonomy in Self-Driving Vehicles (SDVs) into six levels for performing the Dynamic Driving Task (DDT). Beside fully manual (Level 0), automation levels 1 to 3 (semi-automation) involve a human driver within the DDTs, while automation levels 4 and 5 (high-automation systems) do not engage with a human driver in DDTs [7]. Similarly, Unmanned Aerial Systems (UAS), also known as drones, is organised into four levels, where the inspector (human) is responsible for supervising a system at most levels [16].

The engagement between humans and machines is widely known as the Human-In-The-Loop System (HITLS) [10]. The HITLS can add advanced qualifications to the autonomous system model to make it the top safety pick. According to the Insurance Institute for Highway Safety (IIHS) [12], the SDVs can become '*the safety pick*' when it is integrated with technology to prevent crashes. One of these integrated systems is an HITLS that aims to prevent collisions by creating a digital collaboration space between drivers and machines [4].

Although the SDVs can be built on the HITLS architecture, the behaviour of drivers might have caused fatal accidents. When investigating an Uber self-driving crash, the National Transportation Safety Board (NTSB) [8] found the accident was caused by the internal components of an SDV when the autopilot module failed to detect the later victim. The SDV was implemented to give a human driver control of the vehicle in unmanaged areas; however, the driver was distracted and did not react at the appropriate time.

Automotive companies, such as Tesla and Comma.ai, have used notification mechanisms to ensure the responsiveness of human drivers. For instance, the autopilot software of Comma.ai, known as OpenPilot [9], gives a driver 4 seconds to react when the intervention request is sent. However, if the human driver ignores intervention, the OpenPilot will gradually reduce the speed of the vehicle after 6 seconds until the car is totally stopped.

Modelling intervention requests is crucial, especially when human drivers play a key role in potential interventions. The need arises to develop methods that thoroughly investigate the requirements and assumptions linked to human responses. Understanding the intricacies of cognitive processes and decision-making mechanisms during interventions is vital for ensuring the safety of a system and the collaboration between humans and autonomous systems.

Therefore, this article presents the modelling methodology of time in Event-B [1] using *patterns*. The intervention timing pattern is introduced to formally model the timing properties when the automated machines may ask humans to take control of a system. Our pattern is a specialisation of the *trigger-response* pattern, in which trigger and response are both events (i.e., guarded actions) combining with a the *deadline* pattern [18].

The main advantage of the Event-B model is its support for a stepwise modelling approach by refinements. The second strength of an Event-B model is supported by the toolset Rodin [2], which involves both theorem proving and model checking (ProB) [14]. These advantages of Event-B, make it an appropriate method for formal modelling of complex systems.

The rest of the paper is structured as follows. Section 2 provides background on the Event-B formal modelling language and the introduction to our case study. Section 3 introduces the intervention timing pattern for modelling driver responses. Section 4 illustrates the approach using the case study. Section 5 discusses the advantages and results of using the intervention timing pattern. Section 6 presents related work and Section 7 concludes our article.

## 2 Background

In this section, we first review some background information on Event-B (Section 2.1) before giving the introduction to our case study (Section 2.2).

### 2.1 Event-B

Event-B [1] is a formal method commonly used for system analysis and modelling. Event-B is similar to other formal methods, as it uses concise mathematical language to address the inaccuracy and vagueness of requirements specification. The safety properties treated as invariants are verified, which aims to remove any inconsistency in the verified model.

A formal model in Event-B [1] includes two parts: *contexts* and *machines*. Contexts involve the static parts of a model and provide axiomatic characteristics. A context contains the definition of the carrier sets, constants and axioms that constrain the constants and carrier sets. Machines are the dynamic parts of the model. An Event-B machine involves variables v, invariants I(v) that constrain the variables, and events. An event is *'an atomic transition'* that changes the states of the system. The transition state of an event is constrained through the guards and the actions. For instance, for an event e with parameters t, the guard of the event can be written as G(t,v), and the action of the event can be represented as v := E(t, v). An event e can only be enabled when its guard G (t, v) holds for some parameter t and its affects on variable v is specified by the action E(t,v).

```
e == any t where G(t,v) then v := E(t,v) end
```

One of the principal advantages of Event-B is its utilisation of patterns to address complex modelling challenges. The Event-B pattern serves as a generic structure that can be applied to various aspects of modelling the dynamic behaviours of a system. Furthermore, the machine inclusion plug-in [11] facilitates the transformation of a pattern into a concrete example, thereby easing the application of these generic structures across numerous use cases. Machine inclusion in Event-B introduces two principles: 1) machine inclusion (**includes** clause), and 2) event synchronisation (**synchronises** clause). For instance, inclusion allows for modularising and combining models. A machine m0 could be included in a new machine m1 as follows:

```
machine m1 includes m0 as inclusionName
```

(The keyword **as** allows the inclusion of multiple copies of the same machine with appropriate prefixing). This inclusion means that event e1 of m1 may synchronise with event e0 of m0 specified as follows:

```
event e1 synchronises inclusionName.e0 end
```

Synchronisation means the guards of e1 and e0 are conjoined and their actions are executed simultaneously [11].

### 2.2 The ALC Case Study

The case study examined in this article focuses on the Automated Lane Centring (ALC). The primary function of the ALC system is to maintain the SDV in the centre of its desired/target lane, with the human driver responsible for performing lane change manoeuvres [3, p. 3]. Furthermore, it is crucial to acknowledge that the ALC system does not replace the need for a human driver. Even though it assists with lane centring, the human driver is expected to stay alert and prepared to assume manual control when necessary.

Given the unpredictability of human drivers, there is a possibility that warnings and intervention signals may be disregarded. A study from the Crash Warning Interface Metrics program (CWIM) [13] suggests that after receiving an auditory alert, a driver may take roughly 700 milliseconds to override the system and steer manually. Other research indicates a more extended period, with drivers taking around 10 seconds to refocus and attend to the road [15]. Automotive companies such as Tesla and OpenPilot have thus incorporated notification systems to keep human drivers alert. For instance, OpenPilot [9] gives drivers a 4-second window to react to an intervention prompt. If ignored, OpenPilot emits an auditory warning, and after 6 seconds, it decelerates the vehicle until the SDV comes to a complete stop.

For example, Figure 1 depicts a scenario involving an SDV and the need for driver intervention. Subfigure 1a demonstrates the vehicle accurately positioned within its target lane. However, a noticeable shift towards the right lane line occurs in Subfigure 1b, necessitating corrective input from the driver. Despite this assistance, Subfigure 1c continues the SDV's trajectory towards the right. The situation escalates in Subfigure 1d, where the SDV is precariously close to exiting its lane, underscoring the urgency for immediate driver intervention to prevent a potential hazard.

Furthermore, it is critical to expeditiously elucidate how the SDV system signals a human driver to assume manual control of the vehicle. Figure 2 provides a vital illustration of this integral sequence within the operations of SDVs, with a particular emphasis on the transition from autonomous operation to manual intervention. Upon detection of a scenario necessitating human involvement, the SDV system promptly issues an intervention notification to the driver, concurrently initiating a countdown sequence. In the event of a delayed response from the driver, the system activates an auditory alarm, serving as an additional prompt, and institutes a supplementary waiting period. In the continued absence of driver responsiveness, the system proactively transitions to a mitigation strategy mode, implementing a series of safety measures, including vehicular deceleration. Subsequent to this intervention, the SDV system retains control for a predefined duration, ensuring the maintenance of vehicular stability.

In this article, particular attention is given to the scenarios where human drivers might not respond aptly to the ALC's intervention alerts. The objective is to examine the sequence of a driver's actions when the ALC system prompts manual intervention, as follows:

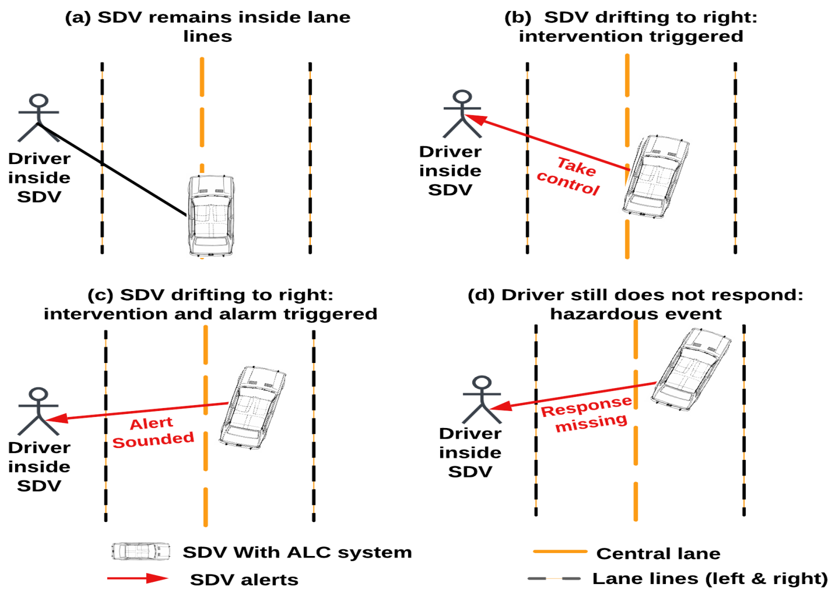
(a) SDV remains inside lane lines
Driver inside SDV
(b) SDV drifting to right: intervention triggered
Driver inside SDV
Take control
(c) SDV drifting to right: intervention and alarm triggered
Driver inside SDV
Alert Sounded
(d) Driver still does not respond: hazardous event
Driver inside SDV
Response missing
SDV With ALC system
SDV alerts
Central lane
Lane lines (left & right)

**Fig. 1.** From stability to hazard: tracing lane-assistance in ALC

1. The ALC system issues a request for intervention and initiates a pre-specified time (countdown) for the driver's response.
2. If the driver does not respond within the specified time, the ALC system triggers an auditory alert to attract the driver's attention.
3. From the moment of the auditory alert, the driver is given a further specific time window in which to react.
4. If the driver still does not react within the specified time, the ALC system proactively reduces the SDV 's speed.

## 3 Intervention Timing Pattern

The intervention timing pattern investigates how a human operator might respond when the automated machine asks for intervention. Our pattern not only models the reaction of a human operator but introduces new requirements and assumptions that need to be considered to make the SDV system safe.

This pattern explains our modelling choice and offers a broad context for understanding key properties such as *time progression*, *the clock (timer)*, *human reaction time* and *alert time*. The primary concept involves using guarded events with time constraints; thus these guarded events can be triggered only when the system reaches a specific time.

The time progression is also designed as *an event*; therefore, there is no need to modify the underlying language of Event-B. The variable *time* is defined as a natural number, which allows time constraints, such as *alert time*, to be expressed as constants or as relationships between different times. Moreover,

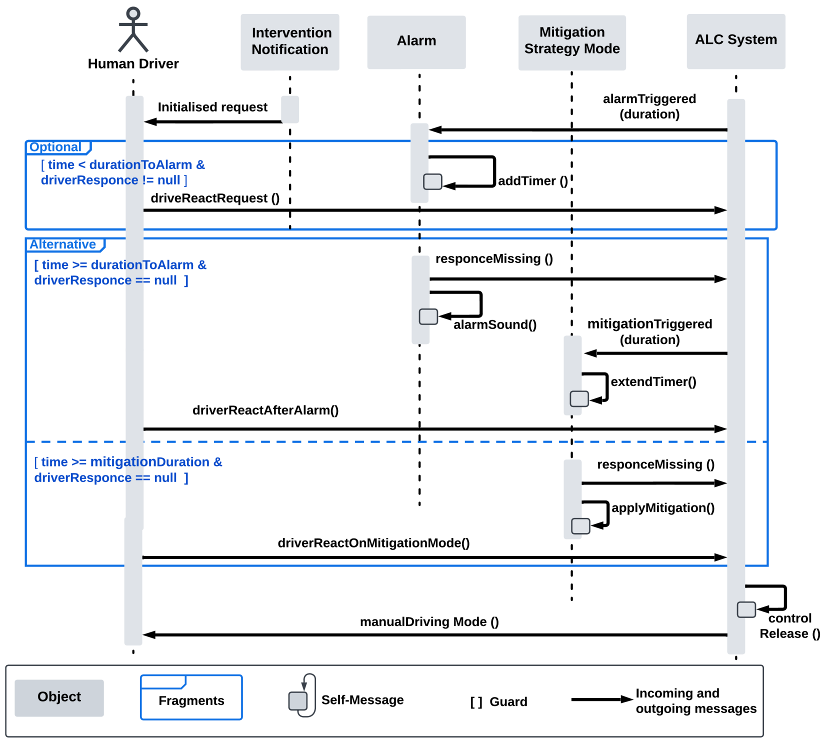
Human Driver
Intervention Notification
Alarm
Mitigation Strategy Mode
ALC System
Initialised request
alarmTriggered (duration)
Optional
[ time < durationToAlarm & driverResponce != null ]
addTimer ()
driveReactRequest ()
Alternative
[ time >= durationToAlarm & driverResponce == null ]
responceMissing ()
alarmSound()
mitigationTriggered (duration)
extendTimer()
driverReactAfterAlarm()
[ time >= mitigationDuration & driverResponce == null ]
responceMissing ()
applyMitigation()
driverReactOnMitigationMode()
control Release ()
manualDriving Mode ()
Object
Fragments
Self-Message
[ ] Guard
Incoming and outgoing messages

**Fig. 2.** Human driver intervention sequence in ALC system

time observations can be represented by other events determining future states (events) of a system.

## 3.1 Defining the Pattern

The intervention pattern is explained through an example Event-B model. This model can be reused in order to add different time considerations. As shown in the below, the intervention pattern has six variables:

```
machine m0
variables
redFlag  //denotes a system enters a hazardous event
time  //indicates any time of a system
requestTime  //time when automated system issues a request to intervene
alarmFlag  //sounding an alarm
alarmTime  //time waiting for a response before the alarm is sounding
reactionTime  //time when a human operator may react
invariants
@inv1: time ∈ ℕ1          @inv2: requestTime ∈ ℕ   @inv3: alarmTime ∈ ℕ
@inv4: redFlag ∈ BOOL    @inv5: alarm ∈ BOOL      @inv6: reactionTime ∈ ℕ
....
```

– time: This represents the current time of a system. The incrementation of this value implies the time progression.

- requestTime: This indicates any time in the future when a system may issue a request to intervene.
- reactionTime: This indicates any time in the future when a driver may respond to a request to intervene.
- alarmTime: This denotes a future time when a driver does not react to a request to intervene, and the auditory notification is immediately sounded.
- redFlag: This is a boolean flag that indicates a system issuing a request to intervene.
- alarmFlag: This is also a boolean flag that explains the status of the sound alert.

The three categories focus on various aspects of timing. The first category involves establishing an intervention timer within hazardous events that require intervention. An example can be found in the request event, which indicates the entrance of a hazardous event when a system waits for a response before an alarm is raised. This event is triggered when the machine prompts a request for intervention. Consequently, the intervention timer is configured within this event as follows:

```
event request
any
/* Maximum time of a system waiting for a response before raises an alert*/
duration
when
/*Any time is given for waiting for a human's response*/
@grd1: duration ∈ ℕ₁
/*No intervention request and alarm is OFF*/
@grd2: redFlag = FALSE ∧ alarmFlag = FALSE
then
/*Specify a time of waiting  for a driver before the alarm sounds*/
@act1: alarmTime := time + duration
/*Update the time of issuing a request to intervene*/
@act2: requestTime := time
/*Update a flag of issuing a request to intervene*/
@act3: redFlag := TRUE
/*No reaction from human yet*/
@act4: reactionTime := 0
end
```

After the creation of the intervention timer in the request event, the timeline for the intervention pattern is as outlined in Figure 3. It includes the first requirement (R1), which must be considered to allow the human operator to respond when the automation issues a request to intervene. The initialisation of time for a request to intervene (requestTime) is defined according to the current time of a system (time). In order to give a driver a chance to respond, the parameter duration indicates the waiting time of a system before the alarm is sounded. Therefore, the alert time (alarmTime) can be defined as the end of the waiting time.

**First Requirement (R1)**: When an intervention request is issued, the automated machine should give the human operator a limited time to react (duration).

The second category of the intervention timer is time progression, as shown in Figure 4. In this modelling technique, the current time can be changed with an observation of the tick event as follows:

```
event tick where
/*Work only if a system issued a request to intervene*/
@flag_intervene: redFlag = TRUE
/*System time doesn't reach an alert time yet*/
@no_alarm: redFlag = TRUE ∧ alarmFlag = FALSE ⇒ time ≤ alarmTime
/*System time arrives on alert time, so the alarm must be operating*/
@alarmOn: (time = alarmTime ∧ redFlag = TRUE) ⇒ alarmFlag = TRUE
```

```
then
/* Increment timer */
@act1 : time := time + 1
end
```

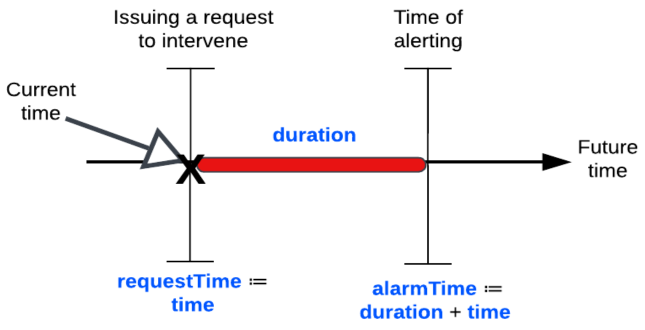

**Fig. 3.** Creation of intervention timer

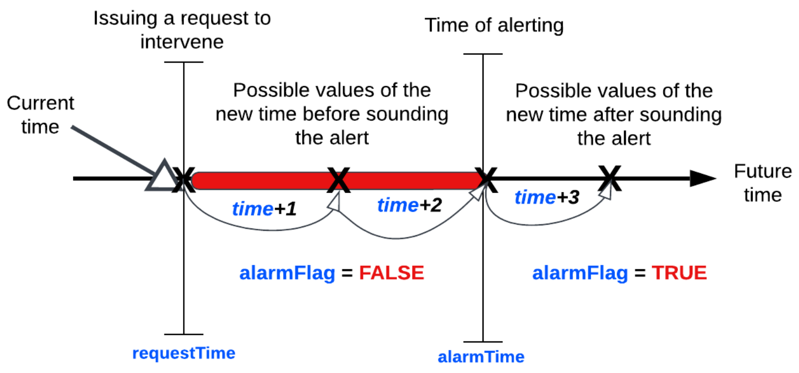
Issuing a request to intervene
Time of alerting
Current time
Possible values of the new time before sounding the alert
Possible values of the new time after sounding the alert
Future time
time+1
time+2
time+3
alarmFlag = FALSE
alarmFlag = TRUE
requestTime
alarmTime

**Fig. 4.** Time progression in intervention timer

The tick schedules the time progression associated with the alarm property. For instance, the guard alarmOn captures a critical specification when the current time is already at the alert time; therefore, the auditory notification must be sounded before computing the new value of time. To model the alarm property, the second requirement (R2) is introduced, signifying that the autonomous system will send an auditory notification if the human response is still pending.

**Second Requirement (R2)**: If the human operator fails to react within the duration time, the automated machine should immediately trigger an alarm (auditory notification)

This aspect is modelled in the notify event as follows:

```
event notify where
/*System issues a request to intervene, while an alarm is not sounding */
@grd1: alarmFlag = FALSE ∧ redFlag = TRUE
/*System time equal to or has moved beyond alert time*/
@timeAlarm: time ≥ alarmTime
then
/*Update value of alarm*/
@act1: alarmFlag := TRUE
end
```

The third category of the intervention timer models human interventions that occur either before or after the auditory notification sounds. To capture these two potential forms of human reaction, the intervene event is outlined with various time intervals. The first variant of this event, addressing human reactions prior to the auditory notification, is modelled as follows:

```
event intervene when
@grd1: redFlag = TRUE
/*Possible values of system time when a human may react*/
@grd2: time < alarmTime
then
/*Update a driver reaction time*/
@receivedReaction: reactionTime := time
@updateflag: redFlag := FALSE
end
```

The guards within the first variant of the intervene event play a crucial role in encapsulating the third requirement (R3), which highlights the potential for a human response prior to the activation of the auditory alarm. The grd2 guard in this initial form of the intervene event sets a confined timeframe, allowing for human reaction before the system's current time aligns with the alert time. Under these specified conditions, Figure 5 demonstrates the narrow window of opportunity available for a human to respond before triggering the auditory notification.

**Third Requirement (R3)**: A human operator might respond before sounding an auditory notification.

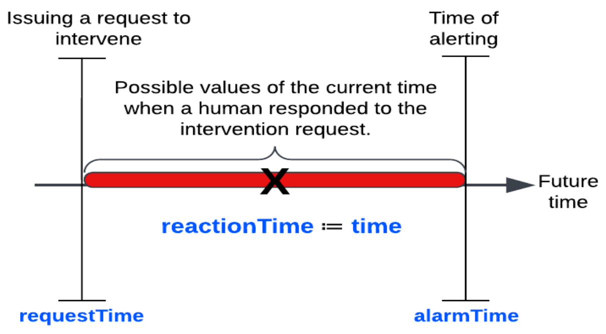
Issuing a request to intervene
Time of alerting
Possible values of the current time when a human responded to the intervention request.
Future time
reactionTime := time
requestTime
alarmTime

**Fig. 5.** Human's response window time before alert notification

Similarly, the second form of the intervene event outlines several conditions that allow a human to react after the auditory notification has sounded, which is modelled as follows:

```
event intervene when
@grd1: redFlag = TRUE
/*System has already raised an alarm*/
@grd2: time ≥ alarmTime
then
/*Update a driver reaction time*/
@receivedReaction: reactionTime := time
/*Update values of flag*/
@updateflag: redFlag := FALSE
end
```

The adjustment in the grd2 guard contributes to capturing the fourth requirement (R4) that indicates the possibility of receiving a human response after the alarm sounds. Specifically, it implies that the system's current time has already surpassed the alert time. Given these conditions, Figure 6 illustrates the possible window of time in which a human may react after the auditory notification is triggered.

**Fourth Requirement (R4)**: A human operator might respond after sounding an auditory notification.

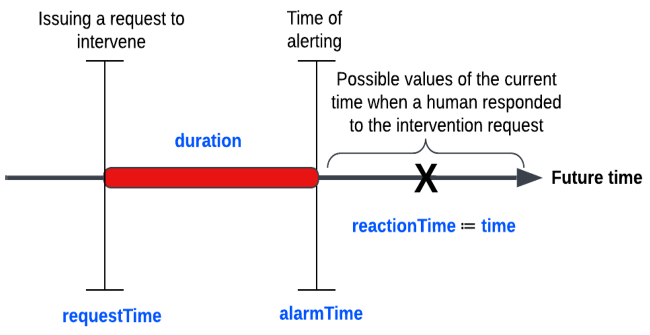
Issuing a request to intervene
Time of alerting
Possible values of the current time when a human responded to the intervention request
duration
X
Future time
reactionTime ≔ time
requestTime
alarmTime

**Fig. 6.** Human's response window time after alert notification

Since the intervention timer is executed only when a request to intervene is issued, the invariants between the automated machine and a human are simple, and we have only to satisfy the following three invariants.

@alarm_state: $\text{alarmFlag} = \text{TRUE} \Rightarrow \text{redFlag} = \text{TRUE}$

@waiting_response: $\text{redFlag} = \text{TRUE} \wedge \text{alarmFlag} = \text{FALSE} \Rightarrow \text{requestTime} \leq \text{time} \wedge \text{time} \leq \text{alarmTime}$

@alerting: $\text{alarmFlag} = \text{TRUE} \wedge \text{redFlag} = \text{TRUE} \Rightarrow \text{time} \geq \text{alarmTime}$

The invariant alarm_state indicates that the alarm signal can only be sent if there is still a need for human intervention. Additionally, the invariant waiting_response underscores that a system allocates a specific duration for the human to respond if an intervention request is dispatched (i.e., redFlag = TRUE). Specifically, the current time of the system (time) can go beyond the moment of issuing the intervention request (requestTime) up to the alert time (alarmTime). This duration is thus represented as requestTime ≤ duration ≤ alarmTime, where the system time equals the time of issuing the intervention request. On the other hand, the invariant alerting denotes that an auditory notification is only activated (i.e., alarmFlag = TRUE) if the system's current time exceeds this defined duration.

In this modelling strategy, three assumptions (As) are incorporated into the formal model:

- **A1:** A human operator might respond immediately or after the auditory notification is activated.
- **A2:** The waiting time, or duration, is not strictly defined, for example 3 seconds. Instead, it is treated as a parameter representing any positive number.
- **A3:** The automated machine is assumed to be in a safe state during the entire process of alerting and receiving responses from the human operator.

## 4 Application to ALC Case Study

This section describes the application of the intervention timing pattern to the ALC case study. Considering that this pattern primarily addresses moments when the system initiates intervention prompts, we have partitioned our application into two distinct Event-B machines[3]. The first machine covers the driving scenarios in which the ALC system is likely to issue intervention requests during the actuation task. Following that, the second machine employs Event-B refinement to show the implementation of the intervention timing pattern in response to a potential intervention request arising from the abstract model. The majority of proof obligations in this application are verified either automatically using Rodin provers or with the assistance of additional external prover plugins, such as Satisfiability Modulo Theories (SMT) for theory solving. Detailed discussions of the Event-B model are presented in their subsequent sections.

### 4.1 Abstract Machine: (*ALC actuation task*)

The abstract machine presents a scenario where intervention might be needed in the operation of the ALC system. A problematic situation can occur if the ALC system autonomously modifies the steering angle and vehicle speed, leading to the SDV drifting from the target lane. This potential issue is elaborated upon and visually represented in Figure 1.

In the static part of the model, the potential positions of an SDV are represented by the abstract set POSITION, which denotes the various positions to which an SDV may travel. Since a lane could be considered part of these positions, we define the constant Lane as a subset of set POSITION in the following manner:

$$\text{@typeof-Lane}: \mathsf{Lane} \subseteq \mathsf{POSITION}$$

To model the movement of an SDV, we introduce an abstract constant MOVE, which activates a specific speed and steering angle to achieve a new position, enabling the SDV to transition across multiple locations.

$$\text{@typeof-move}: \mathsf{MOVE} \in \mathsf{POSITION} \times \mathsf{SPEED} \times \mathsf{STEERING\_ANGLE} \rightarrow \mathbb{P}_1(\mathsf{POSITION})$$

Note that for a position pos, a speed spd, and a steering angle agl, $\mathsf{MOVE}(\mathsf{pos} \mapsto \mathsf{spd} \mapsto \mathsf{agl})$ gives the set of positions that the vehicle might move into.

[3] An Event-B model is publicly available as a Rodin archive at: https://doi.org/10.5281/zenodo.10944865

In the dynamic part of the model, the physical position of the SDV is modelled as a variable ALC_POSITION, accompanied by a safety invariant ALC_POSITION ∈ Lane, signifying that the SDV must always reside within the Lane. Furthermore, the initial position of the SDV is represented as a constant init_position, which stipulates that the SDV's starting position must be within the Lane, denoted as init_position ∈ Lane. The modelling of the actuation task for an ALC system involves five events. The first event, ALC_actuation, abstractly captures the determination of the steering and speed settings for an SDV as follows:

```
event ALC_actuation any speed steering
where
/∗Definition of steering and speed∗/
@grd1: speed ∈ SPEED
@grd2: steering ∈ STEERING_ANGLE
/∗No intervention request∗/
@grd3: redFlag = FALSE
then
/∗Specify steering and speed for ALC system∗/
@act1: ALC_SPEED := speed
@act2: ALC_STEERING := steering
end
```

The second event, accept_move, presumes that the SDV can transition to a new position as described below:

```
event accept_move where
/∗A new position leads to a position inside Lane∗/
@grd1: MOVE(ALC_POSITION ↦ ALC_SPEED ↦ ALC_STEERING ) ⊆ Lane
/∗No intervention request∗/
@grd2: redFlag = FALSE
then
/∗Moved into a new position inside the target lane ∗/
@act1: ALC_POSITION :∈ MOVE(ALC_POSITION ↦ ALC_SPEED ↦ ALC_STEERING)
end
```

The third event, required_intervention, initiates an intervention request when the speed and steering are set to move the SDV to a position outside the lane, as detailed below:

```
event required_intervention where
/∗A new position leads to a position outside Lane∗/
@grd1: MOVE(ALC_POSITION ↦ ALC_SPEED ↦ ALC_STEERING ) ⊈ Lane
@grd2: redFlag = FALSE
then
/∗Initiates an intervention request∗/
@act1: redFlag := TRUE
end
```

The fourth event, mitigate_move, presumes that the ALC system is capable of implementing a mitigation strategy when the proposed speed and steering risk moving the SDV out of the lane. This is achieved by instituting an adjusted or mitigated steering and speed as detailed below:

```
event mitigate_move any mitigated_sp mitigated_steer
where
@grd1: mitigated_sp ∈ SPEED ∧ mitigated_steer ∈ STEERING_ANGLE
/∗Intervention request was sent∗/
@grd2: redFlag = TRUE
/∗Specification on the ALC mitigation mode∗/
@grd3: MOVE(ALC_POSITION ↦ mitigated_sp ↦ mitigated_steer) ⊆ Lane
then
/∗Moved into a new position inside the lane∗/
@act1: ALC_POSITION :∈ MOVE(ALC_POSITION ↦ mitigated_sp ↦ mitigated_steer)
end
```

The final event, human_intervene, simulates the human response to the intervention request as follows:

```
event human_intervene any manual_sp manual_steer
where
@grd1: manual_sp ∈ SPEED ∧ manual_steer ∈ STEERING_ANGLE
@grd2: redFlag = TRUE
/*Assumption that the human responses correctly*/
@grd3: MOVE(ALC_POSITION ↦ manual_sp ↦ manual_steer) ⊆ Lane
then
/*Moved into a new position inside the target lane */
@act1: ALC_POSITION :∈ MOVE(ALC_POSITION ↦ manual_sp ↦ manual_steer)
@act2: redFlag := FALSE
end
```

## 4.2 Refined Machine: (*Intervention Timing Pattern*)

The refined model is specifically developed to incorporate the intervention timing pattern within its structure. This model utilises the concept of machine inclusion in Event-B to extend and build upon the foundational concepts presented in Section 3.1. The main benefit of this approach lies in its ability to instantiate the intervention timing pattern on two separate occasions within the model:

1. The first instantiation models the time-sensitive aspects leading up to the point at which an auditory notification is issued to the driver.

   ```
   machine m1 refines m0 sees c0
   /*First instantiation is included as beforeAlarm*/
   includes interventionTimingPattern.m0 as beforeAlarm
   ```

2. The second instantiation of the intervention timing pattern is activated following the auditory notification. At this stage, the SDV engages its mitigation mode, which is a response mechanism to correct or manage the situation that requires the intervention.

   ```
   machine m1 refines m0 sees c0
   /*Second instantiation is included as afterAlarm*/
   includes interventionTimingPattern.m0 as afterAlarm
   ```

These applications of the intervention timing pattern will be summarised in the following steps.

*1) Pre-Notification Temporal Modelling and Establishing Intervention Timing Before Auditory Alerts:* The implementation of the intervention timing pattern in the refined machine incorporates intervention events. The temporal elements involved are synchronised with the request event, which is integrated into the abstract required_intervention event. Consequently, the model initialises the timing aspects at the moment when the ALC system requires intervention.

```
/*Timing aspects modelled in the request event are included in the required intervention event*/
event required_intervention extends required_intervention synchronises beforeAlarm.request end
```

The tick event, which is associated with the initiation of an intervention request, is incorporated as follows:

```
/*The time progression associated when ALC issues a request to intervene*/
event tick synchronises beforeAlarm.tick end
```

When the time progression reaches the alert moment and the driver has not reacted, an auditory notification is issued. This is facilitated by the synchronised notify event within the alarm event, detailed as follows:

```
/*Auditory notification is issued if driver does not react*/
event alarm synchronises beforeAlarm.notify end
```

However, the driver may react during the time progression before the auditory notification is sent. This response is captured by the synchronised intervene event, which aligns with the abstract human_intervene event, as outlined below:

```
/*Driver may intervene before auditory notification*/
event human_intervene extends human_intervene synchronises beforeAlarm.intervene end
```

*2) Post-Notification Mitigation Strategy and ALC Response Mechanisms Following Auditory Alerts:* The implementation of the intervention timing pattern also addresses the timing properties subsequent to the activation of the auditory notification. Thus, the synchronised request event is integrated within the event that presumes the auditory notification has been issued, as described below.

```
/*Reinitialise the intervention timer after alarm sounded*/
event alarm synchronises afterAlarm.request end
```

The tick event is also synchronised with the progression of time that corresponds to the moment when the alarm sounds, as outlined below:

```
/*Schedule time progression with the alarm is sounded*/
event tick synchronises afterAlarm.tick end
```

To capture the essential timing property that allows a driver to respond after the auditory notification has been issued, the abstract human_intervene event is extended to the human_react_after_alarm event. This expansion includes the synchronised intervene event, as illustrated below:

```
/*Driver may intervene after auditory notification*/
event human_react_after_alarm extends human_intervene synchronises afterAlarm.intervene end
```

However, if a driver fails to respond, the mitigation strategy of the ALC system may be initiated. In this instance, the synchronised notify event is incorporated as follows:

```
/*Mitigated movement applied if a driver fails to respond*/
event mitigate_move extends mitigate_move synchronises afterAlarm.notify end
```

## 5 Discussion

The intervention timing pattern is a critical component in the development of SDV systems, especially when considering the human driver as a fallback option during hazardous driving events. It outlines a structured method for specifying the timing properties needed to model the windows of opportunity for a driver's intervention response. This pattern finds applicability in autonomous or semi-autonomous systems where a human operator has an oversight role and is expected to take more active control in some hazardous situations, such as veering outside lanes. In these systems, a failsafe mode is available, for instance, to decelerate, in case the operator does not respond in a timely manner.

The analysis employs a two-step instantiation process to precisely chart the timing of a driver's potential responses to the ALC system's prompts. This modelling strategy differentiates between scenarios in which a driver might respond before the urgency of auditory notification is perceived, and situations where the driver's reaction occurs after acknowledging the notification. This approach facilitates a thorough examination of the range of a driver's possible reaction times in relation to the auditory signals from the ALC system.

The benefits of utilising the intervention timing pattern in SDV systems are multifaceted:

- **Requirement and Assumption Identification:** The intervention timing pattern serves as a structured framework that identifies specific requirements and assumptions related to the driver's engagement in the SDV's autonomous operations. By explicitly defining these parameters, it assists in shaping a comprehensive understanding of the role and expectations of human intervention, ensuring that both system developers and stakeholders have a clear blueprint to refer to.
- **Addressing Modelling Challenges:** One of the complex aspects of modelling SDV systems lies in accounting for the uncertain nature of a driver's response during autonomous operations. The intervention timing pattern plays a crucial role in overcoming this issue. It establishes a methodical framework that integrates the possibility of human responses into the system's functional procedures, ensuring that these variables are included in the system design even if they are not directly detectable. Therefore, it ensures that the model remains robust and reflective of real-world scenarios where driver reactions may not always be apparent.
- **Driver Inclusion Beyond Fallback:** Traditionally, the role of a driver in an SDV system is often relegated to that of a mere fallback option—intervening only when the system fails or is unsure of the next course of action. However, the intervention timing pattern encourages developers to transcend this limited view. It prompts the consideration of the driver as an active participant, capable of varying responses across different scenarios. This, in turn, aids in crafting a more robust and realistic model of fallback mechanisms within the SDV system.

In summary, the intervention timing pattern not only accentuates the intricacies of human intervention in automated systems but also propels a forward-thinking approach to SDV system development. This pattern is instrumental in acknowledging the range of possible human reactions, guiding the creation of systems that excel technically while also being attuned to human behaviour and needs. Such an approach promotes a seamless integration of humans and machines, aiming for a balanced and cooperative relationship.

## 6 Related Work

Cansell et al. [6] developed a pattern to model the timing and order of events within systems, using time-stamped actions and reactions to simulate real-time processes. Their model uses a clock (timer) to track current time and events set to trigger at future times. The system advances time and activates events accordingly. However, this model does not handle interruptions during event sequences.

Butler and Falampin [5] proposed a refinement strategy of timing properties that introduces a clock variable representing the current time and an operation that progresses the clock. Therefore, time constraints are added to the clock to handle interruptions during event sequences where the clock cannot move beyond the specific point at which the deadline is violated.

Based on this methodology, many studies, such as [18,21], have been carried out to extend Event-B with timing properties. Sarshogh and Butler [18] propose a trigger response pattern to develop Event-B models with several timing properties such as deadline, delay and expiry. Their approach assigns timestamps for trigger and response events and employs a tick event to prevent the global clock from moving to a point where time constraints between the trigger and response events would be violated. Zhu et al. [21] extended the work of [5,18] to provide formally the semantics and syntax between the trigger and response events.

In autonomous systems, our pattern targets how humans react when automated machines trigger intervention. We specify an intervention timeline based on the human reaction time and the alert time. The deadline can be seen as the time when a driver may react, while the alert time combines the delay and expiry based on the received human's response. Therefore, we specify the criteria for triggering intervention based on these timelines. Nonetheless, the time-sensitive characteristics associated with the autonomous functions continue to be a significant issue. For instance, the timing aspects related to observing the driving environment and determining the target path are not fully investigated.

## 7 Conclusion and Future Work

This paper introduced the intervention timing pattern for managing the timing of interventions in SDV systems, conceptualised within the Event-B framework. The focus was on defining the time-sensitive parameters that handle when and how human drivers should take over control in critical situations. The intervention timing pattern specifies the essential temporal constraints when human drivers must intervene. Various driver responses were identified and their implications for system functionality were analysed. The actuation task of the ALC system served as a case study, validating the adaptability of the timing pattern to accommodate different driver behaviours. Future research is set to explore additional temporal dimensions in SDV systems, especially those timing properties critical to autonomous operations such as environmental perception and the determination of driving decisions.

## References

1. Abrial, J.R.: Modeling in Event-B: system and software engineering. Cambridge University Press (2010)
2. Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. International journal on software tools for technology transfer **12**(6), 447–466 (2010)
3. Becker, C., Yount, L., Rozen-Levy, S., Brewer, J., et al.: Functional safety assessment of an automated lane centering system. Tech. rep., United States. Department of Transportation. National Highway Traffic Safety ... (2018)
4. Blanco, M., Atwood, J., Vasquez, H.M., Trimble, T.E., Fitchett, V.L., Radlbeck, J., Fitch, G.M., Russell, S.M., Green, C.A., Cullinane, B., et al.: Human factors evaluation of level 2 and level 3 automated driving concepts. Tech. rep., National Highway Traffic Safety Admin. (2014)

5. Butler, M., Falampin, J.: An approach to modelling and refining timing properties in B. In: Refinement of Critical Systems (RCS) (2002)
6. Cansell, D., Méry, D., Rehm, J.: Time constraint patterns for Event-B development. In: B 2007: Formal Specification and Development in B: 7th International Conference of B Users, Besançon, France, January 17-19, 2007. Proceedings 7. pp. 140–154. Springer (2006)
7. Christensen, A., Cunningham, A., Engelman, J., Green, C., Kawashima, C., Kiger, S., Prokhorov, D., Tellis, L., Wendling, B., Barickman, F.: Key considerations in the development of driving automation systems. In: 24th enhanced safety vehicles conference. Gothenburg, Sweden (2015)
8. Claybrook, J., Kildare, S.: Autonomous vehicles: No driver... no regulation? Science **361**(6397), 36–37 (2018)
9. Comma.ai: Openpilot: an open source driver assistance system (June 2022), https://github.com/commaai/openpilot/
10. Fridman, L.: Human-centered autonomous vehicle systems: Principles of effective shared autonomy. arXiv preprint arXiv:1810.01835 (2018)
11. Hoang, T.S., Dghaym, D., Snook, C., Butler, M.: A composition mechanism for refinement-based methods. In: 2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS). pp. 100–109. IEEE (2017)
12. HSRC: Top Safety Picks by HSRC (Highway safety research & communications). https://www.iihs.org/iihs/ratings/TSP-List (last accessed 2022/06/22)
13. Lerner, N., Jenness, J., Robinson, E., Brown, T., Baldwin, C., Llaneras, R.E., et al.: Crash warning interface metrics. Tech. rep., United States. National Highway Traffic Safety Administration (2011)
14. Leuschel, M., Butler, M.: ProB: an automated analysis toolset for the B Method. International Journal on Software Tools for Technology Transfer **10**(2), 185–203 (2008)
15. Merat, N., Jamson, A.H., Lai, F.C., Daly, M., Carsten, O.M.: Transition to manual: Driver behaviour when resuming control from a highly automated vehicle. Transportation research part F: traffic psychology and behaviour **27**, 274–282 (2014)
16. Radovic, M.: Tech talk: Untangling the 5 levels of drone autonomy. https://droneii.com/project/drone-autonomy-levels (last accessed 2022/11/25)
17. SAE: Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. SAE Standard J **3016**, 1–16 (2014)
18. Sarshogh, M.R., Butler, M.: Specification and refinement of discrete timing properties in Event-B. Electronic Communications of the EASST **36** (2011)
19. Vagia, M., Rødseth, Ø.J.: A taxonomy for autonomous vehicles for different transportation modes. In: Journal of Physics: Conference Series. IOP Publishing (2019)
20. Xu, W.: From automation to autonomy and autonomous vehicles: Challenges and opportunities for human-computer interaction. Interactions **28**(1), 48–53 (2020)
21. Zhu, C., Butler, M., Cirstea, C.: Formalizing hierarchical scheduling for refinement of real-time systems. Science of Computer Programming **189**, 102390 (2020)