UNIVERSITÄT ZUM BEISPIEL
INSTITUT FÜR BEISPIELE

Monitoring der AUTOSAR Timing Extensions mittels TeSSLa

Monitoring of the AUTOSAR Timing Extensions with TeSSLa

Bachelorarbeit

im Rahmen des Studiengangs
Informatik
der Universität zu Lübeck

vorgelegt von
Hendrik Streichhahn

ausgegeben und betreut von
Prof. Dr. Martin Leucker

mit Unterstützung von
Martin Sachenbacher und
Daniel Thoma

Lübeck, den 1.1. 1970

Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

(Hendrik Streichhahn)
Lübeck, den 1.1. 1970

Kurzfassung Abstract Deutsch

Abstract Kurzfassung Englisch.

Contents

Liste der Todos

1. Introduction

Timing behavior is one of the most important properties of computer systems. Especially in safety-critical applications, a wrong timed reaction of the system can have disastrous consequences, for example an intervention of pacemaker, that occurred too early or too late would risk the life of the patient. In Cyber-physical Systems, e.g. the Electronic Stability Control of a vehicle, wrong timing can also lead to property damage, injuries or deaths. Also environmental aspects are affected by timing cyber-physical systems, for example combustion engines need exact timing to produce as little emissions as possible.
The interconnection of several components in cyber-physical systems makes the design and analysis of the timing behavior of these systems significant hard, because not only the components on its own, but also the complete system must be considered. In this context, testing is a major problem, because it is hard to reproduce the exact state of the system, in which the error occurred. In many cases, the error does not lay in the component where it became visible, it was carried off to other parts, which results in a malfunctioning system, where it is extremely hard to find the bug that caused the problem. Online Monitoring is the key technique to address this problem, because you can isolate the error, without the need of storing and recreating the state of the system, when searching the error.
The goal of this thesis is to create a monitoring tool for the *AUTOSAR* (**AUT**omotive **O**pen **S**ystem **AR**chitecture) Timing Extensions, which were created to increase the interoperability and exchangeability of car components.

1.1. AUTOSAR TIMEX

AUTOSAR is a development partnership in the automotive industry. As stated before, the main goal is to define a standardized interface, to increase interoperability, exchangebility and re-usability of parts and therefore simplify development and production. Three different layers are defined in the specification. *Basic Software* is an abstraction layer from components, like network or diagnostic protocols, or operating systems. *AUTOSAR-Software* defines, how application has to be build. For Basic Software and AUTOSAR Software, there are definitions for standardized Interfaces, to enable the communication via the *Autosar Runtime Environment*. It works as middleware, in which the *virtual function bus* is defined [AUT17]. The

AUTOSAR Timing Extension are describing timing constraints for actions and reactions of components, that are communicating via the Virtual Function Bus, for example the latency timing constraint, that describes the amount of time between two subsequent events, or the event triggering constraints, that are used to describe the timing behavior of events, that are created in one component without getting input from another component.
Problematic with the AUTOSAR Timing Extensions is, that the definitions are not very formal and have room left for interpretation. Let's take a look at the *BurstPatternEventTriggering*. The BurstPatternEventTriggerings describe events clusters, with events that occur with short time distances, with large time distances between the clusters. The following attributes are needed:

Attribute	**Type**	**Explanation**
$maxNumberOfOccurrences$	PositiveInteger	maximum number of events per burst
$minNumberOfOccurrences$	PositiveInteger	(optional) minimum number of events per burst
$minimumInterArrivalTime$	TimeValue	minimum time distance between any two events
$patternLength$	TimeValue	length of each burst
$patternPeriod$	TimeValue	(optional) Time distance between the start points of two subsequent bursts
$patternJitter$	TimeValue	(optional) maximum of allowed deviation from periodic pattern

As example, we set:

- $maxNumberOfOccurrences = 3$
- $minNumberOfOccurrences = 1$
- $minimumInterArrivalTime = 1$
- $patternLength = 3$
- $patternPeriod = 3.5$
- $patternJitter = 1.5$

The combination of $patternPeriod$ and $patternJitter$ can be interpreted in an accumulating as seen in 1.1 or non-accumulating way as seen in 1.2 way. In the accumulating interpretation, the reference for the periodic occurrences is only the start point of the previous

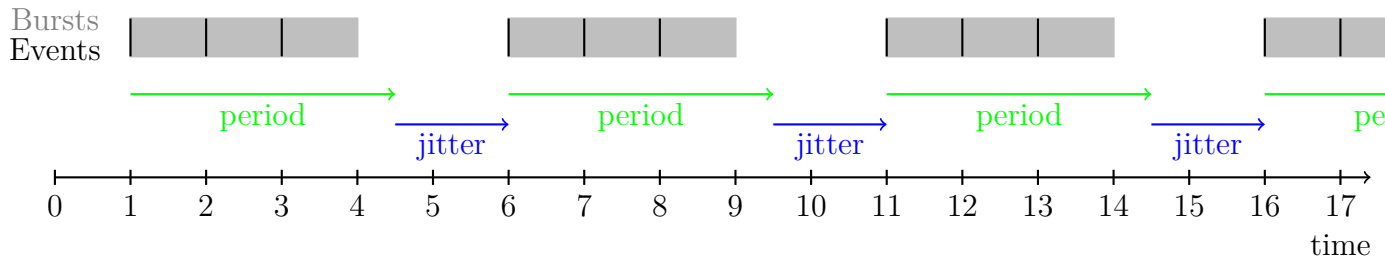

Figure 1.1.: BurstPatternEventTriggering Period-Jitter **accumulating**

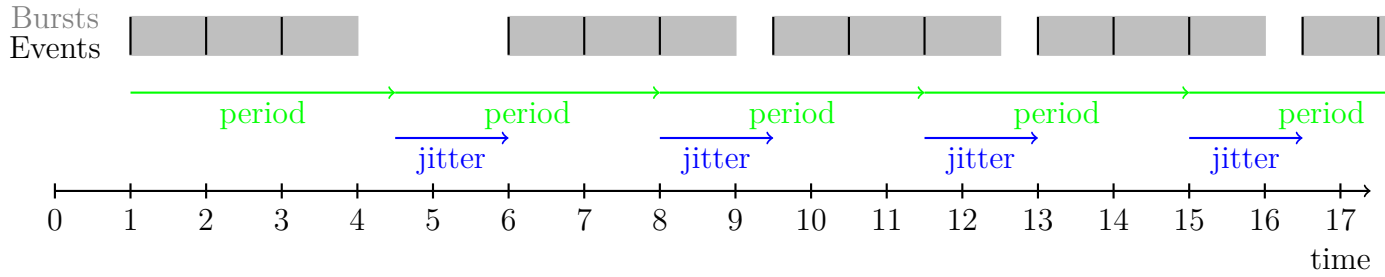

Figure 1.2.: BurstPatternEventTriggering Period-Jitter **non-accumulating**

With the definition of *patternLength* („time distance between the beginnings of subsequent repetitions of the given burst pattern“) you would think, that the accumulating variant is meant. Against that, the period attribute in *PeriodicEventTriggering*-Constraint is also defined as „distance between subsequent occurrences of the event“in the text, hence it is understandable the accumulating way, but there is also the formal definition

$$\exists t_{reference} \forall t_n : t_{reference} + (n+1) * period \leq t_n \leq t_{reference} + (n-1) * period + jitter,$$

where t_n is the time of the n-th Event and $t_{reference}$ is a reference point, from which the periodic pattern starts, so the *PeriodicEventTriggering*-Constraint is meant to be understood in the non-accumulating way. It remains unclear, in which way the *BurstPatternEventTriggering* is meant to be understood.

Another problem of the AUTOSAR Timing Extensions is, that they were made for design purposes, monitoring them can be difficult, as they may need continuously growing time and memory resources, which makes online monitoring impossible (more on monitorability in 3). As example, we will use the burst pattern again, this time using the attributes as

- $maxNumberOfOccurrences = INT\_MAX$
- $minNumberOfOccurrences = 1$

- $minimumInterArrivalTime = 0$
- $patternLength = 3$
- $patternPeriod$ unused
- $patternJitter$ unused

In 1.3 you see the application of the BurstPatternEventTriggering Constraint with the given parameters on a stream with events at the timestamps 3, 3.5, 4, 4.5. You can see the development of possible the burst cluster with ongoing time. The gray lines show, where the burst can lay, the black lines show, where they definitely are. In timestamp 3 with only one event so far, only one burst has to be considered and it can lay between timestamp 0 and 6, the only limitation is, that it must include timestamp 3 with the event in that point. In Timestamp 3.5, there are two events (at 3 and 3.5) so far and there are two possibilities for burst placements. The first possibility with only one burst with both events in it, and the second possibility, where the events are in different bursts. The third graphic shows the trace in timestamp 4 with three different events so far (3, 3.5, 4) and there are three different possibilities for burst placements to consider. One possible burst contains all three events, the second possibility has one burst with the event at timestamp 3 and one burst with the events at 3.5 and 4 and the third possibility has one Burst with the events at 3 and 3.5 and one burst with the event at 4. The possible bursts in graphic 4 are analog to the third graphic, one possibility with one burst containing all 4 events and 3 possibilities with the first burst containing the first event, the first and second event or the first, the second and the third event and the second burst containing the remaining events.
In this Example, we see, that it is possible to create an unlimited number of possibilities for burst placements within one burst length, when the *minimumInterArrivalTime*-attribute is 0, which results in an infeasible resource consumption, as unlimited memory and time is needed to check the constraint in following events. Therefore, online monitoring this constraint is impossible in general, because of the strict resource limitations. More on that in 3.

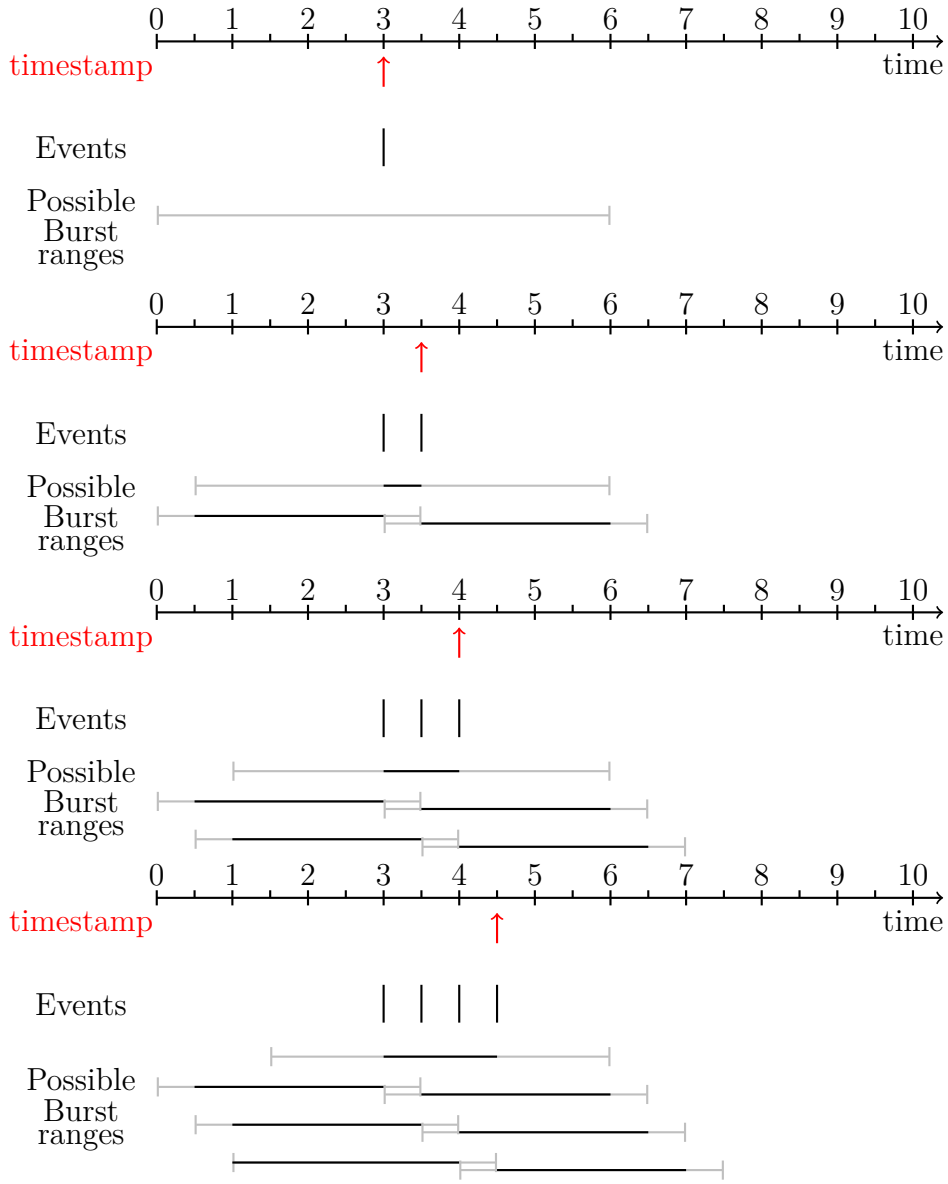

Figure 1.3.: BurstPatternEventTriggering Possible bursts, ↑ shows the current time

2. Basics

2.1. TeSSLa

TeSSLa (**Te**mporal **S**tream-based **S**pecification **La**nguage) is a functional programming language, build for runtime verification of streams. In TeSSLa, streams are defined as traces of events, each event consists of one data value from a data set $\mathbb{D}$ and a time value from a discrete time domain $\mathbb{T}$. This time domain needs a total order and subsequent timestamps must have increasing time values. A TeSSLa Specification can have several streams with different data sets, but each of these streams must use the same time domain $\mathbb{T}$, which timestamps are increasing over all streams. Each stream can have only one event per timestamp, but it is possible to have events on different streams at the same timestamp.
A distinction between synchronous and asynchronous streams is made. A set of synchronous streams have events in the exact same time stamps, events in asynchronous streams do not have this restriction. It is easy to see, that synchronous streams are a subset of the asynchronous ones, therefore we will only use asynchronous streams from now on.
In TeSSLa, calculations are done, when new events are arriving. Based on the specification, output streams are generated with events on the same timestamps as the used input streams, but filtering is possible, where not all input events produce output events. With the *delay*-operator, it is possible to create new timestamps. This possibility will take a large role in this thesis, more on that later.
At the timestamps, in which events arrived and calculations are done, you only have direct access to the youngest event of each stream, but with the use of the *last*-operator, which can be used recursively, the event before that can be accessed. The *lift*-operator applies a function, which is defined on data values $\mathbb{D}$, on each event of one or more streams. Similar to this, the *slift*-operator (signal lift) first applies the given function, when there was at least one event of each input stream. The *time*-operator returns the time value of an event.

3. Monitorability

4. Timmo2Use Constraints

5. Implementierungen

In der Evaluierung wird das Ergebnis dieser Arbeit bewertet. Eine praktische Evaluation eines neuen Algorithmus kann zum Beispiel durch eine Implementierung geschehen. Je nach Thema der Arbeit kann sich natürlich auch die gesamte Arbeit eher im praktischen Bereich mit einer Implementierung beschäftigen. In diesem Fall gilt es am Ende der Arbeit insbesondere die Implementierung selber zu evaluieren. Wesentliche Fragen dabei können sein:

– Was funktioniert jetzt besser als vor meiner Arbeit?
– Wie kann das praktisch eingesetzt werden?
– Was sagen potenzielle Anwender zu meiner Lösung?

5.1. Implementierungen

Wenn Implementierungen umfangreich beschrieben werden, ist darauf zu achten, den richtigen Mittelweg zwischen einer zu detaillierten und zu oberflächlichen Beschreibung zu finden. Eine Beschreibung aller Details der Implementierung ist in der Regel zu detailliert, da die primäre Zielgruppe einer Abschlussarbeit sich nicht im Detail in den geschriebenen Quelltext einarbeiten will. Die Beschreibung sollte aber durchaus alle wesentlichen Konzepte der Implementierung enthalten. Gerade bei einer Abschlussarbeit am Institut für Softwaretechnik und Programmiersprachen lohnt es sich, auf die eingesetzten Techniken und Programmiersprachen einzugehen. Ich würde in einer solchen Beschreibung auch einige unterstützende Diagramme erwarten.

6. Zusammenfassung und Ausblick

Die Zusammenfassung greift die in der Einleitung angerissenen Bereiche wieder auf und erläutert, zu welchen Ergebnissen diese Arbeit kommt. Dabei wird insbesondere auf die neuen Erkenntnisse und den Nutzen der Arbeit eingegangen.

Im anschließenden Ausblick werden mögliche nächste Schritte aufgezählt, um die Forschung an diesem Thema weiter voranzubringen. Hier darf man sich nicht scheuen, klar zu benennen, was im Rahmen dieser Arbeit nicht bearbeitet werden konnte und wo noch weitere Arbeit notwendig ist.

A. Anhang

Dieser Anhang enthält tiefergehende Informationen, die nicht zur eigentlichen Arbeit gehören.

A.1. Abschnitt des Anhangs

In den meisten Fällen wird kein Anhang benötigt, da sich selten Informationen ansammeln, die nicht zum eigentlichen Inhalt der Arbeit gehören. Vollständige Quelltextlisting haben in ausgedruckter Form keinen Wort und gehören daher weder in die Arbeit noch in den Anhang. Darüber hinaus gehören Abbildungen bzw. Diagramme, auf die im Text der Arbeit verwiesen wird, auf keinen Fall in den Anhang.

List of Figures

List of Tables

Quelltextverzeichnis

Abkürzungsverzeichnis

TDO zu erledigen *T*o *D*o

Bibliography

[AUT17] AUTOSAR: *Virtual Functional Bus, 4.3.1.* `https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_VFB.pdf`. Version: December 2017

[CHL+18] Convent, Lukas ; Hungerecker, Sebastian ; Leucker, Martin ; Scheffel, Torben ; Schmitz, Malte ; Thoma, Daniel: *TeSSLa: Temporal Stream-based Specification Language.* 2018