# Languages for Safety-Certification Related Properties

Clara Benac Earle and Elena Gómez-Martínez*, Stefano Tonetta†, Stefano Puri and Silvia Mazzini‡, Jean-Louis Gilbert and Olivier Hachet§, Ramon Serna Oliver¶, Cecilia Ekelin‖, Katiusca Zedda**
*Babel Group. Universidad Politécnica de Madrid, Spain Email: {cbenac,egomez}@babel.ls.fi.upm.es
†Fondazione Bruno Kessler, Italy Email:tonettas@fbk.eu
‡Intecs SpA, Italy Email:{stefano.puri,silvia.mazzini}@intecs.eu
§Thales Communication, France Email:{jean-louis.gilbert,olivier.hachet}@thalesgroup.com
¶TTTech Computertechnik AG, Austria Email:ramon.serna@tttech.com
‖Volvo Technology, Sweden Email:cecilia.ekelin@volvo.com
**Akhela srl, Italy Email:katiuscia.zedda@akhela.com

## I. Introduction

The *Safety Certification of Software-Intensive Systems with Reusable Components* project, in short SafeCer (www.safecer.eu), is targeting increased efficiency and reduced time-to-market by composable safety certification of safety-relevant embedded systems. The industrial domains targeted are within automotive and construction equipment, avionics, and rail. Some of the companies involved are: Volvo Technology, Thales, TTTech, and Intecs among others. SafeCer includes more than 30 partners in six different countries and has a budget of €25.7 millions.

A primary objective is to provide support for system safety arguments based on arguments and properties of system components as well as to provide support for generation of corresponding evidence in a similar compositional way. By providing support for efficient reuse of certification and stronger links between certification and development, component reuse will be facilitated, and by providing support for reuse across domains the amount of components available for reuse will increase dramatically. The resulting efficiency and reduced time to market will, together with increased quality and reduced risk, increase competitiveness and pave the way for a cross-domain market for software components qualified for certification.

## II. System and components' properties related to safety certification

Safety certification requires the production of a big amount of evidence to convince a certifying authority that a system is safe. In order to produce such evidence, the designer of the system has to enrich the system description with SP, i.e. with properties that are relevant to the demonstration of the system's safety. In order to integrate the certification with the development process, the modeling languages must support the specification of these Safety Properties (SP for short).

The research leading to these results has received funding from the ARTEMIS Joint Undertaking pSafeCer and nSafeCer, under grant agreement n$^o$ 269265 and n$^o$ 295373, respectively, and from National funding.

In a component-based software system, where systems are built by assembling existing components, it is beneficial if the SP can be associated with individual components, allowing it to be reused when the component is reused in a new system. Then, it can be desirable to provide together with the component some selected information about its internals, without fully exposing all details.

Typically, a component only has a subset of the available properties associated with it. These properties cover both functional aspects and extra-functional aspects such as timing, memory usage, error handling, etc. Informally, SPs represent an abstraction of a particular functional or extra-functional aspect of a component. Since the focus of the paper is on the area of certification, we are primarily interested in properties relevant to the activities for safety certification.

The left-hand column in Table I shows the classes of SP considered in this survey. They have been identified from the requirements provided by the industrial companies involved in the SafeCer project. Below we give a summary of each of these classes.

*1)* **Types and value ranges:** Value ranges give a crude abstraction of the component behaviour, stating that the values of a particular entity are always in the specified range.

*2)* **Functional Pre/Post conditions:** Traditionally, pre and post conditions (and invariants) are used to formally abstract the functional behaviour of a sequential program. In a component-based context, they can be used to abstract the functionality of a single invocation of a provided service or method, or a single execution of a passive component.

*3)* **Temporal contracts:** As pre/post conditions, temporal contracts structure the properties into assumptions, which are the properties that the component expects to be satisfied by the environment, and guarantees, which are the properties satisfied by the component in response.

*4)* **Valid interaction sequences:** The static definitions of component interfaces (e.g., interface signatures extended with value ranges and pre/post conditions) can be complemented by SPs that restrict the order in which the provided functionality can be accessed. In addition to defining what is considered a valid order of interaction, the SPs could also express timing

TABLE I
LANGUAGES COMPARISON WITH REGARDS TO SP

| Property | AADL | CHESS | EAST-ADL | OCRA | VERDE |
|---|---|---|---|---|---|
| Types and values ranges | Yes | Yes | Yes | Yes | Yes |
| Functional pre/post conditions | Partly | Yes | No | Yes | Yes |
| Temporal contracts | No | Partly | No | Yes | No |
| Valid interaction sequences | No | Partly | Yes | Yes | Yes |
| Memory usage | Yes | No | Yes | Partly | Yes |
| Real-time properties | Yes | Yes | Yes | Yes | Yes |
| Communication resource usage | Yes | No | Yes | Partly | Yes |
| Compliance of code with a particular standard | Partly | Partly | Partly | No | Partly |
| Failure propagation | Yes | Yes | Yes | Yes | Yes |
| Behavioural model | Yes | Yes | No | Yes | Yes |
| Safety integrity level | Yes | No | Partly | No | Partly |
| Fault trees and FMEA tables | No | No | No | No | No |
| Traces and sequences of subcomponent interaction | Partly | No | Partly | No | Partly |

constraints over the sequences.

*5)* **Memory usage:** Information about the memory requirements of a component, for example in terms of program memory and static data memory, is needed to find a suitable allocation of functionality to computational nodes. Moreover, knowledge of the dynamic memory usage of components is needed to argue about the absence of interference caused by memory overflow.

*6)* **Real-time properties:** These are properties related to the execution time of the system or the components or of some operation. In hard real-time systems, where all deadlines must be met, the crucial timing property is the Worst Case Execution Time (WCET). In systems or subsystems with soft or no deadlines, the focus is typically the average execution time or the execution time distribution, allowing for analysis of performance.

*7)* **Communication resource usage:** Similarly to other resources, a component can specify its usage of communication resources such as a communication bus. This can either be specified in a simple form, e.g. bandwidth, or by a complex dynamic model defining how resource usage varies as a result of calls to the component.

*8)* **Compliance of code with a particular standard:** This property represents a guarantee that the component complies with a given standard.

*9)* **Failure propagation:** This SP includes: i) Error propagation: it specifies how errors propagate from the input to the output of components; ii) Fault tolerance: it specifies how the system or component continues the operation also in the presence of faults; and iii) FDIR: it specifies the component's ability to detect, identify and recover from faults.

*10)* **Behavioural model:** This property type covers a variety of complex behavioural specifications, addressing functional aspects, non-functional concerns like resource usage or fault tolerance, or a combination of the two.

*11)* **Safety Integrity level:** A property denoting the safety integrity level (SIL) or Automotive Safety Integrity Level (ASIL) associated with the entity.

*12)* **Fault trees, FMEA tables:** A fault tree analysis (FTA) is a representation of the logical relationships linking basic causes (faults or failure events) to an undesirable high-level event or failure. FMEA (Failure Modes and Effects Analysis) is a technique to systematically analyse the severity and probability of the different failures.

*13)* **Traces and sequences of subcomponent interaction:** Traces and interaction sequences represent possible executions of the sub-components of a composite component. They can be produced by the analysis or specified by the user as scenario.

Note that FTA, FMEA tables, and traces are typically generated from the models and not specified as properties. This is the reason why in Table I it is set as not supported even if there are tools for the generation of these SP.

## III. LANGUAGES CONSIDERED FOR PROPERTIES RELATED TO SAFETY CERTIFICATION

From the various existing languages for specifying SP, we have been chosen temporal logics such as LTL and Othello, and modelling languages such as CHESS, VERDE, AADL and EAST-ADL. Our choice is based on the following reasons: (i) they can express several of the SP of interest (see previous section), (ii) there exist some verification and validation methods and tools for checking the fulfilment of the SP providing further evidence to the safety argument and (iii) some of the partners participating in the SafeCer project have extensive experience using these languages for specifying SP.

There is no explicit support to describe compliance with standards, however this can be stated as a requirement which can be attached to a component. The fulfillment of the requirement can then be indicated by matching it with a verification activity and outcome (e.g. also modelled in EAST-ADL). As can be seen in Table I, AADL, VERDE, and EAST-ADL are the most expressive of the considered meta-models and languages for SP since the majority of the SP considered can be expressed in them. Othello is also rather expressive, while CHESS-ML focusses on real-time contracts and failure propagation.

## IV. CONCLUSIONS

In this paper, we survey the classes of properties that are relevant to the safety certification of systems and a number of architecture and component-based modeling languages detailing which properties are supported. This survey was a fundamental starting point of the SafeCer project, which targets compositional certification of safety-critical systems.