MADE IN FRANCE

CLEARSY

Safety Solutions Designer

FORMAL ANALYSIS OF SOFTWARE

CLEARSY TECHNICAL OFFER
FORMAL ANALYSIS OF SOFTWARE

CONTACT@CLEARSY.COM

FORMAL ANALYSIS OF SOFTWARE

CLEARSY proposes a new innovative analysis approach to establish with mathematical proof that all or part of a software are compliant with respect to a functional or a safety requirement.

This approach establishes a direct formal link between the source code of the software and the properties of the system that integrates that software. It is now possible to detect any kind of noncompliance that may have been introduced in the design phase: from the identification of algorithms during the system definition phase, up to their concrete realization, taking into account possible implementation specific constraints.

This approach is particularly suitable when the traditional verification and validation activities show to be lacking:

- Scenario-based verification is possibly incomplete for systems having too many states.
- Bugs are discovered late in the development cycle, especially when they stem from errors from the system design.

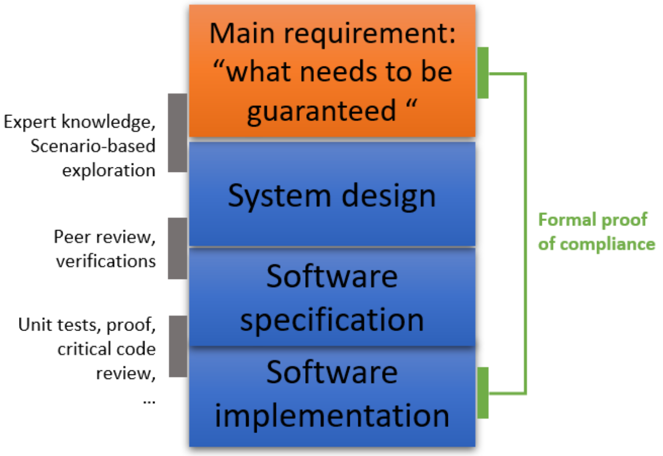
Main requirement: "what needs to be guaranteed "
System design
Software specification
Software implementation
Expert knowledge, Scenario-based exploration
Peer review, verifications
Unit tests, proof, critical code review, ...
Formal proof of compliance

Figure 1. The activity in the workflow.

In comparison, the formal analysis approach proposed by CLEARSY is complete: it is guaranteed to cover all possible functional behaviors, including as a matter of fact all system dysfunctions and failures that have not explicitly been discarded. This is the benefit of using a method based on mathematics and a property-based approach, instead of a case-by-case approach.

This approach may be applied **retroactively on an existing software**. Moreover, it is all the more benefitting to apply it systematically in the industrial process, so that **the analysis is moved upstream** the development. **Errors are detected at the earliest**, the correction steps and the tests to replay being reduced to the minimum.

The benefits of such a formal analysis numerous:

- To consolidate and lay down the functional and safety requirements: to state explicitly what guarantees the system shall enforce.
- To identify noncompliant behaviors: violations of properties that may have deep consequences on the safety or on the operation of the system integrating the software under analysis.
- Recover initial reasonings: the original intentions of the system designers.
- Lay down design choices: by finding their justification, thus providing a guarantee that the software under analysis is technically mastered.
- Uncover useless or obsolete complexity nodes: to simplify the software for improved performance and functional features (the simpler the software, the larger the possibility to be well-behaved).

INDUSTRAL INTEREST

This method is pragmatic, quickly providing concrete results for the project. With the results obtained from previous applications, CLEARSY has reached the conclusion that the formal analysis of software presents a real gain both for the supplier of a product and its operator. The approach benefits from being used on products that evolve over time, or that are widely deployed, or that integrate options based on their use.

For a supplier:

The company in charge of the product development will find its interest in having a more robust product and thus avoiding serious problems during exploitation, to ensure maintainability (of the knowledge and of the explicit reasons underlying choices and needs), and to guarantee the skills are transmitted. This approach also provides significant material to build the safety case for the product.

Moreover, it will be able to take advantage of the experience of applying this approach to **generalize** its deployment and thus create **an alternative to the traditional design and verification cycle** based on scenario analysis (an analysis without any guarantee regarding completeness, of course).

For an operator:

The operator, in the event of non-compliance, and therefore a problem with the operation, is also liable, has a strong motivation to apply this method.

Indeed, CLEARSY offers to conduct an analysis of the deliverables of the supplier of the operator. The benefits are to improve the technical mastery of the product and to obtain a systematic, safe and tooled methodology to **validate mathematically the conformity** of the software with respect to the needs for operation and safety.

Above all, **it avoids the occurrence of problems during operation**, which can be all the more serious if they affect safety and have serious consequences: operation is interrupted, financial penalties might be incurred, the public image of the operator is damaged, a conflict with the supplier.
Multi-supplier systems (interoperable systems) are a particularly relevant target for this type of approach. Indeed, the share of responsibilities between the various subsystems integrating software is all the more important. In case of incident, each supplier defends that its product is not incriminated. The proposed approach produces a mathematical evidence that the composition of the subsystems meets the system-level requirements.

TECHNICAL OVERVIEW

The method is based on an analysis based on properties that must be preserved. The main system requirement is decomposed down to a level of detail where the software variables are directly involved.

This method of analysis provides a modular and layered view of the software and the manipulated entities. Each software variable contributing to fulfill the requirement is identified. The role of these variables in the demonstration of compliance is clearly conveyed by properties linking each such software variable to real / physical / concrete elements interacting with the software. The reasoning establishing the compliance to the main requirement consists in finding a proof that the main requirement is a logical consequence of the properties.

The method demands also that all possible evolutions of the different physical and logical (software) elements be analyzed to verify that these evolutions preserve these properties. This constitutes the evidence that the system main requirement holds in every possible execution scenario, *including breakdowns and dysfunctions*.

ILLUSTRATION

Take the case of a safety-critical software in charge of issuing an authorization to open the doors of an automatic metro train.

The functional conditions to authorize opening can be many: train correctly positioned along the platform, fully stopped, servicing the platform in question, ongoing emergency opening request, etc.

Nevertheless, one thing is clear, the authorization to open the doors shall never be granted while the subway train is moving. This statement corresponds to the expression of a requirement, in this case a safety requirement. In this example, it is the main requirement. This property must always be true, we speak of invariant property.

To realize this, the software is based on more or less complex algorithms to estimate the train speed. This estimation can only be an approximation of the actual speed, guaranteed to a close delta. We see immediately that, to guarantee the main requirement, this algorithm should tend to overestimate the actual speed. Otherwise, it could conclude that the subway train is at a standstill when in fact it is in motion (non-zero speed), which could result in issuing a faulty authorization to open the doors.

Suppose now that the algorithms implemented to obtain this estimate are based on sensors capable of finely analyzing the rotational speed of the wheels (or axles). This information is useful for determining an overall speed of movement but it is not sufficient. In fact, one must also consider a sliding movement (obtained without rotation of the wheels). Otherwise we could wrongly conclude that a rolling stock whose wheels are not rotating is at a standstill while in fact it is still in motion, and enable an erroneous authorization to open doors.

So, the software component under design will integrate somehow the following modules:

- estimate of a rotational speed, say $vrot_{sw}$,
- estimate of a slipping speed, say $vslip_{sw}$,
- computation of a global train speed, say v_{sw}, from the previous speeds.

This constitutes what we previously called a layered view of the software: $vrot_{sw}$, and $vslip_{sw}$ are lower-level entities used to determine the higher-level information v_{sw}.

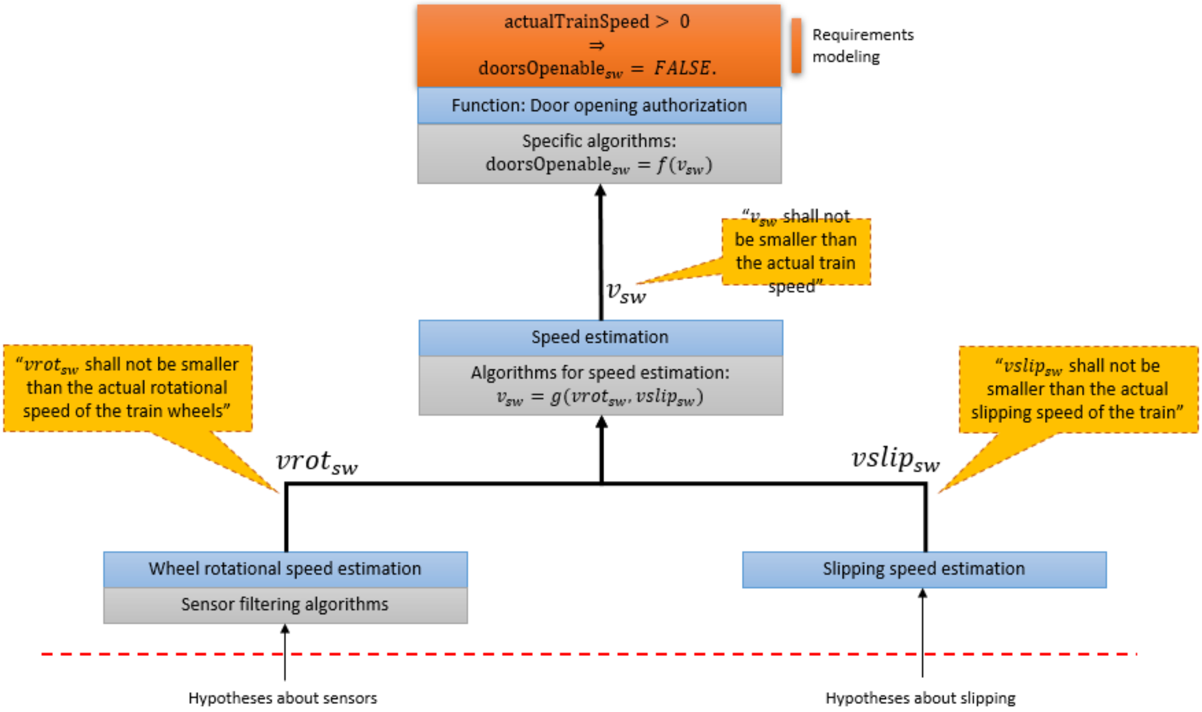
actualTrainSpeed > 0 ⇒ doorsOpenable$_{sw}$ = FALSE.
Requirements modeling
Function: Door opening authorization
Specific algorithms: doorsOpenable$_{sw}$ = $f(v_{sw})$
“v_{sw} shall not be smaller than the actual train speed”
v_{sw}
Speed estimation
Algorithms for speed estimation: $v_{sw} = g(vrot_{sw}, vslip_{sw})$
“$vrot_{sw}$ shall not be smaller than the actual rotational speed of the train wheels”
“$vslip_{sw}$ shall not be smaller than the actual slipping speed of the train”
$vrot_{sw}$
$vslip_{sw}$
Wheel rotational speed estimation
Sensor filtering algorithms
Slipping speed estimation
Hypotheses about sensors
Hypotheses about slipping

Figure 2 – Example: layered, modular view of the software and of the properties to be guaranteed.

TOOLS

CLEARSY proposes to support this approach with a tool capable of supporting system modeling and formal proof of properties. The "Event-B " modeling deployed via the Atelier B industrial tool fulfills these characteristics and has been used in CLEARSY's reference projects in the field.

Nevertheless, the **method is not tool-specific**. CLEARSY has already conducted some analyses by using other tools, the motivation being to employ tools already used by the client, thus facilitating the interaction and the subsequent adoption of the approach by this client.

The interest of supporting the process with a tool is manifold:

- Ensures that the invariant properties of the software are expressed with the appropriate level of detail and accuracy.
- Ensures there is no logical fault in pen and paper proof so that a demonstration of compliance is achieved, supported by a certified tool.
- Finds implicit hypothesis used in the reasoning and make them explicit: Atelier B, or other similar tools, has no built-in domain specific knowledge.

SYNTHESIS

The diagram below presents in a synthetic way the different steps described above to obtain the formal proof that a software is compliant with respect to a requirement.

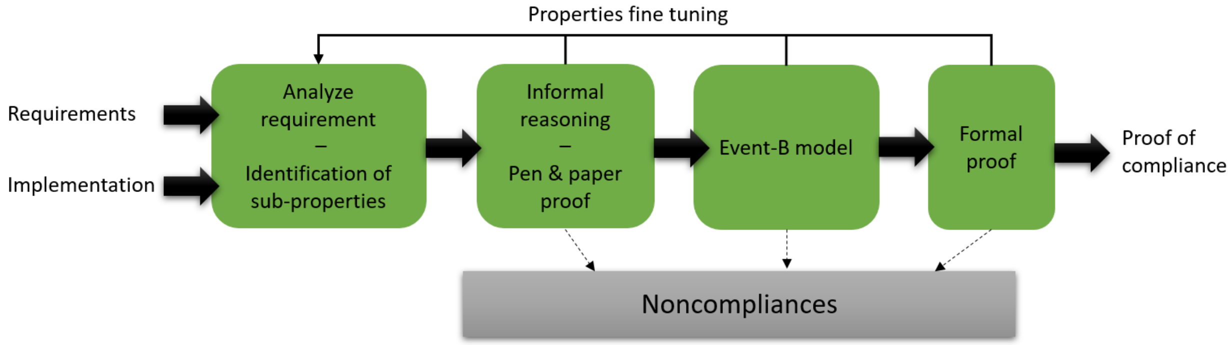

Figure 3 – Synthesis

INPUTS

The team in charge of conducting the analysis needs the following two key elements:

• **The main requirement**, that against which one seeks to evaluate the conformity of the algorithms implemented. In general, it is a set of properties resulting from the analysis of the catastrophic events but it can also be a functional property, unrelated to safety objectives.

• **A faithful representation of the software code** implementing this feature: the software specification, an abstract model of the code if it has been developed using formal methods (B-software for example), or directly the code itself.

In practice, the analysis is achieved more efficiently if the team has a channel for **on-demand meetings with a professional familiar** with the software and more specifically with the feature under study.

DELIVERABLES

The deliverables of a formal analysis of software are a collection of documents containing:

- The precise scope of the study,
- The hypotheses made on the inputs of the analyzed function (incoming messages, sensors, results from another function, etc.),
- The properties expressing the functional and safety requirement involving the software variables,
- The proof of compliance to the main requirement,
- Any possible noncompliant situations as well as the elements necessary to conduct their assessment.

Also delivered are the formal models developed with Atelier B (or any other tool chosen for the delivery). These models can be maintained and used to evaluate possible software evolutions by integrating them in the models even before starting the software design cycle.

These models and the accompanying proof guarantee mathematically the soundness and completeness of the conducted reasoning.

COMMERCIAL REFERENCES

Our customers are big companies and prime contractors in the railway market.

ALSTOM
RATP
SNCF
MTA New York City Transit

This approach has already been applied to safety-critical software qualified SIL4 according to the CENELEC - EN 50128 standard: a railway product with more than 12kloc and a 100-pages software specification. In terms of volume, the conducted analysis required globally 1.5 man-year to experts. The project quickly delivered results and steadily delivered output to the customer all along the project.

A similar approach has also been applied in the automotive domain for the validation of embedded software responsible to manage an anti-theft device mounted on the steering column of motorized vehicle. Another similar study was conducted in the micro-electronics domain, to demonstrate the compliance of smart card specifications with the level 5+ of the Common Criteria standard (used for computer security certification).

CLEARSY has also experience for inter-system analysis of complex SIL4 applications applying a formal approach. This kind of analysis aims to build a demonstration that the global safety of the system is guaranteed, assuming that each sub-system guarantees a set of desired properties (and this can be the subject of a formal analysis of software as presented in this document). Such approach has been conducted to prove formally the absence of collision and of derailment of trains from a set of inter-system specification documents (assuming that each supplier correctly implements such specifications) for the following CBTCs: New York line 7, RATP (Octys), and the ERTMS system for SNCF.

FORMAL ANALYSIS OF SOFTWARE

CLEARSY

Safety Solutions Designer

OFFER
2020

320 AVENUE ARCHIMEDE
LES PLEIADES III BAT A
13100 AIX-EN-PROVENCE - FRANCE

TEL. +33 (0)4 42 37 12 70 FAX. +33 (0)4 42 37 12 71

WEB. contact@CLEARSY.com
WWW.CLEARSY.COM