# Multi-Facets Contract for Modeling and Verifying Heterogeneous Systems

A. Abdelkader Khouass[0000−0002−2075−602X] 1,2, J. Christian Attiogbé[0000−0002−7815−1752] 1, and Mohamed Messabihi[2]

[1] University of Nantes, LS2N CNRS UMR 6004, France
christian.attiogbe@univ-nantes.fr
[2] University of Tlemcen, LRIT, Algeria
abderrahmaneabdelkader.khouass@univ-tlemcen.dz
mohamedelhabib.messabihi@univ-tlemcen.dz

**Abstract.** Critical and cyber-physical systems (CPS) such as nuclear power plants, railway, automotive or aeronautical industries are complex heterogeneous systems. They are perimeter-less, built by assembling various heterogeneous and interacting components which are frequently reconfigured due to evolution of requirements. The modeling and analysis of such systems are challenges in software engineering. We introduce a new method for modeling and verifying heterogeneous systems. The method consists in: equipping individual components with *generalized contracts* that integrate various *facets* related to different concerns, composing these components and verifying the resulting system with respect to the involved facets. We illustrate the use of the method by a case study. The proposed method may be extended to cover more facets, and by strengthening assistance tool through proactive aspects in modelling and property verification.

**Keywords:** Heterogeneous systems · Components assembly · Generalized contracts · Modeling and verifying · Formal analysis.

## 1 Introduction

Critical and cyber-physical systems (CPS) that exist in large industries, such as nuclear power plants, railway, automotive or aeronautical industries are complex heterogeneous systems. They do not have a precise perimeter, they are open and often built by assembling various components. Their complexity forces one to have a wide variety of heterogeneous components, frequently reconfigurable due to requirements evolution.

With the advent of concurrent and distributed systems, Component Based Software Engineering (CBSE) [10] has known a high interest. The construction of a distributed system involves several specific components; this requires rigor, methods and tools. The involved components may deal with various *facets*. A facet is a specific concern or a property such as data, behaviour, time constraints, security, etc[3]. Therefore, if the integration of components into a global system is not mastered, it may generate considerable time losses and overcharges, because of inconsistency of requirements, incompatibility of meaning and properties, late detection of composition errors, etc. For these

[3] This idea also appears as the separation of concerns in aspect-oriented programming/design

reasons, the modeling and formal analysis of such heterogeneous systems are challenging. The use of efficient methods and techniques is required to face these challenges.

We aim at studying and alleviating the difficulties of practical modelling and integration of heterogeneous components. We propose a novel approach based on contracts for modeling and verifying complex and heterogeneous systems. Our approach (named "ModelINg And veRifying heterogeneous sysTems with contractS" (Minarets)) consists in modeling and verifying a system with the concept of *generalized contracts*. The contract is generalized in the sense that it will allow one to manage the interaction with the components through *given facets*: the properties of the environment, the properties of the concerned components, the communication constraints and non-functional properties (quality of service for example). The use of contracts during the verification reduces the complexity of the analysis of heterogeneous systems; moreover, the structuring of contracts with facets and priority of properties makes it possible to decrease the difficulty of checking heterogeneous systems, to save time and to increase performance during verification.

The rest of the article is structured as follows. Section 2 introduces the modeling and verification methodology. In Section 3 we illustrate our approach with experimentation and assessment. Section 4 provides an overview of related work, and finally, Section 5 gives conclusions and future work.

## 2 Modeling and Verifying Using Generalized Contracts

An issue to be solved for heterogeneous systems is that, the interface of involved components should be composable. For the sake of simplicity of the composition, we adopt the well-researched concept of contract which is therefore extended for the purpose of mastering heterogeneity of interfaces. Moreover, for a given system we will assume *agreed-upon facets* such as data, functionality, time, security, etc.

In this work we chose the PSL language [1] to specify contrats. PSL is a formal language for specifying properties and behaviour of systems. It is an extension of the Linear Temporal Logic (LTL) and the Computation Tree Logic (CTL). PSL could be used as input for formal verification, formal analysis, simulation and hybrid verification tools. PSL improves communication between designers, architects and verification engineers. We use the ALDEC Active-HDL [4] tool that supports PSL.

For experimentation purpose, we use ProMeLa and SPIN [6] to model and verify components. SPIN is an automated model checker which supports parallel system verification of processes described with its input PROtocol MEta LAnguage (ProMeLa). We also use the model checker UPPAAL [4].

### 2.1 Definitions

We extend the traditional A-G contract with the purpose of mastering the modeling and verification of complex and heterogeneous systems.

[4] https://www.aldec.com/en/products/fpga_simulation/active-hdl

**Definition 1 (Generalized contract).** *A generalized contract is a multi-faceted Assume-Guarantee contract. It is an extension of contract, structured on the one hand with its assume and guarantee parts, and structured on the other hand according to different clearly identified and agreed-upon facets (data, functionality, time, security, quality, etc.) in its assume or guarantee.*

The generalized contract will be layered to facilitate properties analysis. Every layer will have a priority. Therefore, an analysis of a facet may be done prior to another facet.

**Definition 2 (Well-structured component).** *A well-structured or normalised component is a component equipped with a generalized contract, acting as its interface with other components.*

Normalising a component $C_i$ consists in transforming $C_i$ into a component equipped with a generalized contract. A multi-faceted A-G contract will be expressed in PSL.

### 2.2 Outline of the Proposed Method

The working hypothesis is that a heterogeneous system should be an assembly of *well-structured components* (see Def. 2). The method that we propose (Minarets) consists in, given a set of appropriately selected or predefined elementary components, normalizing these input components prior to their composition, building a global heterogeneous system, and finally analysing this global system with respect to the required properties.

For this purpose there are many issues to be solved:
*i)* Elementary components are from various languages and cover different facets, a pragmatic means of composition is required. We consider PSL as a wide purpose expressive language to describe generalized contracts. Each component will be manipulated through its *generalized contract* (see Def. 1) written in an appropriate language.
*ii)* Global properties are heterogeneous; they should be clearly expressed, integrated and analysed; they will be expressed with a wide purpose language such as PSL; we will decompose them according to the identified agreed-upon facets and spread them along the analysis of composed components.
*iii)* Composition of elementary components should preserve their local requirements and should also be weakened or strengthened with respect to global-level properties. For instance, some facets required by an elementary component could be unnecessary for a given global assembly, or some facets required at a global assembly may be strengthened at a component level.
*iv)* Global properties require heterogeneous formal analysis tools; this generates complexity. We choose to separate the concerns, so as to target various tools and try to ensure the global consistency.
*v)* Behaviours of components should be composable.

The Minarets method integrates solutions to these issues, as we will present in the sequel. We adopt a correct-by-construction approach for the assembly of components. Therefore, local compositions should preserve required properties of components. In the same way, global properties may impact the components; therefore, global properties are decomposed and propagated through the used components when necessary.

## 3 Case Study

We consider a case study of an automotive industry: a car painting workshop. This workshop is composed of three main components; a control station (CS) which manages a paint station (PS) and an automatic robot painter (RP). These components interact with each other to achieve the painting process correctly; that means to paint the cars with the desired color, in time, without any damages and without wasting color. More details can be found in [7].

Now we follow step by step the Minarets method to model and verify this system.

**Step 1** ***(modeling $M_1$).*** Express the informal global requirements and global properties. They could be expressed with any desired language, even the natural one. The only goal here is to clearly state the requirements of the given system.
**Global requirements:** cars data are provided (type, color with RGB quantity, painting time); sufficient RGB colors are in the tanks; freeing time must be defined (the freeing time is the release time of the painting station) .
**Global expected properties:** the system respects the correct RGB dosage; there is no loss of color; the painting time should be equal to the given time; the painting is done without damages (respect of car dimensions, it is known from the car type); freeing time must be equal to the given freeing time for each car; painti,g shloud be stopped and notified when there is no sufficient color; painting station status must be free before use, and busy when a car is inside; the painting starts after the end of configuration, and it finishes when the painting time is equal to given painting time.
**Step 2** ***(modeling $M_2$).*** Formalise the required global properties with appropriate expressive formal specification languages (PSL in our case), we obtain the following formalized global properties.
**Global requirements:**

```
get_type = true; get_color = true; get_painting_time = true;
/* the painter never start working without car dimensions (get_type)*/
get_freeing_time = true;
R_tank_quantity >= R_GivenColor_quantity;
...
```

**Global properties:**

```
R_color_PaintedQuantity = R_GivenColor_quantity;
painting_time = GivenPainting_time; car_type = given_type;
freeing_time = given_freeing_time;
/*each color is controlled separately*/
if (RGB_tank_quantity <= RGB_GivenColor_quantity)
Then (stop_process and warning_message);
painting_time = GivenPainting_time imply painting =  finished;
...
```

**Step 3** ***(modeling $M_3$).*** Here we have to model or select the components. We use UPPAAL and ProMeLa to model the components RP, CS and PS. Figure 1 shows the models of the components CS and PS within both environments.

**Step 4** ***(modeling $M_4$).*** We decompose the global properties with respect to the facets that we considered (Data, functionality, time, security); we obtain the *generalized contract* decomposed

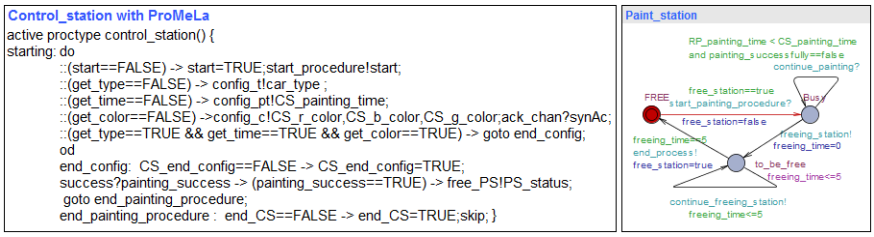


**Fig. 1.** CS model with ProMeLa and PS with UPPAAL

with the facets. Figure 2 shows a part of the faceted and formalized properties (the guarantee part); this will emphasize the concern to be dealt with later on.

**Step 5** ***(modeling $M_5$).*** We express the structured and formalized properties with the PSL language as depicted in Figure 2.

```
for RP in painting_status
  property p0 :always
               (get_type = true  and
                get_color = true and
                get_time = true )
   Data : assert p0;
end
```

```
for RP in painting_status
   property p1 :always
        (PS in Busy_status  and
         CS in end_configuration and
         R_tank_color >= CS_R_GivenColor and
         B_tank_color >= CS_B_GivenColor and
         G_tank_color >= CS_G_GivenColor)
   Security : assert p1;
end
```

```
property p5: always  (CS_time_painting = given_time -> RP in OFF_status)
Time: assert p5;

property p6 : always (deadlock -> (PS in FREE and RP in OFF and CS in End ))
Functionality assert p6;
```

**Fig. 2.** A part of faceted and formalized global property with *PSL* and *Aldec Active-HDL*

**Step 6** ***(modeling $M_6$).*** Normalizing the individual components (see Def. 2). We integrate the assumptions and guarantees of each individual component. Figure 3 shows the normalized individual components.

**Step 7** ***(modeling $M_7$).*** According to the result of *Step 4 (modeling $M_4$)*, we may add a facet of the global property to a component, or ignore some of its facets, if necessary. In the current case study it is not necessary to add or ignore a facet (see Figure 3).

**Step 8** ***(modeling $M_8$).*** We attribute the following priorities to each facet (data=1, security=2, time=3 functionality=4; were 1 is the highest priority). We obtain ordered layers with respect to facets and properties as well. The order of layer is mentioned in Figure 3. The verification by layer allows one to verify the contracts by order; from a very important layer (primary) to a less important layer (secondary). If the behaviour of our system does not satisfy a primary layer of contract, then, it is not necessary to continue the verification with the other layers.

**Step 9** ***(verification $V_1$).*** We have to check the appropriate functioning of each normalized individual component if tools exist for that and if the required data are available. As we use ProMeLa we have an adequate tool (SPIN) but, the only component CS modelled with ProMeLa cannot be verified without composition with its environment.

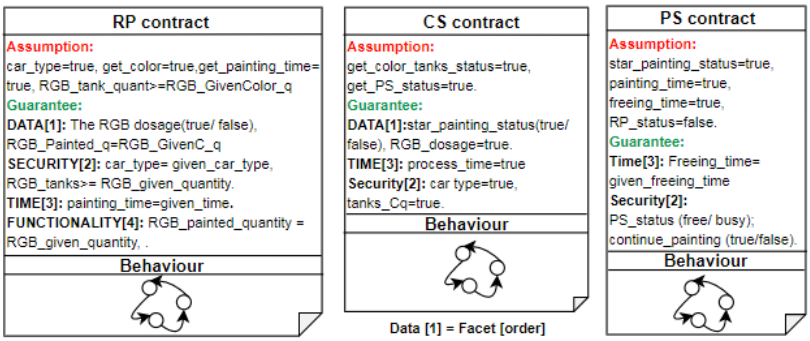


**Fig. 3.** Normalized components with the prioritised facets

**Step 10** ***(modeling $M_9$).*** As the checking of the normalized individual component CS cannot be carried out, due to the need of composition with its environment; we translate the ProMeLa component CS to UPPAAL using the algorithm presented in our RR [7], we obtain a component CS ready for composition (see Figure 4).

**Step 11** ***(modeling $M_{10}$).*** We compose the translated component CS with the other components PS and RP with the UPPAAL tool. As we focus on behaviours expressed with LTS, the composition results in parallel composition.We obtain the composed system depicted in Figure 4.

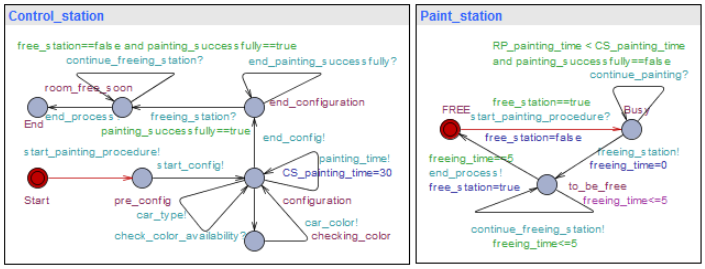


**Fig. 4.** CS and PS models after the component translation (in UPPAAL)

**Step 12** ***(verification $V_2$).*** We translate the *generalized contract* (only the desired properties to be verified) of each individual component, from *PSL* to the UPPAAL language. We obtain properties ready for verification with UPPAAL. The following property is an extract of the translation from the RP component.

```
A[ ]  Robot_painter.painting imply get_type == true
    and get_color == true and get_time == true
```
[5]

**Step 13** ***(verification $V_3$).*** We verify the properties of the same layer together; i.e. to verify each component by layer: data, security, time, functionality; also, the primary properties before the secondary. At the end of this step, we obtain the verified components. Figure 5 shows the verification status of the translated properties.

[5] were "A [ ] Prop" denotes the "always property".

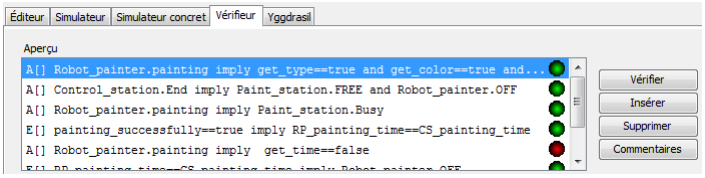


**Fig. 5.** Status of properties verification with UPPAAL

***Assessment*** This experimentation was conducted in order to improve our method. We have considerably detailed the steps of the method when thinking thoroughly about its applicability through the case study. Despite the success of applying in preliminary trivial exercises and on this case study, and the reproducibility of the steps, it appears that more tool assistance is needed to guide the users. The experimentations even if not yet scalable to industrial cases, give the opportunity to tune the method steps and design some translators to help in modeling and verification. We are aware of the impact of treated facets on interactions between various tools; but, more lessons from various case studies will help to promote good practices through tools. Our report [7] contains more detailed experimention examples.

## 4 Related Work

Several works contributed to the heterogeneity issues, by proposing different methods and techniques. Interface theory [2] and theories of contracts [5] are based on the contracts of components and the pieces of information given by their interfaces; they use intensively contracts as well as the behaviours; they propose techniques and methods to promote concurrent development.

The Ptolemy project [8] proposes an approach of interaction between heterogeneous components based on models of computation (MOC); here the heterogeneity is linked to different models of computation. From our point of view, this composition method is heavy, general and constrains the use of contracts, especially when one deals with a small net of communicating processes [3]. In our Minarets method we aim to exploit more advantages of contracts in a simpler and explicit way to model and verify the assembly of heterogeneous components.

Our modeling and verification method is based on previous results [3] for heterogeneous components composition; this deals with the formal composition of heterogeneous components, and it is based on the dynamic behaviour of the components where labelled transition systems have been used as common semantic domain; the method which was proposed there is focused on the use of an algebra of operators to manage the communication mechanisms between the assembled components. This approach is supported by the aZIZA tool [6].

As a part of the B.I.P project, [9] proposes a technique with three layers: "*Behavior, Interactions, Priorities*" (B,I,P). It is a low level solution that deals with the interaction between components; it focuses on the composition of the system with the different interaction semantics; however, unlike Minarets, this approach didn't deal with the heterogeneous components modeling but it deals with the property analysis and composition of components using the B.I.P language.

## 5 Conclusion

We have proposed the Minarets method for complex and heterogeneous systems modeling and analysis; it is based on an extension of the traditional contracts, resulting in generalized con-

[6] https://aziza.ls2n.fr/ [3]

tracts used as standard interfaces between components. Generalized contracts are structured with several facets, depending on the concerns or the properties that we are dealing with. Minarets emphasizes the stepwise composition of heterogeneous components through their generalized contracts. We have shown how one can reduce the complexity of the global modeling and the global analysis of complex and heterogeneous systems.

We illustrated our approach with an example of a painting workshop in the automotive industry domain; it involves different facets (data, functionality, security, time). We have checked the properties concerning the various facets. Concerning verification, generalized contracts are first expressed in PSL, then translated into input languages of UPPAAL/SPIN model checkers.

Future works address not only the scalability but also the study of various policies for the composition of contracts; it is important to verify the components composition as we done through the used tools but, to improve the method we are defining in a formal way the semantics of parallel composition of our normalized components. This will furthermore strengthen the foundations of the proposed method, and enable the contract management tool construction.

## References


1. IEEE standard for Property Specification Language (PSL). IEEE Std 1850-2010 (Revision of IEEE Std 1850-2005) pp. 1–182 (2010). https://doi.org/10.1109/IEEESTD.2010.5446004
2. de Alfaro, L., Henzinger, T.A.: Interface theories for component-based design. In: Henzinger, T.A., Kirsch, C.M. (eds.) Embedded Software, First International Workshop, EMSOFT 2001, Tahoe City, CA, USA, October, 8-10, 2001, Proceedings. LNCS, vol. 2211, pp. 148–165. Springer (2001), https://doi.org/10.1007/3-540-45449-7_11
3. Attiogbé, J.C.: Mastering heterogeneous behavioural models. In: Ouhammou, Y., Ivanovic, M., Abelló, A., Bellatreche, L. (eds.) Model and Data Engineering - 7th International Conference, MEDI 2017, Proceedings. LNCS, vol. 10563, pp. 291–299. Springer (2017), https://doi.org/10.1007/978-3-319-66854-3_22
4. Behrmann, G., David, A., Larsen, K.: A Tutorial on UPPAAL. vol. 3185, pp. 200–236 (01 2004). https://doi.org/10.1007/978-3-540-30080-9_7
5. Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J., Reinkemeier, P., Sangiovanni-Vincentelli, A.L., Damm, W., Henzinger, T.A., Larsen, K.G.: Contracts for system design. Found. Trends Electron. Des. Autom. **12**(2-3), 124–400 (2018), https://doi.org/10.1561/1000000053
6. Holzmann, G.J.: The SPIN Model Checker. Addison-Wesley (2004)
7. Khouass, A., Attiogbé, C., Messabihi, M.: Multi-facets contract for modeling and verifying heterogeneous systems. CoRR **abs/2012.13671** (2020), https://arxiv.org/abs/2012.13671
8. Lee, E.A.: Disciplined heterogeneous modeling - invited paper. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) Model Driven Engineering Languages and Systems - 13th International Conference, MODELS 2010, Proceedings, Part II. LNCS, vol. 6395, pp. 273–287. Springer (2010), https://doi.org/10.1007/978-3-642-16129-2_20
9. Sifakis, J.: Rigorous system design pp. 292–292 (2014). https://doi.org/10.1145/2611462.2611517
10. Tiwari, U.K., Kumar, S.: Component-Based Software Engineering: Methods and Metrics. CRC Press (2020)