Modeling Frames

Stefan Klikovits
University of Geneva
Carouge, Switzerland
Stefan.Klikovits@unige.ch

Joachim Denil
University of Antwerp
Antwerp, Belgium
Joachim.Denil@uantwerpen.be

Alexandre Muzy
CNRS, I3S
Université Côte d'Azur, France
Alexandre.Muzy@cnrs.fr

Rick Salay
University of Toronto
Toronto, Canada
rsalay@cs.toronto.edu

***Abstract*—Modeling activities such as calibration, verification and validation are often executed in under-specified environments. This hinders reproducibility, reduces re-usability and generally decreases the modeling precision and quality. This paper describes a framework for the definition of model frames. Model frames capture the context an activity/model is executed in and facilitate re-use, replacement, validation and verification of models. We show the application of the frames approach onto a real-world example, introduce several kinds of frames and show their application on this case study.**

I. Introduction

When it comes to systems modeling, much effort is spent on vigorously describing, verifying and validating models. The description of these activities' execution environment however lacks the required emphasis and model interactions are often performed in under-specified environments, where execution contexts are only partially documented at best.

Motivated by scenarios such as the selection of suitable models from model libraries, the (de-)composition of models, sound environment documentations are gaining importance. These scenarios require a clear definition of model's context to test the validity and suitability of the resulting compositions. A more concrete example is the safety certification process: While repeated re-certifications are possible, the process requires a lot of resources (effort, time, money). Using a component's valid context specification, the proof of correctness can be reduced by showing that modified/replaced components have the same behaviour within equal environments and hence maintain their properties.

Zeigler [1] discovered early on that each model is surrounded by a structure which captures how it can be interacted with. This notion of an experimental frame (EF) helps documenting the information that is necessary to execute the model itself. He primarily focussed on the activity of model execution (not to be confused with model activity) and simulation. Denil et al. [2] recently observed that a model's frame depends on the activity that is performed and describes why different activities require different frames.

This paper extends the activity-based notion of Denil et al. and introduces a clear record of the context that the process is executed in. The definition of a generic *modeling activity frame* (MAF) builds a clear foundation that allows for the specification of model activities. MAF help in the creation, re-use and adaption of models as well as the model-based system synthesis.

We illustrate the applicability of our approach using the example of a traffic light (TL) system whose documentation is unavailable. The TL regulates oncoming vehicle traffic next to a pedestrian crossing as shown in Figure 1.

While the system itself might appear trivial at first, its environment is not. Checking the model's validity relies on specific assumptions. It is necessary, for example, to state how much data is assumed to be necessary to truthfully certify the model's correctness or whether the time of day has any influence on the TL's behaviour. Without clear specification of these assumptions, the modeling processes might not be clearly reproducible. The need for detailed frames specifications is underlined when system models are used to compose larger systems-of-systems, e.g. when combining traffic lights to model, verify and analyse the traffic behaviour of an entire city.

This paper is structured as follows: Section II introduces the frames concept. Section III documents the details of the running example. Section IV describes a number of different frame types and Section V formalises the relations between the frame's components and different types of frames. Section VI positions our research into a broader context and Section VII concludes.

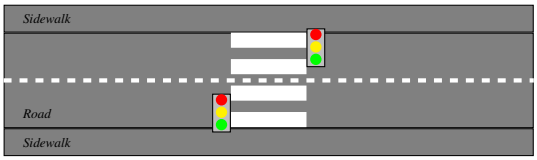

Figure 1. Schema of the traffic light scenario

II. Frames

Modelling frames capture all information that is required to capture both *context* and *modelling activity*. Considering a system (e.g. a traffic light), the latter can be the *verification* of corresponding model's behavior or the *calibration* model's parameters, etc. Model activities correspond to the different phases of a model's life cycle, such as *creation*, *validation*, *verification* and *optimization* of a model, etc.

An activity can be composed of a set of sub-activities. Then, it is necessary to capture the execution order of these sub-activities and which data are required (*inputs*) and produced (*outputs*). For example, a TL's calibration process consists of two steps: 1) *system data collection* and 2) *parameter*

calculation. The latter accounts for the *observed data* and produces calibrated *parameter values*.

Alternatively, an activity can be *atomic*, which states that the developer chose to not further split it. This applies for example to the data collection process which describes the recording of the TL's phases and each phase's length. Since it is trivial, no further decomposition is needed.

In addition to an activity's process, it is necessary to capture a model's execution environment. This includes all necessary information, expressed as the *objective(s)*, *assumptions* and *constraints*. In the TL scenario, the calibration process' objective is to find the correct values for the phase lengths so that the model reflects the system under study. Assumptions might specify how much data is needed for the calibration process, e.g. 24 hours to verify unchanged behaviour during the night, and a constraint might specify the minimum precision (e.g. ± 0.1 sec).

The concepts are visually depicted as an ontology in Figure 2. As can be seen, a frame captures the required system and environment information. The *frame* itself consists of a *modeling activity* (defined by *inputs*, *outputs* and a *process* description), a *context* (consisting itself of *objectives*, *assumptions* and *constraints*) and a set of zero or more sub-frames.

As there are many different modeling activities, the figure provides only a few examples. The "classical" activity of *modeling* (model creation) corresponds to linking a system and a model, it is displayed more prominently.

Each modeling activity is related to a *system activity*. Depending on the action performed, a system can be either analysed, i.e. the system remains unmodified (e.g. model creation) or synthesized, i.e. the system itself is changed (e.g. during model driven system calibration). In this paper we focus on system analyses.

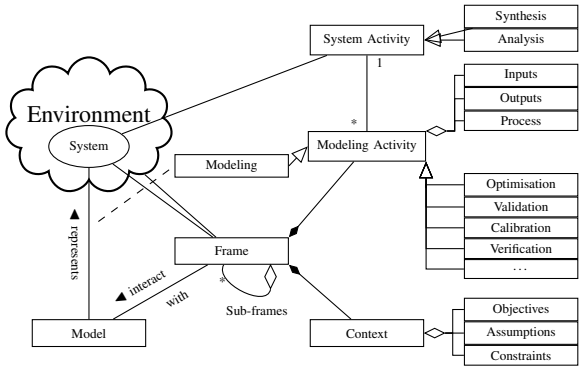

Figure 2. An ontology of the frame concepts

III. Traffic Light System

Applying the above definitions to the traffic light system allows us to define various frames. To not exceed the scope of this publication, we limit ourselves to two frame descriptions (the *model creation frame* and the *validation frame*).

For clarification of the modeling processes below we first provide some additional information about the system and its assumptions: Both traffic lights in the system are connected to the same controller and always show the same phase. Apart from the equality in the behaviour of the two traffic lights, we also assume that there are only three phases (red, green and yellow) that occur in the same order and that the length of these phases is fixed (i.e. does not change under any circumstances).

A. Creation Frame

Our first task is to create a model of the "real-life" TL system. To aid the creation of the model and make the process formally traceable, we define a *model creation frame*. Based on the definitions in the previous section, we identify the process required to model the system and formalize the context of the model creation process, namely all assumptions and constraints we choose on this process.

Since the modeling frame is a very general concept, the high-level objective is stated as O_1**:** *Create a model in order to learn about the TL timing*. Next, we try to capture assumptions towards the system:

A_1 *Time of day has no influence on the behaviour.*
A_2 *Weather has no influence on the behaviour.*
A_3 *The sequence of phase colours is fixed.*
A_4 *The phase lengths are constant.*

We also specify constraints on our activity, namely that C_1**:** *The model needs to have ample precision*. C_2**:** *The model must be represented as state machine*.

We remind the reader that this list is non-exhaustive and that a complete list of objectives, assumptions and constraints would exceed the scope of this work.

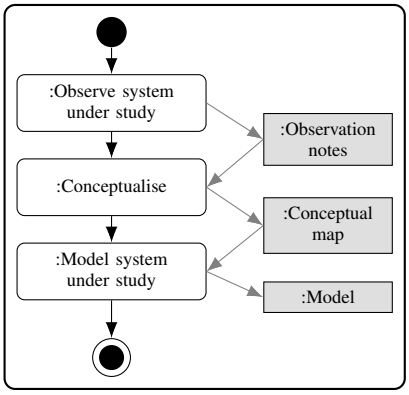

Figure 3. Modeling frame process for the traffic light as UML activity diagram

Following the definition of the context, it is necessary to specify the process that has to be followed to perform the modeling activity. Although many formalisms are applicable, in this work the UML 2.0 Activity Diagram formalism was chosen for modeling the frame processes. Figure 3 displays the process that is required to create a model and the artefacts that are created and used alongside. Reaching the end state means the *Objective has been achieved under the given assumptions while respecting the constraints*.

After the execution of the process (Figure 3) in the given context, the finished model of the system is shown in Figure

4. It displays the order of colours and contains parameters for the phase transition time parameters T_r, T_g, and T_y.

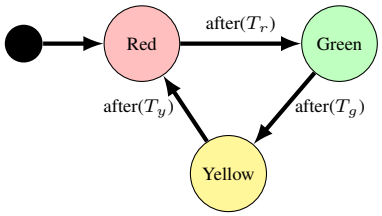

Figure 4. The state automaton of the traffic light

B. Validation Frame

The validation frame formalises the validation activity. Within the frame (a subset of) the modeling frame's assumptions are tested for correctness. If the assumptions hold, the model can be seen as valid with regards to the properties under consideration.

The validation frame's process is split into several subtasks that can be executed in parallel. Each subtask has its own context and process defined in a separate frame. Figure 5 displays the process defined to validate two assumptions of the modeling frame above.

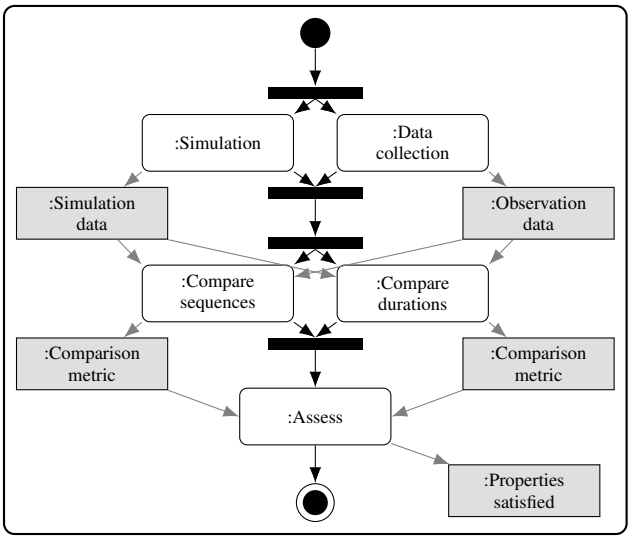

Figure 5. Validation frame process for the traffic light as UML activity diagram

The frame tests assumptions A_3 and A_4, namely that the sequences of the light phases are set to *Red* → *Green* → *Yellow* → *Red* ...(`Compare sequence`) and that the transition times T_r, T_g, and T_y are constant (`Compare durations`).

A part of the context of the validation process is given as follows:

O_1 Assert that colour sequence is fixed.
O_2 Assert that light transition times are constant.
A_1 48 hours of data gathering are sufficient.
A_2 The system runs in standard (automatic) mode (i.e. it is not remote influenced).
A_3 Simulation speed does not influence measurements.
C_1 Precision of time measurements has to be in seconds or finer.
C_2 The output of the simulation and data collection are `(colour - time)` tuples.

Contexts can be written directly to the diagram, e.g. using UML notes. To increase the diagram's legibility, we decided against their use in the figures here.

As each subtask in the process is defined by its own frame, each step should specify its own context. For spatial reasons we will not provide descriptions for each subtask, as the contexts are subsets of the validation frame's context.

IV. Frame Types

When comparing the traffic light scenario to other case studies it becomes evident that certain activities and frames tend to reoccur. This means that while adjusted to each individual scenario, the basic process concepts reoccur. In the following we provide a (nonexhaustive) list of frames that are essential for many projects and hence worth of study. We remind the reader that the presented frames are blueprints and need to be adapted to each new situation.

A. High-level Frames

Next to the model creation and validation frames described above, the following non-exhaustive list of high-level frames has been discovered. Each of these activities consists of several sub-activities. Due to spatial constraints, we do not describe the processes in detail, but restrict ourselves to the list of subtasks.

Calibration frame: Calibration frames modify parameter values so that the model has close similarity to the real system. The steps involved in this activity are *system data collection*, *data analysis*, and *parameter calculation*.

Verification frame: A verification frame's purpose is the review of the model's correctness in terms of meeting the requirements, i.e. that the model can fulfil its purpose and produce the required information. In the TL scenario a verification frame could test whether the model's precision is as high as specified. The verification frame's sub-processes are *model simulation*, *data collection*, *data comparison*, and *result assessment*.

Optimization frame: The optimization frame is responsible for adapting the model's parameters in order to improve the model. The improvement itself can be of different kinds (i.e. better execution performance, higher prediction precision, etc.). The process itself is a loop, where the model's parameters are iteratively changed, the model is executed and the results analysed until the execution reaches the predefined quality or a timeout. Sub-steps include *parameter setting*, *model execution*, *data comparison* and *result assessment*.

Experimentation frame: Experimentation is the task of executing a model to test hypotheses, draw conclusions or gather data for other purposes. It is important to capture the goals of the execution, as all activities (*model setup*,

measurement setup, *data gathering*, *data analysis*) have to be adapted towards these goals.

B. Basic Frames

The frames above clearly show that some subframes occur repeatedly. One prominent example is the task of data collection, which is required in almost every activity. While each instance of this activity might differ due to different data being gathered, the base concept remains the same.

In this section will describe three examples of such *base frames*. They are building blocks of more complex (high-level) activities and reoccur repeatedly. Base frames are not necessarily atomic, but often on a lower level than the frames in the previous section.

Model execution frame: Execution means that a model is brought into active state using a *setup*, whose exact process depends on the frame's objective. Subsequently the model is *run* using specific inputs, while at the same time the model's output is being *recorded*. As this frame is very generic, a clear context definition is paramount.

Data collection frame: Data collection is the activity of observing the system and recording information. Specifically, the subtasks include the choice of data being collected (*data collection specification*) and the setup of the measurement structures (*measurement setup*). Next, during the *system observation* phase the raw data is collected and subsequently corrected, cleaned and adapted to correct format (*data adaptation*).

Data comparison frame: Data comparison is a task that usually follows a data collection or model execution step. Within this frame two or more records of data are examined for similarities and differences. The activity consists of the *data adaptation*, where the data is brought into comparable format, the actual *data comparison* itself and the *assessment* of the comparison results. Within this frame it is important to clearly specify the comparison strategy and assessment method as constraints and the success criteria as objectives.

V. Formalizing Frames

Above we have given the intuition and examples of frames. In this section, we propose first steps toward a formalization of frames. The objective of the formalization is to provide a basis for tasks such as reasoning about modeling activity re-use, composition and consistency.

A. Frame Definitions

Having established a generic relation between the individual frame components, this section will provide formal definitions of the individual concepts.

First, the specification of a *modeling frame* (or simply: *frame*) is provided.

Definition 1 (Modeling Frame)**.** Assuming a model m of a system which is surrounded by an environment, and an activity act that is a modeling activity performed on m, we define a *modeling frame* to consist of all process information required to perform the activity, the context the activity is performed in, and a (potentially empty) set of frames defining sub-activities:

$$Frame = \langle Activity, Context, Frame^* \rangle$$

Definition 2 (Activity)**.** Given a frame F, we define an activity act as follows:

$$Activity = \langle Inputs, Outputs, Process \rangle$$

$Activity$ consists of the detailed information about *Inputs*, *Outputs* and the *Process* itself. The $Process$ defines the order in which the individual tasks of the activity are performed in and the coordination of the tasks inputs and outputs.

Definition 3 (Context)**.** The context formalises the environment an activity is performed in. It captures goals, expectations and restrictions in the form of *Objectives*, *Assumptions* and *Constraints*.

$$Context = \langle Objectives, Assumptions, Constraints \rangle$$

Objectives: represent the set of properties that should be satisfied after the activity execution.

Assumptions: is the set of preconditions in which the activity will lead to the objectives being satisfied.

Constraints: is the set of properties that have to be upheld during the execution.

Definition 4 ($Frame^*$)**.** The set of frames declared within a frame represents the sub-frames that define the frame activity's type. If the set of frames is empty, then the activity is *atomic*, meaning there are no finer-grained sub-tasks. Otherwise the activity is an *orchestrator*, implying that the activity's process defines the order in which the sub-activities are executed and which data is passed between them.

Formally, the type of the activity is defined as follows:

$$type(Activity) = \begin{cases} atomic, & \text{iff } Frame^* = \{\} \\ orchestrator, & \text{otherwise} \end{cases}$$

It is to be mentioned that virtually all activities can be divided into more finer-grained sub-processes and it is not possible to provide a general definition of a "good" granularity. Factors that influence this choice include the activity executor's domain familiarity, the complexity of the task, etc. We suggest to choose a granularity that is detailed enough to be non-ambiguous, but does not contain redundant (obvious) information.

B. Frame Integrity Constraints

As defined, a frame's objectives are captured within the context together with assumptions and constraints. Hence, the objectives' fulfilment depends on a frame's activity, assumptions and constraints.

Definition 5 (Frame Objective Fulfilments)**.** Let F be a frame consisting of an activity act and a context $\langle O, A, C \rangle$. The

execution of the activity with the right assumptions and good constraints leads to the fulfilment of the frame's objectives.

$$\bigwedge F.A \wedge \bigwedge F.C \wedge F.act \Rightarrow \bigwedge F.O$$

where $F.act = True$ iff the activity is completed.

This also implies that if the frame objectives are not satisfied, then the frame definition itself has to be changed (i.e. the activity and/or the context).

In fact, constraints can be seen as refinements of activity objectives and a constraint's satisfaction as an activity's objective. Hence, we define a constraint to be a refinement of an objective.

Definition 6 (Objective refinement)**.** Let F be a frame consisting of an activity act and a context $\langle O, A, C\rangle$. Let $F.c$ be a constraint in the set of constraints C ($c \in C$) and $F.o$ an objective in the set of objectives O ($o \in O$). We define c to be a refinement of a more coarse objective o.

$$F.c \leq F.o$$

where $x \leq y$ denotes that x is weaker than or equal to y.

This further leads to the conclusion that an activity has to satisfy all constraints, otherwise an unsatisfied constraint c implies a failed objective o and the set of objectives O being non-fulfilled.

$$\neg F.c \Rightarrow \neg F.o \Rightarrow \neg \bigwedge F.O$$

One such an objective refinement is presented in the TL case study. The objective of the modeling frame is *to learn about the traffic light timing*. This objective leads to the constraint: *the measuring device has to have ample precision*. Failing the constraint implies failure of the objective.

A satisfaction of a constraint c through an activity act is defined as follows:

Definition 7 (Constraint satisfaction)**.** Given a frame F consisting of an activity act and a context $\langle O, A, C\rangle$, act has to satisfy each constraint $c \in C$.

$$\forall c \in C : F.act \vdash F.c$$

where $\vdash$ denotes satisfaction.

C. Parent - Child Frame Relationship

We demand consistency between parent and child frames. Depending on whether composition or decomposition is performed, a different view points might be useful. The first one is of particular interest for the top-down refinement of activities. Such frame decomposition, where one frame's (the *parent's*) process is replaced by several sub-frames (*child frames*). We demand that a parent frame's context $\langle O, A, C\rangle$ has to be reflected within the the child frames.

There is a direct link between the contexts of the child frames and their parent frames, namely that the concepts have to be reflected within the sub-frames. This means that a frame's assumptions need to remain valid within the child frames.

Definition 8 (Top-Down Frame Refinement)**.** Let F be a frame with activity act and context $\langle O, A, C\rangle$. The process act is split into several steps, defined by the frames $F'_1, \ldots, F'_n$. We demand that that for each child frame all assumptions are consistent with the parent frame assumptions. We require the same for constraints.

$$\forall F'_i : \bigwedge F.A \Rightarrow \bigwedge F'_i.A$$

$$\forall F'_i : \bigwedge F.C \Rightarrow \bigwedge F'_i.C$$

For the bottom-up composition another viewpoint might be more applicable. Since the new high-level activity will be composed of several existing frames, the assumptions and constraints of the new activity's context are a union of the child frame's assumptions and constraints.

Definition 9 (Bottom-Up Frame Composition)**.** Let F be a frame with activity act and context $\langle O, A, C\rangle$ to be composed of several frames $F'_1, \ldots, F'_n$. We require that the assumptions and constraints of F are the union of the constraints of the subframes F'_i ($i \in \{1, \ldots, n\}$)

$$F.A = \left(\bigcup_{i=1}^{n} F'_i.A\right), \quad F.C = \left(\bigcup_{i=1}^{n} F'_i.C\right)$$

In case an assumption/constraint is defined in multiple subframes, the weakest assumption/constraint is chosen for the parent frame.

There exist several other composition and consistency relationships between frames (e.g. the sequential composition). Their description would however, exceed the bounds of this publication and is therefore left out.

VI. Related Work

The idea of framing a model with extra information about its context has been proposed by Zeigler [1]. He defined an experimental frame as the conditions under which the system is observed and experimented with. Zeigler defines an operational view on this experimental frame with three components: (a) a generator that models that allowable system inputs to the model, (b) a transducer to observe and analyse the output of the model and (c) the acceptor to decide on the validity of the model. Rozenblit created an implementation of the experimental frame in [3]. His implementation mends some of the gaps that the theoretical framework left open, such as initial conditions. Zeigler's approach has been used in several works by Ören, where he studies the acceptability of simulations [4]. Muzy and Traore extend the work of Zeigler by formalising the frame and providing a specification hierarchy [5]. Finally, our work is a generalisation of the work presented in [2] to different phases in the engineering or scientific process.

Re-use of components in (component-based) software engineering is closely related to our contribution. One such contribution for re-use is contract-based design. Contracts for software components by Meyer [6] structure the properties of a component into a set of pre- and post-conditions, similar to

frame assumptions and constraints. The Eiffel programming language for example has Require and Ensure clauses to express these conditions. Contract-based approaches for cyber-physical systems typically use the terminology of assumptions and guarantees. Under the assumed conditions, the component promises to fulfil the guaranteed properties. Damm et al. [7] introduce the notion of Rich Components. Rich components extend model components such that: (a) they cover all the specifications of the involved viewpoints, (b) they contain a set of assumptions and guarantees with respect to the context of the component, and (c) they provide classifiers to the assumptions. Heterogeneous rich components extend these rich components to support the integration of heterogeneous viewpoints on a system with different semantics originating from multiple design layers and tools [8]. Benveniste et al. describe the mathematical foundations of three contract operators: refinement, composition and conjunction [9].

VII. Conclusion and Future Work

In this paper we introduce a structured framework for the definition of modeling activities and their execution contexts, called frames. These frames not only specify the process that has to be applied, but also clearly capture the activity's objectives, assumptions and constraints. Such a vigorous definition facilitates the reproducibility and modeling activities and validation, re-use and composition of the model itself.

The paper identifies several generic parametrisable frame types. These frames can be used for various modeling situations which require clear context definitions, in particular model validation and verification.

In future we plan to extend our work into several directions. First, we aim to extend our research of common basic frames, frame types and the formalisations of the frame relations. Second, we intend to investigate how frames can support the development of safety cases and facilitate evidence reuse as part of a safety certification process. Next, we aim to facilitate the usage of frames by providing a domain-specific language and tool-support for the specification of frames, to support the automatic verification and validation of context and process relations. Lastly, we want to apply the framework on an industrial use case in order to verify its usability. This would further give us the opportunity to discover best practices and frame patterns which facilitate frame usability.

Acknowledgements: We thank the organizers of the Computer Automated Multi-Paradigm Modelling (CAMPAM) 2017 workshop for providing a place where this research started. We also express our gratitude to Mamadou Traoré for his inputs and insights during CAMPAM 2017.

References

[1] B. P. Zeigler, *Multifaceted Modelling and Discrete Event Simulation*. London: Academic Press, 1984.

[2] J. Denil, S. Klikovits, P. J. Mosterman, A. Vallecillo, and H. Vangheluwe, "The experiment model and validity frame in M&S," *Proceedings of the Symposium on Theory of Modelling and Simulation*, 2017.

[3] J. W. Rozenblit, "Exp—a software tool for experimental frame specification in discrete event modelling and simulation," in *Proceedings of the Summer Computer Simulation Conference*, 1984, pp. 967–971.

[4] T. I. Ören, "Concepts and criteria to assess acceptability of simulation studies: A frame of reference," *Communications of the ACM*, vol. 24, no. 4, pp. 180–189, 1981.

[5] M. K. Traoré and A. Muzy, "Capturing the dual relationship between simulation models and their context," *Simulation Modelling Practice and Theory*, vol. 14, no. 2, pp. 126–142, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.simpat.2005.03.002

[6] B. Meyer, "Applying'design by contract'," *Computer*, vol. 25, no. 10, pp. 40–51, 1992.

[7] W. Damm, A. Votintseva, A. Metzner, B. Josko, T. Peikenkamp, and E. Böde, "Boosting re-use of embedded automotive applications through rich components," *Proceedings of Foundations of Interface Technologies*, vol. 2005, 2005.

[8] L. Benvenuti, A. Ferrari, L. Mangeruca, E. Mazzi, R. Passerone, and C. Sofronis, "A contract-based formalism for the specification of heterogeneous systems," in *Specification, Verification and Design Languages, 2008. FDL 2008. Forum on*. IEEE, 2008, pp. 142–147.

[9] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen, "Contracts for System Design," INRIA, Research Report RR-8147, Nov. 2012. [Online]. Available: https://hal.inria.fr/hal-00757488