Validity Frames for Self-Adaptive Systems

Raheleh Biglari[1][a], Joachim Denil[1][b] and Claudio Gomes[2][c]
[1]*Flanders Make@UAntwerpen & University of Antwerp, Antwerp, Belgium*
[2]*Aarhus University, Aarhus, Denmark*
{*raheleh.biglari, joachim.denil*}*@uantwerp.be, claudio.gomes@ece.au.dk*

Keywords: Modelling and Simulation, Validity Frame, Software Contract, Composition, Self-adaptive System, Optimisation

Abstract: Validity Frames are a key concept in modelling and simulation, encoding the contexts in which models provide valid results. This position paper explores how Validity Frames propagate through optimisation processes and affect self-adaptive systems. We demonstrate that optimisers inherit constraints from model Validity Frames and generate their own validity conditions for the configurations they produce. These propagated Validity Frames can be formalised as software contracts with explicit assumptions and guarantees. Using a semi-active suspension system as a running example, we show how Validity Frames constrain design space exploration, enable runtime monitoring of validity boundaries, and support informed adaptation strategies. Our approach reveals a fundamental tradeoff between optimality and applicability: configurations optimised for narrow scenarios may achieve peak performance but fail when conditions change, while configurations evaluated across broader Validity Frames sacrifice peak performance for robustness. By explicitly tracking Validity Frames throughout the optimisation process, self-adaptive systems can reason about when adaptation is needed and select configurations appropriate to their current operational context. The full formalisation of validity frames into contracts and their composition, which allows for the precise integration of assumptions and guarantees across models, optimisers, and adaptive components, will be our main future work path.

1 INTRODUCTION

Validity Frame is a key concept in modelling and simulation. Validiy frames explicitly encode the contexts in which a model provides valid results for specific properties relative to a real-world system (Van Mierlo et al., 2020; Denil et al., 2017). It extends the foundational concept of the experimental frame, introduced by Zeigler, which represents a set of conditions under which a system is observed or experimented with by real or virtual experiments (Zeigler, 1984). By clearly defining the validity frame, and particularly by formalising it to a software contract (Benveniste et al., 2007), modelers can ensure that their models are used appropriately and within their domain of validity.

Models have numerous applications, offering visualisation, decision support, and in particular, optimisation. In this paper, models are used for optimisation where it is part of the cost function to evaluate how well a particular model configuration performs (a.k.a., design space exploration).

Since models can be imbued with contracts, it is natural to ask *what the consequences are of using those contracts in optimisation applications*. Specifically, when models with their associated validity frames are used within optimisation loops, it is unclear how these frames influence and constrain the validity of the resulting solutions. This gap becomes particularly critical in self-adaptive systems, where both the optimisation process and its outputs must maintain clear boundaries of validity to ensure safe and effective adaptation.

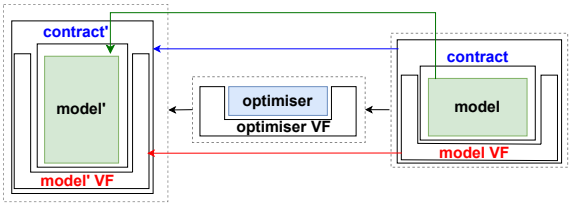

Figure 1: Conceptual overview of validity frame propagation.

In this paper, we argue that an optimiser should

[a] https://orcid.org/0000-0003-2042-1868
[b] https://orcid.org/0000-0002-4926-6737
[c] https://orcid.org/0000-0003-2692-9742

not only produce an optimal solution but also articulate the validity frame within which that solution holds. In other words, the optimiser itself comes with its own validity frame, one that governs the applicability of its output. Recognising and formalising this additional layer of validity is essential for building trustworthy adaptive systems.

Figure 1 provides a visual summary of this conceptual flow, showing how validity frames propagate from models to optimisation processes, how they constrain the configurations explored and the scenarios evaluated, and how they ultimately shape the validity of the resulting solutions. This diagram serves as a high-level guide to the ideas developed in the paper.

In sum, we address the following research questions: **RQ1:** How do validity frames propagate from models into optimisation processes, and how do they constrain the validity of the optimised configurations? **RQ2:** How can these propagated validity frames be formalised as software contracts to support runtime reasoning and adaptation in self-adaptive systems?

We also show that by explicitly producing validity frames, self-adaptive systems (Weyns, 2020) can support flexible adaptation strategies. This enables the selection of adaptation approaches, whether favoring sharper, more precise solutions or broader, more general ones, based on the current operational context.

We adopt a mixed formal and empirical methodology to define and implement how validity frames propagate through optimisation loops. To demonstrate our approach, we present a running example of semi-active suspension system, evaluated in a car encountering a bump. The disturbance may push the system outside the scenarios initially considered, which in turn triggers an adaptation that involves optimisation.

In the remainder of this paper, Section 2 provides background on validity frames and software contracts, situating our work within the broader modelling and self-adaptive systems literature. Section 3 presents our approach, illustrated through a running example of a semi-active suspension system. Section 4 discusses the implications of validity frame propagation and contract composition, and Section 5 concludes.

2 BACKGROUND & RELATED WORK

Software contracts have long been used to formalise the expected behavior of components and systems. Meyer (Meyer, 2002) introduced the concept of Design by Contract (DbC) to improve software reliability by making assumptions and guarantees explicit. In DbC, each software component is governed by a contract consisting of preconditions, postconditions, and invariants. These elements define what a component expects from its environment, what it promises in return, and what must always hold true during execution. This approach enables modular reasoning and correctness by construction, laying a foundation for formal verification and runtime verification (Baier and Katoen, 2008; Bartocci et al., 2018), where these contracts are either proven to hold true offline, or monitored at runtime, respectively.

Building on this foundation, Benveniste et al. (Benveniste et al., 2007) extended contract theory to system-level design, particularly in the context of embedded and cyber-physical systems. Their work introduced assume/guarantee contracts, where each component assumes certain behaviors from its environment and guarantees its own behavior in response. Crucially, Benveniste formalised contract composition, allowing contracts to be combined, refined, and compared across hierarchical system architectures. This enables scalable and modular verification, ensuring that system-level properties are preserved when integrating multiple components.

In simulation-based modelling, the concept of an experimental frame defines the conditions under which a model is valid and applicable. We generalise this idea into validiy frames, which specify the domain of applicability for both models and their outputs.

Design space exploration (DSE) is the systematic process of evaluating and comparing many possible design alternatives in order to understand how different choices affect system performance, cost, reliability, or other key criteria, i.e., their fitness (Kang et al., 2011). Models play a central role in this process as their simulations provide the input to computing the fitness of the model parameterisation which corresponds to the system configuration. By automating the evaluation of many simulated design variations, DSE enables informed decision-making, and in particular enable self-adaptive systems.

Self-adaptive systems can monitor their own behavior and operating environment and modify their configuration or behavior in response to changes, without requiring manual intervention. They are designed to achieve certain goals, such as performance, reliability, or energy efficiency, even when conditions vary unpredictably at runtime. To support this, self-adaptive systems often rely on models that capture both how the system works and how it should respond to different situations. These models are used in DSE, enabling the system to select appropriate adaptations.

As the next section shows, the validiy frame of the model interacts with both the optimisation loop it is being used in, as well as the resulting optimal system configuration, thus affecting the behavior of self-adaptive systems.

3 APPROACH

In this section, we first describe our proposed contribution, visually summarised in Figure 2, and later illustrate it with a running example.

3.1 Validity Frames for Self Adaptive Systems

Validity Frames of Models. We assume that the system under study contains a model with its own *validity frame*. The model reproduces the system properties of interest, while the validity frame determines whether a virtual experiment (i.e., simulation) can be meaningfully carried out with a given model instantiation, defined by specific parameters and initial conditions.

A validity frame can be implemented in different ways. Conceptually, it is a software component that, given the experimental conditions, outputs a measure of how much the model can be trusted to reproduce the system phenomena under those conditions and for a model instantiation. One way is a *Boolean function* that returns `true` if the experiment lies within the model's applicability, and `false` otherwise. Second way is a *quantitative function* that quantifies the degree of trust in the model results. For instance, (Biglari et al., 2025) proposed using the distance between a new experiment and those used to validate the model as a quantitative validity measure.

In this paper, we adopt a quantitative measure of validity, assessing how trustworthy a model instantiation is for a given experiment.

Building Validity Frames. A straightforward way to construct a validity frame is to collect validation experiments for a set of model instantiations, and define a similarity measure between any new experiment and these reference experiments. If the similarity is high, the new experiment is considered valid.

Validity Frames as Runtime Monitors. If this similarity (or distance) measure can be computed efficiently, validity frames can serve as runtime monitors, continuously checking whether the model is being applied within its domain of validity. A practical implementation is an auto encoder neural network, where the reconstruction error acts as the similarity measure.

For example, in a design space exploration scenario where a model is instantiated with different parameters and evaluated across multiple virtual experiments. Ideally, these experiments should fall within the model's validity frame, see (1) and (2) in Figure 2. However, because parameterisations can shift the range of valid experiments, the validity frame must still be checked at runtime to determine which experiments remain meaningful for each instantiation.

Validity Frames for Self-Adaptation. An important consequence of this mechanism is that an optimised set of model parameters defines its own subset of acceptable virtual experiments. When such an optimised model is deployed to reconfigure a system, as in self-adaptive systems, it is natural for the system to continue using the model's validity frame to ensure that post-reconfiguration operation remains within acceptable environmental conditions, see (4) and (5) in Figure 2.

To achieve this, the system's environment must be translated into virtual experiments that can be evaluated by the validity frame. This translation is typically straightforward, as it mirrors the process used during model construction and validation.

If the operating environment falls outside the model's validity frame, this indicates that such conditions were not considered in the original design space exploration. Detecting this discrepancy naturally triggers re-optimisation and further self-adaptation, as illustrated by "Adaptation Trigger" arrow in Figure 2.

Validity Frame Trade-offs. When a model's validity frame is constrained by the virtual experiments used to assess model fitness during design space exploration, it becomes possible to trade off model optimality against the boadness of its validity frame—(3) in Figure 2. A configuration tuned for peak performance in a narrow set of scenarios may fail when conditions change, whereas a configuration optimised across a broader validity frame may sacrifice peak performance but remain valid in more diverse environments. Thus, validity frames introduce a new dimension to optimisation: balancing sharp optimality against broader applicability.

3.2 Application to Running Example

To demonstrate our approach, we consider a semi-active car suspension system. Such systems improve

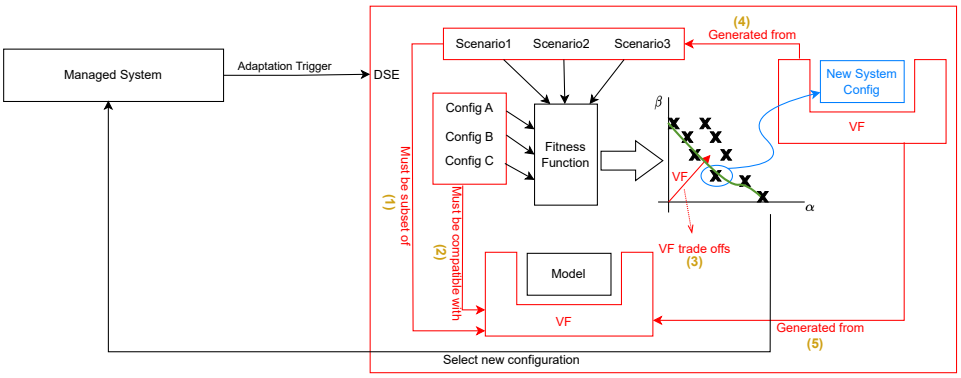

Figure 2: Summary of our contributions (numbered in parenthesis): validiy frame of models affects the possible choices of configurations to try out (1), as well as simulation scenarios to evaluate each configuration on (2). The use of validiy frame adds a new dimension to the tradeoff of an optimal solution, since one may opt for a wider or narrows validiy frame (3), and the validiy frame of the solution is generated from the scenarios used (4) as well the validiy frame of the model (5).

ride comfort and stability by adjusting shock absorber parameters in response to road conditions.

Figure 3 shows a quarter-car model, a widely used model in automotive engineering for suspension analysis and design (Khajavi and Abdollahi, 2007; Roukieh and Titli, 1993). The model consists of a sprung mass supported by a spring with stiffness of k and damper with coefficiency of c in parallel. When the vehicle encounters a disturbance, the spring provides a restoring force proportional to displacement, while the damper dissipates energy and controls oscillations. To simplify the example, the controller here can only adjust the stiffness parameter of the suspension in order to better handle bumps found at different speeds or change the riding conditions to be more comfort oriented (minimises vibrations transmitted to the driver) or performance oriented (Minimises peak oscillations to maintain stability).

Figure 4 shows the system response when encountering a road bump. The bottom graph depicts the bump profile, while the top graph shows the resulting vertical displacement of the car mass. When the wheel hits the bump, the system exhibits a characteristic spring-mass-damper response, an initial upward movement followed by damped oscillations. This directly impacts both passenger comfort and vehicle stability.

Validity Frame. For this example, the validity frame is defined as a distance measure to a set of experiments conducted at different speeds and different suspension stiffnesses. While we do not do that here, these experiments would be performed in a lab test bench with a real suspension prototype (Konieczny et al., 2013; Salah, 2017), by which the model is validated.

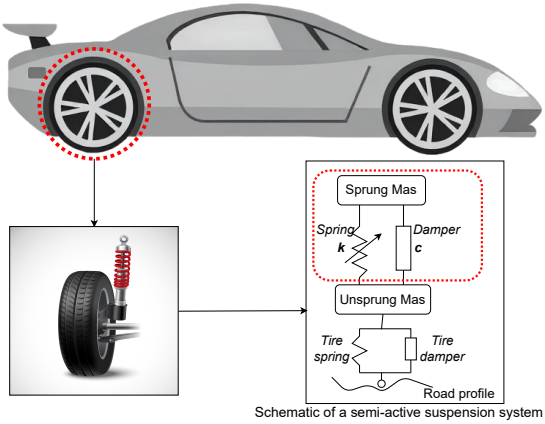

Figure 3: Quarter-car model of the semi-active suspension system. The red dotted box indicates the subsystem that is modelled in this study: the sprung mass supported by a spring and damper in parallel, interacting with the tire spring–damper pair and the road profile.

Validity frame of system configuration. Now assume the particular car in which this suspension system is deployed is being drived by a driver that commutes through city streets, rarely exceeding 45 m/s. Their driving profile is calm, predictable, and comfort-oriented. This driving profile translates to simulation experiments where the car drives through bumps at speeds lower than 45m/s.

In a design space exploration that focuses on these restricted scenarios, the data consistently points to a spring stiffness of 12000 N/m as the optimal choice, shown in Figure 5 and in the heatmap of Figure 7. This configuration minimises oscillations, absorbs disturbances smoothly, and delivers a comfortable ride.

Therefore, this self adaptive system is reconfig-

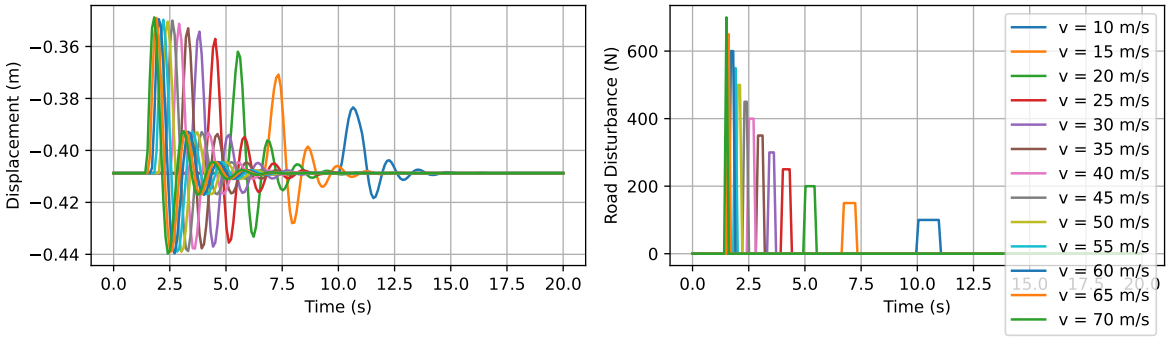

Figure 4: System response to a road bump. The right plot shows the bump profile, while the left plot depicts the vertical displacement of the car mass.

ured to set the spring stiffness to 12000 N/m, and its validity frame is restricted to experiments at speeds below 25 m/s.

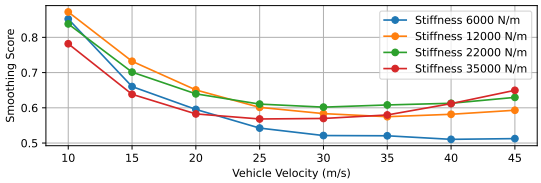

Figure 5: Smoothing score of the suspension system at low speeds for different spring stiffness values.

Example validity frame violation. Now a second driver starts using the car. This one prefers high speeds, pushing the vehicle to speeds well above 45 m/s. This is detected by the validity frame of current system configuration and triggers self adaptation. In this self adaptation we rerun the design space exploration, this time focusing on the virtual experiments at high speeds (reflecting most recent usage profile of the car). Now, the previous configuration, once ideal, begins to falter. The suspension struggles to settle quickly, and peak displacements become more pronounced. At these higher speeds, stability and control take precedence. In this new situation, a stiffer spring, 35000 N/m, as shown in Figure 6 emerges as the superior choice, offering faster settling times and better handling of rapid disturbances. This can also be seen in the heat map of Figure 7.

The optimiser then finds a new optimal for the spring stiffness (k) to achieve optimal suspension performance. Its objective function balances three key metrics: minimising vehicle displacement to enhance comfort, minimising the number of oscillations, and minimising settling time for better stability.

The previous example illustrates the relationships between the scenarios chosen to validate each model

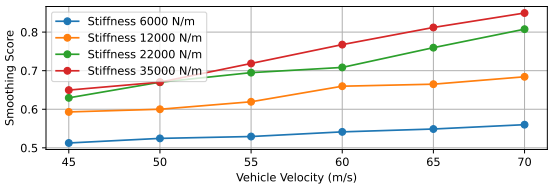

Figure 6: Smoothing score of the suspension system at high speeds.

association as part of the designed space exploration and the validity frame of the model. The model itself has a very broad validity frame containing all different speeds as well as all possible different stiffnesses. However the choices made as to which scenarios are given focus in the design space exploration effectively restrict the validity frame of the new system configuration which ends up being a subset of the original model validity frame.

The previous example also shows that two model configurations chosen for deployment can have completely different validity frames that have no intersection (i.e., Experiments at less than 45 meters per second and experiments at more than or equal to 45m/s). In the following we now consider an example where the design space exploration is considering the full range of virtual experiments that are covered by the validity frame of the model.

Tradeoffs. Now consider an alternative scenario in which instead of the optimiser considering only the most recent usage patterns of the car it considers the full history of usage of the car: this means that the car has been driven at both low speeds as well as high speeds. In the design space exploration the new optimal stiffness that seems to perform satisfactorily across the full range of scenarios is now 22000 N/m, showing the tradeoff. This new system configuration carries the broadest validity frame since it is being evaluated on the broadest set of virtual experiments.

This example illustrates how validity frames evolve with context: what is optimal in one scenario may be invalid in another. Recognising these shifts enables self-adaptation. The suspension system that perfectly serves the comfort-oriented city driver becomes unsuitable for high-speed conditions. By recognising these distinct validity frames, one prioritising comfort, the other emphasising stability, and one focused on tolerating the broadest set of scenarios, we establish the foundation for runtime monitoring and self-adaptation.

Practical implementation considerations. To implement this adaptation, we introduce an optimiser that continuously monitors driving patterns and adjusts the suspension parameters accordingly. The optimiser operates within the bounds of the current validity frame while being prepared to shift between validity frames as conditions change. Figure 7 shows that for low speed scenarios (10-25 m/s), optimal spring stiffness values cluster around 12000-15000 N/m, forming one distinct validity frame. As speed increases beyond 45 m/s, the optimal values shift dramatically toward 35000-40000 N/m, establishing a separate high-speed validity frame. These clear boundaries help the optimiser make informed decisions about when and how to adapt the suspension system.

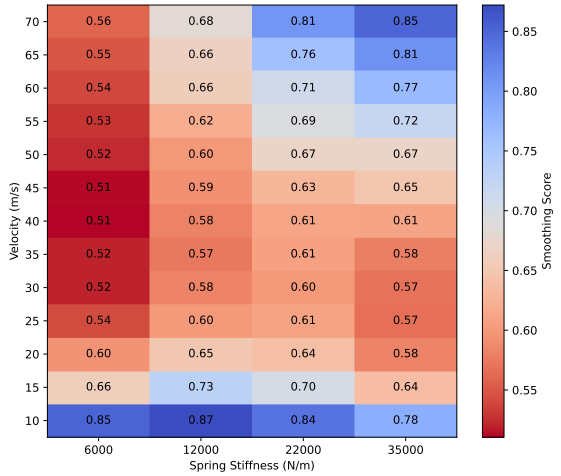

Figure 7: Heatmap of smoothing scores across different vehicle velocities and spring stiffness values. Low-speed scenarios favour softer springs optimised for comfort, while high-speed scenarios require stiffer springs to ensure stability. An intermediate stiffness around 22,000 N/m provides a solution valid across the broadest range of conditions.

This dynamic approach shows how validity frames serve not just as constraints but as guides for adaptation. The system maintains optimal performance across varying conditions by explicitly acknowledging and switching between different operational contexts, each with its own well-defined validity frame.

4 ASSUMPTIONS & DISCUSSION

In Section 3, we have shown, using the example, that using the model for doing optimisation, propagates the assumptions and constraints of the model upwards in the process or component hierarchy. Each configuration derived from the optimiser inherits assumptions from the model's validity frame and contributes its own guarantees, such as smoothing score thresholds. This validity frame can be used to monitor runtime validity and trigger adaptation when assumptions no longer hold.

Assumptions. We use a simple example to illustrate a general approach. In practice the validity frame of the model will be much larger than what we illustrate here, and comprises many other dimensions such as the numerical simulation parameters to be used.

The example chosen can be made non-adaptive by simply choosing the spring stiffness that performs acceptably across the whole range of experiments. This would then by an example of a traditional design of a suspension system.

Validity Frames as Software Contracts. Since models are software, validity frames can be formulated as part of the model's contract: they formalise what the model assumes about its operating context and what it guarantees in return. For instance, Biglari et al. (Biglari et al., 2025) show that validity frames can be transformed into runtime monitors that act as contracts, preventing the use of a surrogate model in situations where its inputs do not belong to the same distribution as the dataset it was trained on. Validity frames capture assumptions about the environment, inputs, and expected behaviors, and they evolve as systems adapt to new conditions. Braun et al. (Braun et al., 2024) formalise validity domains, closely related to validity frames, as assume-guarantee design contracts for sufficiently valid simulation setups, while Klikovits et al. (Klikovits et al., 2017) explicitly relate frame assumptions and constraints to Meyer-style pre- and postconditions.

In our example, we can distinguish several scenarios of the components that will be created with the optimiser: (a) an online optimisation component that will do the optimisation (and run the simulations) during operation, (b) based on the results of the optimi-

sation, the designer will select a value for the suspenstion component or, (c) an adaptive component that combines three specific controllers for the suspension, that need to be switched once a certain input speed has been exceeded.

In the first case, we generate a real component, not a derived component from the model and the optimiser. As such, we have the validity frame of the model where the *Assume* part of the contract relates very well to the constraints and influence factors of the validity frame. The *Guarantees* capture the outcome of the model, relating to what the model can or cannot output (operational domain), but also it has to indicate how well that the model does this, e.g., what is the accuracy of the solution. We also have the optimiser, that adds its own assumptions to the component, e.g., solver accuracy of the optimiser, convergence requirements, or discretisation constraints. When using the model within the optimiser as a component, we thus have to compose contracts (Benveniste et al., 2007). A crucial property emerging from this composition is robustness. This robustness is not merely inherited from the scenarios admissible under the model's validity frame; it also arises from the optimiser itself. The solver may intentionally avoid the sharpest or most extreme optimum in favour of configurations that are more adaptable, less sensitive to uncertainty, or more stable across neighbouring scenarios. Consequently, robustness becomes a contractual property resulting from the combined influence of (i) the admissible scenario space defined by the model's validity frame and (ii) the optimiser's own strategy, numerical behaviour, and trade-offs.

The second case and the third case are different as the process does not result in a component that will be used at run-time, it is just an analysis tool to find the correct parameter values of the suspension system. In the second case, the suspension system will inherit the contract partially from the analysis done. A single value will be used, and as such, the smoothness score will be based on this. Finally, in the third case a derived component is created, together with three independant components. Here the component still inherits a lot of information from the validity frame of the model, as the admissable input of the component is determined from the validity frame of the model. If you were to use the component outside of this range, no guarantees can be given on the performance of the suspension.

Let's take a look at the contracts generated in the third case, shown in Table 1. We define three precomputed controllers, each valid for a specific operational sub-range of the vehicle speed. Each controller inherits its *assumptions* from the validity frame of the suspension model, together with the optimisation process that adds its own *guarantees* on ride comfort, expressed as a smoothing score.

Composition of these contracts at the system level yields the global assume-guarantee specification:

$$\mathcal{C}_{\text{sys}}(v) = \begin{cases} (A_{C_1}, G_{C_1}), & \text{if } v \in [0,30] \text{ m/s}, \\ (A_{C_2}, G_{C_2}), & \text{if } v \in (30,50] \text{ m/s}, \\ (A_{C_3}, G_{C_3}), & \text{if } v \in (50,70] \text{ m/s}. \end{cases}$$

Equivalently, the system satisfies

$$\forall i \in \{1,2,3\}, \quad A_{C_i} \Rightarrow G_{C_i},$$

and an adaptation mechanism ensures that one and only one assumption A_{C_i} holds at any time, activating the corresponding controller.

Overall, this shows how validity frames operationalise the abstract assume–guarantee notion with simulation models. They provide a concrete structure for expressing assumptions and guarantees that can be automatically monitored, composed, or refined. This alignment of validity frames and contracts enables not only the traceability of model validity through optimisation as the validity frame keeps all the (meta-)data related to the admissable inputs, constraints, the processes for establising validation or parameter estimation, and the provinance of the simulation model. While contracts allow us to reason over the component within a design process or at run-time. Furthermore, is also allows systematic management of adaptation: when assumptions break, the system can reason about which contract (and corresponding component) remains valid.

5 CONCLUSION

In this paper we have shown yet another benefit of permeating models with their validity frames beyond being able to trust their behavior: it allows permeating solutions to optimization processes using those models with their own validity frames, which in turn enable more effective decision making in self-adaptive systems. The validity frame of deployed active system configurations can further be used to trigger new adaptations when its boundary is being approached. A semi-active suspension system was used to illustrate these concepts.

This work is inspired by the propagation of software contracts: when a component has a contract in a software system, that contract propagates both to the upstream components as post-conditions, as well as the downstream components as pre-conditions. We

Table 1: Assume–Guarantee contracts for pre-computed suspension controllers.

Controller	Assumptions (A)	Guarantees (G)	Notes
C_1: 12 kN/m	$v \in [0, 25)$ m/s	Smoothing Score $\geq 0.65 \pm 0.05$	Comfort-oriented (urban speeds)
C_2: 22 kN/m	$v \in [25, 40]$ m/s	Smoothing Score $\geq 0.60 \pm 0.05$	Transitional regime (mixed conditions)
C_3: 35 kN/m	$v \in (40, 70]$ m/s	Smoothing Score $\geq 0.65 \pm 0.05$	Stability-oriented (highway speeds)

highlight also the need for dynamic contracts in self-adaptive systems, where the solution produced bring its own contract, or changes the managed system contract. To the best of our knowledge, the theoretical work in the propagation of these dynamic contracts is in its infancy, as also argued by in the RoboSAPIENS project (Larsen et al., 2024).

Looking ahead, the full formalisation of validity frames into contracts and their composition remains future work, alongside exploring other applications of models beyond optimisation, such as anomaly detection, where validity frames may help distinguish true anomalies (where the system is operating within the model validity frame but the model predictions are still triggering an anomaly) from potentially false ones (where the system is operating outside of the model validity frame).

ACKNOWLEDGEMENTS

The work presented here is partially supported by the RoboSAPIENS project funded by the European Commission's Horizon Europe programme under grant agreement number 101133807. We used an LLM only to improve the grammar and clarity of the manuscript. All content was reviewed and approved by the authors.

REFERENCES

Baier, C. and Katoen, J.-P. (2008). *Principles of model checking*. MIT press.

Bartocci, E., Falcone, Y., Francalanza, A., and Reger, G. (2018). Introduction to runtime verification. In *Lectures on Runtime Verification: Introductory and Advanced Topics*, pages 1–33. Springer.

Benveniste, A., Caillaud, B., and Passerone, R. (2007). A generic model of contracts for embedded systems. *arXiv preprint arXiv:0706.1456*.

Biglari, R., Gomes, C., and Denil, J. (2025). Towards a validity frame of multi-modal surrogate models for traffic simulation. In *Annual Modeling and Simulation Conference (ANNSIM)*, pages 1–13, Madrid, Spain.

Braun, N., Steimle, M., Torngren, M., and Maurer, M. (2024). A concept for semi-automatic configuration of sufficiently valid simulation setups for automated driving systems. In *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2042–2049.

Denil, J., Klikovits, S., Mosterman, P. J., Vallecillo, A., and Vangheluwe, H. (2017). The experiment model and validity frame in m&s. In *Proceedings of the Symposium on Theory of Modeling & Simulation*, TMS/DEVS '17, San Diego, CA, USA. Society for Computer Simulation International.

Kang, E., Jackson, E., and Schulte, W. (2011). An Approach for Effective Design Space Exploration. In Calinescu, R. and Jackson, E., editors, *Monterey Workshop*, pages 33–54. Springer Berlin Heidelberg, Berlin, Heidelberg.

Khajavi, M. N. and Abdollahi, V. (2007). Comparison between optimized passive vehicle suspension system and semi active fuzzy logic controlled suspension system regarding ride and handling. In *Proceedings of world academy of science, engineering and technology*, volume 21, pages 57–61.

Klikovits, S., Denil, J., Muzy, A., and Salay, R. (2017). Modeling frames. In *CEUR workshop proceedings*, pages 315–320.

Konieczny, J., Kowal, J., Raczka, W., and Sibielak, M. (2013). Bench tests of slow and full active suspensions in terms of energy consumption. *Journal of Low Frequency Noise, Vibration and Active Control*, 32(1-2):81–98.

Larsen, P. G., Ali, S., Behrens, R., Cavalcanti, A., Gomes, C., Li, G., De Meulenaere, P., Olsen, M. L., Passalis, N., Peyrucain, T., Tapia, J., Tefas, A., and Zhang, H. (2024). Robotic safe adaptation in unprecedented situations: The RoboSAPIENS project. *Research Directions: Cyber-Physical Systems*, 2:e4.

Meyer, B. (2002). Applying'design by contract'. *Computer*, 25(10):40–51.

Roukieh, S. and Titli, A. (1993). Design of active and semi-active automotive suspension using fuzzy logic. *IFAC Proceedings Volumes*, 26(2):73–77.

Salah, M. (2017). A laboratory automotive suspension test rig: Design, implementation and integration. *Jordan Journal of Mechanical & Industrial Engineering*, 11(2).

Van Mierlo, S., Oakes, B. J., Van Acker, B., Eslampanah, R., Denil, J., and Vangheluwe, H. (2020). Exploring validity frames in practice. In *International Conference on Systems Modelling and Management*, pages 131–148. Springer.

Weyns, D. (2020). *An introduction to self-adaptive systems: A contemporary software engineering perspective*. John Wiley & Sons.

Zeigler, B. (1984). *Multifacetted modelling and discrete event simulation*. Academic Press Professional, Inc.