# X-by-Construction Meets Runtime Verification

Maurice H. ter Beek[1(✉)], Loek Cleophas[2,3(✉)], Martin Leucker[4], and Ina Schaefer[5]

[1] ISTI–CNR, Pisa, Italy
m.terbeek@isti.cnr.it
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
l.g.w.a.cleophas@tue.nl
[3] Stellenbosch University, Stellenbosch, Republic of South Africa
[4] University of Lübeck, Lubeck, Germany
leucker@isp.uni-luebeck.de
[5] KIT, Karlsruhe, Germany
ina.schaefer@kit.edu

**Abstract.** In recent years, researchers have started to investigate X-by-Construction (XbC)—beyond correctness as considered by the more traditional Correctness-by-Construction (CbC) paradigm—as a refinement approach to engineer systems that by-construction satisfy certain non-functional properties—also, and in particular, in the setting of probabilistic systems and properties. In line with the need to join forces with concepts from runtime verification (RV), this track brings together researchers and practitioners working to share their views on the many possible synergies between CbC/XbC at design time and RV at runtime.

## 1 Motivation

*Correctness-by-Construction* (CbC) sees the development of software (systems) as a step-wise refinement process from specification to code, ideally by CbC design tools that automatically generate error-free software (system) implementations from rigorous and unambiguous requirement specifications. Afterwards, testing only serves to validate the CbC process rather than to find bugs.

A lot of progress has been made on CbC, and after a successful track on the combination of CbC with post-hoc verification at ISoLA 2016 [7], at ISoLA 2018 it was time to look further than correctness by investigating a move from CbC to *X-by-Construction* (XbC), i.e., by considering also non-functional properties [6]. XbC is thus concerned with a step-wise refinement process from specification to code that automatically generates software (system) implementations that by construction satisfy specific non-functional properties (i.e., concerning security, dependability, reliability, resource or energy consumption, and the like). In line with the growing attention to fault tolerance and the increasing use of machine-learning techniques in modern software systems, which make it hard to establish guaranteed properties [22], as witnessed in other tracks at ISoLA 2022 [16,20],


T. Margaria and B. Steffen (Eds.): ISoLA 2022, LNCS 13701, pp. 141–148, 2022.
https://doi.org/10.1007/978-3-031-19849-6_9

a third track in this series, at ISoLA 2020/2021, focussed on XbC in the setting of probabilistic systems and properties [5].

*Runtime verification* (RV) is concerned with monitoring and analysing actual software (and hardware) system behaviour [17]. RV is of paramount importance to system correctness, reliability, and robustness by providing an additional level of rigour and effectiveness when compared to testing, and improved practical applicability when compared to exhaustive formal verification. RV can be used prior to deployment, for testing, verification, and debugging purposes, as well as after deployment, for ensuring reliability, safety, and security—and for providing fault containment and recovery or online system repair, possibly paired with a digital twin acting as the virtual counterpart of the actual real-time (software and/or hardware) system behaviour [9].

## 2 Aim

Building on the highly successful ISoLA tracks mentioned above, the aim of this track is to bring together researchers and practitioners who are interested in CbC/XbC, and who acknowledge the need to join forces with concepts from RV. We believe this is important since (1) achieving correctness starting at design time is difficult—if not impossible—with the current proliferation of systems with data-driven AI components, while (2) a system failure detected by RV may possibly be repaired, but how can it be ensured that the corrected system is indeed better than before?

Given this specific topic, this ISoLA 2022 track fits perfectly as fourth track in the aforementioned series of ISoLA tracks:

| | |
|---|---|
| **ISoLA 2016** | Correctness-by-Construction and Post-hoc Verification: Friends or Foes? |
| **ISoLA 2018** | X-by-Construction |
| **ISoLA 2020/2021** | X-by-Construction: Correctness Meets Probability |
| **ISoLA 2022** | X-by-Construction Meets Runtime Verification |

We have therefore invited both researchers and practitioners working in the following communities to participate in this track and share their views on the many possible synergies between CbC/XbC at design time and RV at runtime:

- People working on system-of-systems, who address modelling and analysis (considering correctness, but also non-functional properties concerning security, reliability, resilience, energy consumption, performance, sustainability, and the like) of networks of interacting legacy and new software systems, and who are interested in applying CbC/XbC or RV techniques in this domain in order to provide guarantees for system-of-systems.
- People working on quantitative modelling and analysis, for instance through probabilistic/real-time systems and probabilistic/statistical model checking, in particular in the specific setting of dynamic, adaptive or (runtime) reconfigurable systems with variability. These people typically work on lifting successful formal methods and (runtime) verification tools from single systems to

families of systems, i.e., design and (runtime) verification techniques that need to cope with the complexity of systems stemming from behaviour, variability, and randomness—and which focus not only on correctness but also on non-functional properties concerning safety, security, performance, dependability, and the like. As such, they may be interested in applying CbC/XbC or RV techniques in this domain to provide proper guarantees for families of systems.
– People working on systems involving components that employ machine-learning (ML) or other artificial intelligence (AI) approaches. In these settings, models and behaviour are typically dependent on what is learned from large data sets, and may change dynamically based on yet more data being processed. As a result, guaranteeing properties (whether correctness or non-functional ones) becomes hard, and probabilistic reasoning needs to be applied instead with respect to such properties for the components employing AI approaches. As a consequence, people working in this domain may be interested in applying CbC/XbC or RV techniques to provide guarantess for such AI-based systems.

## 3 Contributions

We briefly describe the contributions of this track and group them thematically.

### 3.1 CbC: Robustness, Co-piloting, and Digital Twinning

In [18], Nayak et al. consider *correctness-by-construction* in the light of robustness. While an optimal strategy may meet the correctness criteria under the given model of the environment, small deviations of the environment's behaviour may render the strategy non-optimal or even incorrect. Especially, if the environment is non-antagonistic, strategies could be improved. Based on robust specifications in so-called robust Linear Temporal Logic (rLTL), the authors study the problem of monitoring the behaviour of the environment and adapting strategies accordingly to achieve *robustness-by-construction*.

In [3], Ahrendt et al. suggest a novel approach to development of safety-critical software, combining learning-based *co-piloting* with formal methods in order to assist agile, simultaneous development of (1) implementation, (2) specification, and (3) tests. The vision is that an IDE supporting this approach would assist users by suggesting necessary changes to the other elements if one of the three were changed. Controlled by the user, such changes could be used to reestablish consistency, possibly in an iterative fashion. The authors describe the approach, and argue that the community is in a good position to realise this vision, covering challenges and possible solutions.

In [14], Kamburjan et al. address the problem of reestablishing the twinning property between a digital twin and a physical twin when the latter changes over time due to adaptation and reconfiguration. To this aim, the authors propose

to combine feedback loops from the well-known MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) reference control model for organising autonomous and self-adaptive systems with semantic reflection to automatically ensure that digital artefacts twine correctly with physical systems, i.e., that the resulting system is *twinned-by-construction*.

In [10], Coto et al. explore the impact that the introduction of syntactic or semantic restrictions to rule out models that could lead to communication glitches like message loss or deadlocks have on the usability of formal modelling. To this aim, the authors benchmark the use of a formal choreographic modelling language designed to support the *correctness-by-construction* principle of message-passing systems for the modelling of real business processes taken from the official documentation of European customs business process models. As such, the current paper provides an initial comparison between modelling with the Business Process Modelling Notation (BPMN), widely used in practice, and formal choreographic approaches, deeply investigated theoretically.

### 3.2 CbC and RV: Configurable and Cyber-Physical Systems

In [12], Dubslaff and Köhl coin *configurable-by-construction* runtime monitoring inspired by automata-theoretic *runtime verification*, featured transition systems from software product lines, and stream-based runtime monitoring. The authors consider runtime monitoring with variability in the system being monitored as well as in the monitor itself. The need for configurable monitors is motivated with an example of real driving emissions tests. In this running example, both the system being monitored as well as the monitor itself are configurable as they depend on the sensor configuration of the car. Considering monitors to be themselves configurable opens many new challenges, several of which are discussed in the current paper. The authors introduce an automata-based framework for configurable monitors and their synthesis from featured LTL specifications, and they present an extension of the stream-based specification language Lola to the setting of *configurable systems*.

In [15], Kittelmann et al. introduce a method to refine hybrid automata representing verified (safety-critical) *cyber-physical systems* into corresponding executable source code amenable to *runtime verification*. Their approach employs ArchiCorC, a component-based architecture-level tool incorporating the *correctness-by-construction* paradigm, to generate code. As a case study, the authors consider the context of driving maneuvers, and apply ArchiCorC in this context. Subsequent simulation of executable and verified maneuvers allows requirement validation for various scenarios.

In [1], Abbas and Bonakdarpour address the setting of a hybrid (continuous or discrete) distributed system (model). They analyse the opportunities and challenges concerning the exploitation of various kinds of knowledge for the benefit of more effective, or more efficient, *runtime verification* in the context of *cyber-physical systems*. The current paper describes cutting-edge problem space, provides ideas and directions for solutions, and points out numerous directions for further research.

### 3.3 XbC: Security, Resilience, and Consumption Properties

In [11], Dam et al. present a proof of concept combining fault localisation techniques and automated program repair methods for *security* errors in C programs. The approach is to exploit program traces using statistical model checking and to apply patch candidates that are identified by a genetic algorithm. For this purpose, the authors evaluate populations of patches using a novel Q-function, which indicates the probability to choose patches at a certain validation state. They demonstrate the effectiveness of their approach for memory usage-related errors using benchmarks from the automotive domain.

In [2], Adelt et al. discuss the analysis of safety and *resilience* of intelligent hybrid systems. In contrast to purely deductive verification methods, which mostly focus on worst-case behaviour based on pessimistic assumptions, they propose a novel methodology that combines deductive verification of hybrid systems with statistical model checking. This enables (1) to construct provably safe and resilient systems, but also (2) to achieve certain performance levels with a statistical guarantee. The authors demonstrate applicability on an intelligent water distribution system, whose behaviour was learned through reinforcement learning. Based on the proposed approach, the water distribution system is proven to be safe and resilient towards pump failures with respect to failure probability and repair time, while guaranteeing low energy costs.

In [19], Riganelli et al. present the test tool Test4Enforcers that is able to validate the correctness of software enforcers for both functional and non-functional properties. The functional correctness part was presented in prior work. The current paper focusses on the *non-functional properties* of *power consumption*, *memory consumption*, *launch time*, and *responsiveness*. To this aim, Test4Enforcers generates a test suite and executes this suite on apps with and without enforcers. By comparing the performance measurements, degradation introduced by faulty enforcers is detected.

### 3.4 CbC and RV: Reinforcement Learning and Synthesis

In [21], Tappler et al. present a way to avoid safety violations during *reinforcement automata learning* using *shield synthesis*. The unknown environments are modelled as Markov Decision Processes (MDPs) with associated cost/reward functions. Shields are constructed from recorded traces, and subsequently serve as runtime guards/monitors that block actions that introduce a too high risk of safety violations within the next $k$ steps of the automaton. Iteratively, the collected data is used to learn new MDPs with higher accuracy, resulting in shields able to prevent more safety violations. An implementation and application to a case study of a Q-learning agent show that while the agent explores the environment during training, the improved learned models lead to shields that are able to prevent many safety violations.

In [8], Berducci and Grosu present a pipeline for solving Constrained Markov Decision Processes (MDPs), starting from formal requirements, in a *correct-by-construction* style. A *reinforcement learning*-based iterative approach to the

*synthesis* of control policies has to balance target, safety, and comfort in its objective. The model-based reinforcement-learning algorithm presented by the authors ensures with high probability that policy updates only occur when safety performance stays the same or improves. To improve on previous approaches, the authors propose to combine model-based and model-free approaches for more data-efficient algorithms. The authors suggest that the resulting dynamics model fit to the data can be used to validate the obtained policy before deployment.

In [4], Azzopardi et al. study the potential of combining *controller synthesis* and *runtime verification*. Controller synthesis is a general technique to ensure *correctness-by-construction*: given a controllable system (and environment) description and a specification of the system behaviour to achieve, controller synthesis allows to obtain a controlled version of the system meeting the specification. However, this promise is only met if the environment in which the system is running is modelled correctly, and the computational resources allow the synthesis of a corresponding strategy. Runtime verification considers the actual behaviour of the system and its environment may be used in combination with controller synthesis in several ways. The authors identify and discuss three different patterns: (1) monitors that identify when control is needed and what needs to be controlled; (2) monitors that identify violation of environment assumptions; and (3) monitors that mediate between different controllers and other agents.

In [13], Gorostiaga et al. elaborate on a concrete combination of *correctness-by-construction* and *runtime verification* in the setting of controlling Unmanned Aerial Vehicles (UAVs). In this setting, often temporal planning is used as a method for getting an a priori correct plan to steer the UAVs. Again, assumptions about the environment as well as simplifications have to be made to obtain a plan. In the current paper, it is shown how stream runtime verification can be integrated with the temporal planning to monitor the assumptions about the environment and to mitigate deficiencies of the current plan.

## References

1. Abbas, H., Bonakdarpour, B.: Leveraging system dynamics in runtime verification of cyber-physical systems. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 264–278. Springer, Heidelberg (2022)
2. Adelt, J., Herber, P., Niehage, M., Remke, A.: Towards safe and resilient hybrid systems in the presence of learning and uncertainty. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 299–319. Springer, Heidelberg (2022)
3. Ahrendt, W., Gurov, D., Johansson, M., Rümmer, P.: TriCo – triple co-piloting of implementation, specification and tests. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 174–187. Springer, Heidelberg (2022)
4. Azzopardi, S., Piterman, N., Schneider, G.: Runtime verification meets controller synthesis. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 382–396. Springer, Heidelberg (2022)
5. ter Beek, M.H., Cleophas, L., Legay, A., Schaefer, I., Watson, B.W.: X-by-construction: correctness meets probability. In: Margaria, T., Steffen, B. (eds.)

ISoLA 2020. LNCS, vol. 12476, pp. 211–215. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61362-4_11
6. ter Beek, M.H., Cleophas, L., Schaefer, I., Watson, B.W.: X-by-construction. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11244, pp. 359–364. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03418-4_21
7. ter Beek, M.H., Hähnle, R., Schaefer, I.: Correctness-by-construction and post-hoc verification: friends or foes? In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9952, pp. 723–729. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47166-2_51
8. Berducci, L., Grosu, R.: Safe policy improvement in constrained Markov decision processes. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 360–381. Springer, Heidelberg (2022)
9. Colombo, C., et al.: COST action IC1402 runtime verification beyond monitoring. In: Colombo, C., Leucker, M. (eds.) RV 2018. LNCS, vol. 11237, pp. 18–26. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03769-7_2
10. Coto, A., Barbanera, F., Lanese, I., Rossi, D., Tuosto, E.: On formal choreographic modelling: a case study in EU business processes. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 205–219. Springer, Heidelberg (2022)
11. Dam, K.H.T., Duchêne, F., Given-Wilson, T., Cordy, M., Legay, A.: Automated repair of security errors in C programs via statistical model checking: a proof of concept. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 279–298. Springer, Heidelberg (2022)
12. Dubslaff, C., Köhl, M.A.: Configurable-by-construction runtime monitoring. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 220–241. Springer, Heidelberg (2022)
13. Gorostiaga, F., Zudaire, S., Sánchez, C., Schneider, G., Uchitel, S.: Assumption monitoring of temporal task planning using stream runtime verification. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 397–414. Springer, Heidelberg (2022)
14. Kamburjan, E., Din, C.C., Schlatte, R., Tapia Tarifa, S.L., Johnsen, E.B.: Twinning-by-construction: ensuring correctness for self-adaptive digital twins. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 188–204. Springer, Heidelberg (2022)
15. Kittelmann, A., Runge, T., Bordis, T., Schaefer, I.: Runtime verification of correct-by-construction driving maneuvers. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 242–263. Springer, Heidelberg (2022)
16. Larsen, K.G., Legay, A., Nolte, G., Schlüter, M., Stoelinga, M., Steffen, B.: Introduction to formal methods meet machine learning (F3ML). In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13703, pp. 393–405. Springer, Heidelberg (2022)
17. Leucker, M., Schallhart, C.: A brief account of runtime verification. J. Log. Algebraic Methods Program. **78**(5), 293–303 (2009). https://doi.org/10.1016/j.jlap.2008.08.004
18. Nayak, S.P., Neider, D., Zimmermann, M.: Robustness-by-construction synthesis: adapting to the environment at runtime. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 149–173. Springer, Heidelberg (2022)
19. Riganelli, O., Micucci, D., Mariani, L.: Non-functional testing of runtime enforcers in Android. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 320–334. Springer, Heidelberg (2022)

20. Seisenberger, M., et al.: Safe and secure future AI-driven railway technologies: challenges for formal methods in railway. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13704, pp. 246–268. Springer, Heidelberg (2022)
21. Tappler, M., Pranger, S., Könighofer, B., Muškardin, E., Bloem, R., Larsen, K.: Automata learning meets shielding. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022, LNCS 13701, pp. 335–359. Springer, Heidelberg (2022)
22. Wing, J.M.: Trustworthy AI. Commun. ACM **64**(10), 64–71 (2021). https://doi.org/10.1145/3448248