XANDAR: Exploiting the X-by-Construction Paradigm in Model-based Development of Safety-critical Systems

Leonard Masing[1], Tobias Dörr[1], Florian Schade[1], Juergen Becker[1], Georgios Keramidas[2,9], Christos P. Antonopoulos[2], Michail Mavropoulos[2], Efstratios Tiganourias[2,9], Vasilios Kelefouras[2], Konstantinos Antonopoulos[2], Nikolaos Voros[2], Umut Durak[3], Alexander Ahlbrecht[3], Wanja Zaeske[3], Christos Panagiotou[4], Dimitris Karadimas[4], Nico Adler[5], Andreas Sailer[5], Raphael Weber[5], Thomas Wilhelm[5], Geza Nemeth[6], Fahad Siddiqui[7], Rafiullah Khan[7], Vahid Garousi[7], Sakir Sezer[7], Victor Morales[8]

[1]Karlsruhe Institute of Technology, Germany
Email: {leonard.masing, juergen.becker}@kit.edu
[2]University of Peloponnese, Greece
[3]German Aerospace Center (DLR), Institute of Flight Systems, Germany
[4]AVN Innovative Technology Solutions Limited, Cyprus
[5]Vector Informatik GmbH, Germany
[6]Bayerische Motoren Werke Aktiengesellschaft, Germany
[7]Queen's University, Belfast, UK
[8]Fent Innovative Software Solutions, SL, Spain
[9]Aristotle University of Thessaloniki, Greece

***Abstract*—Realizing desired properties "by construction" is a highly appealing goal in the design of safety-critical embedded systems. As verification and validation tasks in this domain are often both challenging and time-consuming, the by-construction paradigm is a promising solution to increase design productivity and reduce design errors. In the XANDAR project, partners from industry and academia develop a toolchain that will advance current development processes by employing a model-based X-by-Construction (XbC) approach. XANDAR defines a development process, metamodel extensions, a library of safety and security patterns, and investigates many further techniques for design automation, verification, and validation. The developed toolchain will use a hypervisor-based platform, targeting future centralized, AI-capable high-performance embedded processing systems. It is co-developed and validated in both an avionics use case for situation perception and pilot assistance as well as an automotive use case for autonomous driving.**

***Index Terms*—X-by-Construction, Model-based development, Real-time systems, Safety-critical systems, Hypervisors**

I. Introduction

The next generation of networked embedded systems has caused the need for software technologies that combine high performance with a sufficient degree of resilience from both a safety and a security perspective. However, current trends such as the steadily increasing importance of Machine Learning (ML) and Artificial Intelligence (AI) functions turn the design of trustworthy systems into a challenging endeavor.

A prominent example of this is the domain of autonomous driving, where malfunctions and security vulnerabilities can lead to serious incidents causing physical harm to humans or their environment. Therefore, approaches to minimize the risk caused by faults of such intelligent systems while minimizing development costs are essential in order to maintain vital services and public confidence in them.

The XANDAR project tackles these challenges by leveraging the X-by-Construction paradigm for the model-based development of embedded software systems.

The project consortium consists of eight universities, research institutions, and companies across Europe: Karlsruhe Institute of Technology as developer of the dynamic modeling extension and automatic system software/service generation considering non-functional requirements and coordinator of the project; the University of Peloponnese as designer of system architecture and compiler-level transformations; AVN Innovative Technology Solutions Limited as provider of the continuous integration/deployment platform; Queen's University of Belfast as reference center for the design and development of the platform security architecture; Vector Informatik GmbH as major supplier of software tools for the automotive market; fentISS as major supplier of virtualization solutions of highly critical embedded systems in space applications; and, finally, Bayerische Motoren Werke (BMW) and the German Aerospace Center (DLR) as executors of use cases in the automotive and avionics domain, respectively.

This article gives an overview of the project and its first progress after successfully completing the initial planning stage. Since it is still early in the project timeline, this contribution focuses on the big picture and presents the pursued approach in a concise manner. It does not intend to provide a detailed description of the achieved technical progress. Instead, it will serve as a reference for future publications treating individual topics in detail.

978-3-9819263-6-1/DATE22/©2022 EDAA

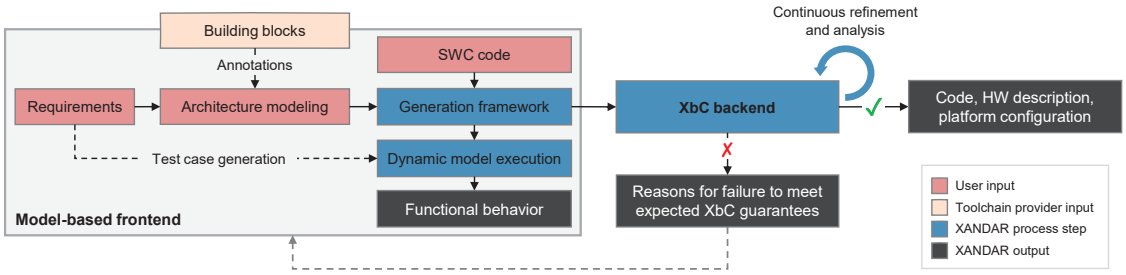

Fig. 1. Simplified flow chart of the XANDAR process

II. Background

Multicore architectures are becoming an essential building block of modern embedded systems [1]. Issues caused by concurrent execution on such computing platforms have been intensively researched and, over the years, resulted in various new approaches to develop software for such architectures. Increasingly powerful parallelizing compilers, which automate crucial steps of this process, have been introduced and reduce the potential for manual design errors significantly. Especially in safety-critical domains, where tasks are often subject to hard real-time requirements, nondeterminism caused by interleaved execution is highly critical. Therefore, tools and approaches used in such a context need to be validated and qualified in order to guarantee that implementations they generate are correct. This means that it is necessary to consider implementation details such as hardware properties, tasks schedules, and the potential for interferences on communication channels. Approaches to tackle this such as [2] and [3] exist, but they usually focus on low-level code transformations and do not consider the system design from a requirement-driven perspective. Furthermore, they are often concerned with timing and do not deal with other non-functional properties such as safety, security or performance.

Model-based development at the system level has also gained significant traction and impact during the past years. For example, MathWorks provides a toolchain for the design of DO-178C-compliant systems by enhancing their modeling framework with capabilities for coverage analysis, model conformance checks, Model-in-the-Loop (MiL) functional testing as well as automatic code generation. However, the possibility to specify non-functional requirements at a high level of abstraction and to automatically transform them into suitable implementation artifacts is often lacking in such tools. Furthermore, they usually do not provide desired flexibility without vendor lock-in and have a limited support for AI/ML components, which is a major focus of XANDAR.

III. The XANDAR Approach

The XANDAR project will deliver a holistic toolchain for the development of future high-performance embedded systems in safety-critical environments. The toolchain allows designers to make use of the XANDAR design methodology and, at the same time, is an implementation of the XANDAR

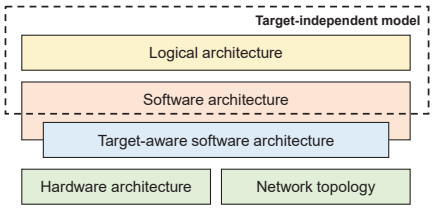

Fig. 2. Abstraction levels captured by the XANDAR metamodel

process. This process is shown in Fig. 1 and defines inputs to be supplied by the toolchain user as well as design steps that are automatically applied to the initial or a refined version of this user input. Logically, it can be divided into a model-based portion at the frontend and an automated backend that generates implementation artifacts in such a manner that the desired X-by-Construction guarantees are obtained.

In the model-based frontend of the process, the user interacts with the XANDAR toolchain to describe relevant requirements and to specify a high-level representation of the envisaged software system. A key element in this modeling process is the *software component* (SWC), which represents a certain functionality to be implemented in software. When it comes to the actual code to be executed by a SWC, the methodology offers full flexibility: the code can be either generated from behavioral models or it can be provided as source code. This enables compatibility with legacy code and frameworks, especially for tasks such as AI algorithms.

Verification and validation (V&V) are firmly embedded into the whole process, utilizing a behavioral simulation for dynamic model execution pairing the user model with an environment model and a test framework.

The toolchain backend will implement the transformations while considering the semantic preservation and fulfillment of timing bounds implied by the model.

A. Model-based Design

The model-based frontend of the toolchain is designed to be used interactively and in an iterative manner. Eventually, it is expected to provide the backend of the toolchain with a sufficiently detailed description of the envisaged system, which will then be processed in the automated analysis and system

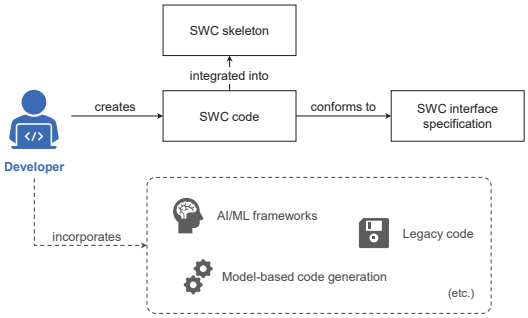

Fig. 3. Code creation process for SWCs

generation steps. Before the backend is triggered, however, the frontend allows the designer to perform early V&V activities.

The iteratively defined input model spans the abstraction levels shown in Fig. 2. They are inspired by PREEvision, a tool for the model-based design of electric/electronic (E/E) architectures in the automotive or similar domains [4]. For the purposes of the project, a subset of the abstraction levels covered by PREEvision was selected, while necessary metamodel extensions are currently being developed.

At the *logical architecture* level, the toolchain user captures an abstract representation of the functions that the system under development is expected to deliver, for instance by specifying an activity chain from a sensor via logical functions to an actuator. Important aspects that the XANDAR metamodel supports at this level are the annotation of timing requirements to segments of activity chains, modeling notations for the behavior of logical functions, and description capabilities for desired safety and security policies.

The *software architecture* describes the envisaged system as a network of SWCs, where each SWC has input and output ports. Communication channels connect output and input ports of different SWCs. To facilitate the code specification process, the XANDAR toolchain automatically generates a *SWC skeleton* for each SWC. The designer is then expected to populate all such skeletons as shown in Fig. 3. In addition, a set of *execution parameters* needs to be specified for each SWC. These parameters describe envisaged timing properties such as the trigger period. It is important to note that up until this point, the model instance is entirely independent of execution platforms. Especially the execution parameters need to be interpreted as an idealized description of the desired timing behavior. Models such as the logical execution time (LET) paradigm [5] or Ptides [6] are currently under investigation to capture these parameters.

The *target-aware software architecture* is comparable to the software architecture from PREEvision, captures very early deployment decisions, and is therefore no longer independent of a target. Finally, the combination of *hardware architecture* and *network topology* captures all on-chip and system-level aspects of the utilized execution environment. Most importantly, this is the level at which each SWC is mapped to a specific execution platform. Key extensions that the XANDAR project develops in the context of these target-specific abstraction levels are capabilities to specify target parameters related to low-level mechanisms for safety and security as well as to the timing of executed SWCs.

The following sections give an overview of three specific toolchain capabilities that build up on the metamodel extensions outlined above.

B. Dynamic Model Execution

One such capability is the *dynamic model execution* that the toolchain performs as part of its model-based frontend. In this step, the dynamic behavior of the described SWC network is simulated using the Ptolemy II environment [7], which has been specifically designed to tackle challenges associated with the modeling of cyber-physical systems [8]. More specifically, the toolchain uses relevant knowledge of the software architecture (such as SWC code or specified execution parameters) to predict its runtime behavior. This allows toolchain users to perform an early validation or verification of the functional behavior described in the target-independent model.

In particular, XANDAR will advance the state of the art by wrapping Ptolemy II in an orchestration framework that automatically creates a discrete-event simulation for a given software architecture and performs test cases the toolchain user derived from functional requirements. Test cases can target the open-loop or the closed-loop behavior of the modeled system and consider the timing of system-level events (such as the time at which a sensor input is read or two SWCs exchange information). In the open-loop case, inputs from the environment must be specified as a time series of values. In the closed-loop case, the designer has the opportunity to supply a plant model created in Ptolemy II. Plant models are automatically connected to the simulation model of the SWC network by the orchestration framework. In both cases, expected events (such as an actuator writing a certain output to the environment at a specific time) can be specified and will be automatically compared to the simulation results.

This capability guides the toolchain user towards a target-independent model instance (including suitable SWC code) that meets functional and system-level timing requirements. A key property of the XANDAR process is that the behavior exhibited by such a model instance is entirely deterministic and that the assumptions that lead to this determinism are well-defined. In the backend, they serve as requirements that must be met and are therefore essential in the achievement of behavioral X-by-Construction guarantees.

C. X-by-Construction Patterns & Transformations

The envisaged XANDAR process utilizes a library of safety and security patterns, which will be developed as part of the project and can be annotated during system modeling as described in Section III-A. Annotated patterns will be applied to the respective system artifacts throughout the whole software and system generation process.

The patterns may be annotated to model artifacts such as SWCs by the toolchain user. In addition, XANDAR will

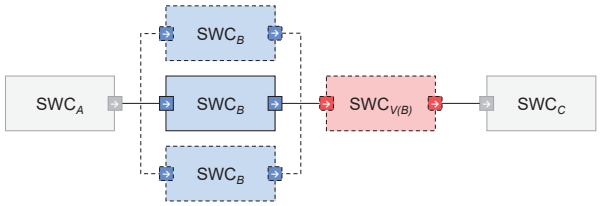

Fig. 4. Software architecture transformation by the TMR pattern

also investigate automated policy inference based on specified requirements. The toolchain will then realize the patterns by applying transformations, either on model level, on code level, or on a mapping and configuration level. The rules and templates needed for these transformations are part of the pattern in the pattern library.

XANDAR will deliver a set of patterns, ranging from simple examples such as N-modular redundancy at the hypervisor level, similar to the approach presented in [9], over the synthesis of runtime mechanisms for the control of information flows based on [10], anomaly detection through monitors based on RTLola [11] to security-specific patterns for confidentiality of data-at-rest and data-in-motion [12], to name a few.

As an example, Fig. 4 shows the application of a triple-modular redundancy (TMR) safety pattern. In the software modeling phase of the process, the pattern is annotated to software component SWC$_B$ in the software architecture. The TMR pattern is applied in the backend of the toolchain and consists of a model transformation and a code generation step. First, the toolchain transforms the software architecture by triplicating SWC$_B$, adding a voter component SWC$_{V(B)}$, and re-routing the inter-SWC communication channels as shown using dashed lines. Finally, it generates SWC code for the voter. Additionally, mapping constraints can be added, enforcing that the redundant SWC instances are mapped to different cores or execution platforms.

This example shows how the envisaged process, which aims to close the gap between high-level requirements and the low-level implementation, can contribute to the fulfillment of safety and security requirements by design.

D. V&V of Timing Requirements

A main goal of XANDAR is to guarantee non-functional properties such as safety, security, and performance by construction. In this regard, it is important to note that timing is considered from both a functional and a non-functional perspective in the XANDAR project.

As described in Section III-B, we consider high-level timing aspects during the behavioral simulation performed in Ptolemy II. This is done to account for the fact that certain timing properties have a direct impact on the behavior of an embedded software system.

In addition, timing is considered a non-functional constraint that is not covered by a specific XbC pattern. However, is must be thoroughly considered at all abstraction levels shown in Fig. 2 as well as during the automated steps performed by the XbC backend. Since the toolchain performs a variety of model and code transformations, e.g., as part of a safety pattern application, this treatment is a challenging endeavor.

At the logical architecture level, the toolchain frontend will enable the user to perform early timing validations (for example considering the consistency of time budgets allocated to parts an activity chain). Furthermore, timing simulations will give the designer the opportunity to predict how decisions related to the target-aware software architecture or the hardware network topology impact the achievable timing characteristics. To some degree, these target-specific simulations take resource sharing effects into account and support the designer in the iterative refinement of the input model. For this purpose, the project will particularly build up on the TA Tool Suite developed by Vector Informatik GmbH.

Finally, timing-related V&V activities are an essential step in the XbC backend. Here, these activities must ensure that derived implementation artifacts meet timing requirements given by the toolchain user or defined by a previous design step. This applies particularly to the execution parameters of SWCs described in Section III-A. They capture an idealized runtime behavior that must be ensured by the actual target in order to obtain the desired X-by-Construction guarantees. In the XbC backend, a fine-grained timing simulation will be performed to verify that this is the case.

IV. Target Platform

The project is specifically considering centralized high-performance processing platforms, which are expected to replace the vast amount of controllers in many of today's systems. These platforms will contain multiple processing cores and special hardware accelerators, especially for AI, hosting many different applications and functions in a mixed-criticality system. For realizing the safety and security patterns defined by XANDAR, the underlying platform needs to provide corresponding capabilities.

A basis for operating such applications is a hypervisor, acting as a separation kernel. It is used to partition the platform and to provide guest software with processing resources such as CPU time, memory, as well as access to I/O components. The logical containers in which guest software resides are referred to as *partitions*. By restricting the access of partitions to processing resources, hypervisors isolate guest software from each other and are powerful platforms to provide freedom from interference between partitions.

In XANDAR, the XtratuM hypervisor [13] will be used. It is a paravirtualizing type-1 hypervisor providing time and space isolation between partitions. This is achieved by leveraging isolation mechanisms provided by the hardware, or alternatively by paravirtualization. Temporal execution of partitions follows a time-based activation scheduling policy, inter-partition communication is provided by sampling-type and queuing-type communication channels, inspired by the avionics standard ARINC 653 [14].

To provide isolation between software components, each component may be mapped to a separate hypervisor parti-

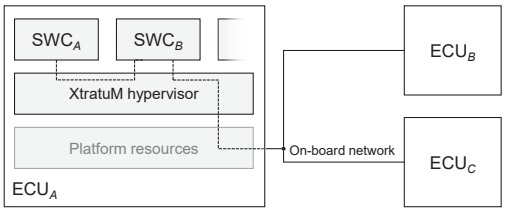

Fig. 5. XANDAR platform architecture

tion. Consequently, inter-SWC communication will be realized using the hypervisor's communication mechanisms. Thereby, the hypervisor's isolation features can be used as a base to fulfill safety and security requirements. Access to platform resources such as sensor and actuator interfaces and hardware accelerators can be controlled on an SWC level. The resulting architecture is outlined in Fig. 5.

Based on the system models defined by the toolchain user and modified by the pattern-based transformation, the target platform configuration is derived during the system generation phase of the XANDAR process. With respect to the envisioned target platform, central challenges tackled by the XANDAR project include the generation of implementations exhibiting the same timing behavior as the simulation. For compatibility with XANDAR metamodel described above, concepts such as LET [5] as well as its system-level extension [15], [16] are investigated as possible approaches.

V. Validation

The development of the XANDAR toolchain is guided by two use case partners who motivate the development of the model-based XbC approach and support the consortium in identifying and eliminating potential shortcomings. The use cases will serve as validation scenarios for the entire toolchain at the end of the project. However, concepts developed within the project will be described in a way that is applicable to application development in various safety-critical contexts.

The use cases of the project cover the two major fields of avionics and automotive mobility. The avionics use case is aimed at urban air mobility concepts, which demand a high degree of autonomous features with fail-operational and fault-tolerant requirements. Specifically, flight assistance features developed with the XANDAR toolchain will be validated on existing prototypes. The automotive use case is focused on autonomous driving, including passenger detection and steering functionality. Both use cases define a set of requirements towards the toolchain, specifically addressing requirements about functionality, timing, reliability, safety and security. The functional requirements are further elaborated on multiple levels, i.e., at the system, sub-system, software item, hardware, and the AI/ML level.

VI. Conclusion

The XANDAR project will deliver a model-based toolchain following the X-by-Construction paradigm to reduce cost and errors in the design of safety-critical systems.

This article describes the results of the initial stage of the project and gives an overview of the planned activities. Specifically, the extensions to the utilized metamodel, the relationship to the MiL framework for dynamic model execution, X-by-Construction patterns and transformations, and timing considerations of XANDAR are presented. In future publications, the consortium will report on specific aspects of the project in greater detail.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957210.

References

[1] M. Hassan, "Heterogeneous MPSoCs for mixed-criticality systems: Challenges and opportunities," *IEEE Design & Test*, vol. 35, no. 4, pp. 47–55, 2018.

[2] S. Derrien, I. Puaut, P. Alefragis, M. Bednara, H. Bucher, C. David, Y. Debray, U. Durak, I. Fassi, C. Ferdinand, D. Hardy, A. Kritikakou, G. Rauwerda, S. Reder, M. Sicks, T. Stripf, K. Sunesen, T. ter Braak, N. Voros, and J. Becker, "WCET-aware parallelization of model-based applications for multi-cores: The ARGO approach," in *Design, Automation & Test in Europe Conference Exhibition (DATE)*, 2017.

[3] K. Didier, D. Potop-Butucaru, G. Iooss, A. Cohen, J. Souyris, P. Baufreton, and A. Graillat, "Correct-by-construction parallelization of hard real-time avionics applications on off-the-shelf predictable hardware," *ACM Trans. Archit. Code Optim.*, vol. 16, no. 3, Jul. 2019.

[4] J. Schäuffele, "E/E architectural design and optimization using PREEvision," in *SAE 2016 World Congress and Exhibition*, 2016.

[5] C. M. Kirsch and A. Sokolova, "The logical execution time paradigm," in *Advances in Real-Time Systems*. Springer, Berlin, Heidelberg, 2012.

[6] J. Cardoso, P. Derler, J. C. Eidson, E. A. Lee, S. Matic, and J. Zhao, Yang amd Zou, "Modeling timed systems," in *System Design, Modeling, and Simulation using Ptolemy II*, C. Ptolemaeus, Ed. Ptolemy.org, 2014.

[7] C. Ptolemaeus, Ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014.

[8] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling cyber–physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[9] E. Missimer, R. West, and Y. Li, "Distributed real-time fault tolerance on a virtualized multi-core system," in *10th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*, Madrid, Spain, 2014.

[10] T. Dörr, T. Sandmann, and J. Becker, "A formal model for the automatic configuration of access protection units in MPSoC-based embedded systems," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, Kranj, Slovenia, 2020.

[11] P. Faymonville, B. Finkbeiner, M. Schledjewski, M. Schwenger, M. Stenger, L. Tentrup, and H. Torfah, "StreamLAB: Stream-based monitoring of cyber-physical systems," in *Computer Aided Verification*. Springer, Cham, 2019.

[12] N. A. Delessy and E. B. Fernandez, "A pattern-driven security process for SOA applications," in *2008 Third International Conference on Availability, Reliability and Security*, 2008.

[13] M. Masmano, I. Ripoll, A. Crespo, and J. J. Metge, "XtratuM: a hypervisor for safety critical embedded systems," in *Proceedings of the 11th Real-Time Linux Workshop*, 2009.

[14] ARINC, "ARINC Specification 653, Avionics Application Software Standard Interface Part 1 – Required Services," 2010.

[15] R. Ernst, L. Ahrendts, and K.-B. Gemlau, "System level LET: Mastering cause-effect chains in distributed systems," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, Washington, DC, USA, 2018.

[16] K.-B. Gemlau, L. Köhler, R. Ernst, and S. Quinton, "System-level logical execution time: Augmenting the logical execution time paradigm for distributed real-time automotive software," *ACM Transactions on Cyber-Physical Systems*, vol. 5, no. 2, 2021.