# A Contract-Based Requirement Engineering Framework for the Design of Industrial Cyber-Physical Systems

Michele Lora[1,2] and Pierluigi Nuzzo[2]
[1]University of Verona, Verona, Italy
[2]University of Southern California, Los Angeles, CA, US
{loramich,nuzzo}@usc.edu

## ABSTRACT

This work-in-progress paper presents our current effort toward the development of compositional modeling formalisms and scalable algorithms for high-assurance design of industrial cyber-physical systems, with emphasis on smart manufacturing systems. A requirement engineering methodology is implemented within CHASE, a software framework supporting contract-based representations of systems and components to facilitate analysis and design space exploration. We provide an overview of CHASE and discuss its application to the design of a robotic arm.

This paper is accompanied by a poster describing the architecture of CHASE and a demonstration of its application to the case study.

## 1 INTRODUCTION

Production lines are evolving into complex networked systems, possibly equipped with teams of robots that transport and manipulate materials and products, sensors that gather large amounts of data, and computing platforms that execute algorithms for situational awareness, intelligent control, and optimization of various aspects of the value production chain [5]. These opportunities come, however, with a series of intellectual and engineering challenges. Smart manufacturing systems must integrate a very diverse set of components while offering strong guarantees in terms of functionality, reliability, safety, and cost. The heterogeneity in components and system requirements inevitably calls for the orchestration of models, specification languages, and design constraints of different nature to effectively represent a design space that is challenging to exhaustively explore in reasonable time. Structured methodologies, supported by formalisms with rigorous semantics, and automated by tools, become crucial to tackle the complexity of designing today's industrial cyber-physical systems (CPSs) [4]. In this context, the *DeFacto (Design Automation for Smart Factories)* project[1] aims at advancing industrial CPS design and its automation by developing new modeling formalisms, scalable algorithms, and tools.

DeFacto addresses industrial CPS design using compositional abstractions of system behaviors based on assume-guarantee (A/G) contracts [1]. Contracts are mathematical models offering rigorous composition rules and providing mechanisms to validate the design requirements, analyze complex system behaviors, and develop system components in a modular and hierarchical way. Central to the project is the *Contract-based Heterogeneous Analysis and System Exploration (CHASE)* framework [6], a requirement engineering framework combining specification and modeling formalisms with rigorous verification and synthesis procedures relying on A/G contracts. CHASE also provides interfaces for the designers to implement novel design tools and methodologies.

[1]DeFacto Project website: `https://defacto-h2020.github.io`

This research is supported in part by the European Commission through the project DeFacto, under grant no. H2020-MSCA-IF-2019-894237, by the US National Science Foundation (NSF) under Awards 1839842 and 1846524, the Office of Naval Research (ONR) under Award N00014-20-1-2258, and the Defense Advanced Research Projects Agency (DARPA) under Award HR00112010003.

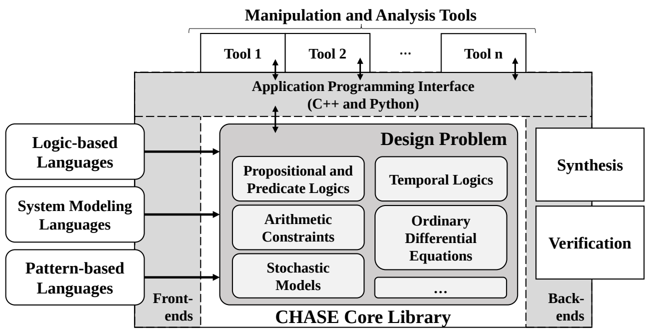

**Figure 1: Structure of the CHASE framework.**

We present an overview of the architecture of CHASE and its capabilities together with preliminary results on its application to a real-world industrial scenario. The results, presented in a practical demonstration, show the effectiveness of a rigorous, compositional requirement engineering approach based on A/G contracts. CHASE is released open-source[2]. The source files implementing the case study are available in the *demo* section of the framework repository.

## 2 THE CHASE FRAMEWORK

Figure 1 shows the architecture of CHASE. Its main component is the representation *core library*, which provides a set of classes to represent requirements, components, and system models in terms of A/G contracts. Intuitively, an A/G contract represents the interface of a component as a pair of assumptions and guarantees. The assumptions are the behaviors that a component expects from the environment. The guarantees are the behaviors the component promises in the context of the assumptions. Contracts are then mathematical models with rigorous composition rules that provide mechanisms to analyze system behaviors, validate design requirements, and develop system components in a modular and hierarchical way. The CHASE library supports the representation of A/G contracts expressed in propositional logic or Linear Temporal Logic (LTL) and implements the operations defined by the contract algebra [1]. We can then exploit the compositionality and rigor provided by A/G contracts and their algebra to automate verification and synthesis tasks. The library has been further extended to support the representation of Signal Temporal Logic and

[2]CHASE repository: `https://github.com/chase-cps/chase`

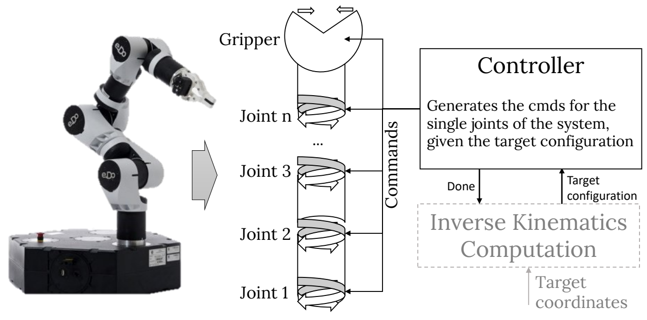
Gripper
Joint n
...
Joint 3
Joint 2
Joint 1
Commands
Controller
Generates the cmds for the single joints of the system, given the target configuration
Done
Target configuration
Inverse Kinematics Computation
Target coordinates

**Figure 2: Structure of the robotic arm.**

Metric Temporal Logic, enabling the use of the most appropriate formalism for the specification of the design requirements. Finally, CHASE supports the representation of arithmetic constraints on real numbers, dynamical systems in the state space, and probability distributions to specify stochastic components and probabilistic constraints.

A set of methods allow accessing the functionalities of the core library to manipulate the design representations. These methods are exported to designers by C++ and Python Application Programming Interfaces (APIs). Design methodologies can be implemented on top of CHASE by writing tools that exploit these APIs.

Front-end and back-end tools can also be developed on top of the CHASE library. Front-end tools are used to aid the formalization of the requirements, usually expressed in semi-formal languages, by encoding them in the formal constructs of the core library. CHASE currently provides two front-end tools. The `DSL-Tool` parses a domain specific language capturing requirements on system architectures specified by graphs. The `Logic-Tool` allows explicitly specifying requirements and components using A/G contracts in propositional logic or LTL. Back-end tools interface the internal representation provided by the core library to external solvers. CHASE currently provides a back-end tool interfacing to NuSMV [2] to perform contract-based verification tasks such as checking contract consistency, compatibility, and refinement. A second back-end tool interfaces to a reactive synthesis tool [3] to synthesize control strategies from LTL contracts.

## 3 CASE STUDY

We use CHASE to synthesize control strategies for a robotic arm. Figure 2 shows the structure of the robotic system, whose motion relies on a set of *revolute joints*, each one connecting two rigid bodies rotating along a common axis. The robotic arm is equipped with a gripper to grasp objects. Given a target position, a controller generates the configuration of the joints required to let the gripper reach the requested coordinates. We evaluate the scalability provided by a compositional methodology based on contracts by creating two different models of the system and specifying their requirements via the `Logic-Tool` as follows:

- *Single-contract model*: the components' behaviors and the system requirements are captured by a single contract $C_S$, expressed in LTL.
- *Compositional model*: the behavior of each joint $j_i$ is captured by a contract $C_i$, the gripper by a contract $C_g$, and the overall system mission is expressed by a contract $C_M$. All contracts are expressed in LTL. The contract $C'_S$ modeling the entire system can then be obtained by applying the *composition operation* to the components' contracts.

**Table 1: Results from the application of CHASE to the synthesis of control strategies for the robotic arm.**

| # of Joints | Single Contract ($C_S$) | | Composition ($C'_S$) | |
|---|---|---|---|---|
| | Synthesis time (s) | # of States | Verification time (s) | Synthesis time (s) |
| 1 | 0.215 | 44 | 0.04 | 0.193 |
| 2 | 19.84 | 2951 | 0.08 | 0.286 |
| 3 | 38892.82 | 163586 | 0.13 | 0.379 |
| 4 | >12 hours | – | 0.16 | 0.472 |

In the single-contract scenario, a control strategy is synthesized directly from $C_S$, resulting in a single Mealy machine. On the other hand, in the compositional scenario, we first verify whether $C'_S$ is a *refinement* of $C_S$. If the refinement holds, we synthesize control strategies from each component contract individually. The result is a set of synchronized Mealy machines, each one implementing the control strategy of either a joint or the gripper. Each Mealy machine receives as input the target status of the controlled component and generates the commands leading the component to reach its target. The Mealy machines for the joints and the gripper are composed in parallel and do not share variables.

Table 1 reports the results of our experiments. We vary the number of controlled joints to evaluate the scalability of the proposed design method. For all the experiments, we set up a time-out of twelve hours. For each configuration, the second and third columns report the time required to perform the synthesis of $C_S$ and the number of states in the generated Mealy machine, respectively. The last two columns report the time required to verify whether $C'_S$ is a refinement of $C_S$ and the total time required to synthesize the Mealy machines for each component in the system.

These results show the effectiveness of a contract-based methodology and the CHASE framework in facilitating provably-correct modular design of control protocols for industrial systems. While considering the system as a single entity leads to a quick growth in computation time, partitioning it into components allows improving the scalability of the synthesis task. Moreover, by relying on the contract algebra we can ensure that the proposed decomposition strategy is sound.

## REFERENCES

[1] Albert Benveniste et al. 2018. *Contracts for system design.* Now Publishers.
[2] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. 1999. NuSMV: A new symbolic model verifier. In *International conference on computer aided verification.* Springer, 495–499.
[3] Rüdiger Ehlers and Vasumathi Raman. 2016. Slugs: Extensible gr (1) synthesis. In *International Conference on Computer Aided Verification.* Springer, 333–339.
[4] Amit Fisher, Clas A Jacobson, Edward A Lee, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Eelco Scholte. 2014. Industrial cyber-physical systems–iCyPhy. In *Complex Systems Design & Management.* Springer, 21–37.
[5] Jay Lee, Behrad Bagheri, and Hung-An Kao. 2015. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing letters* 3 (2015), 18–23.
[6] Pierluigi Nuzzo, Michele Lora, Yishai A Feldman, and Alberto Sangiovanni-Vincentelli. 2018. CHASE: Contract-based requirement engineering for cyber-physical system design. In *Proc. of IEEE/ACM DATE 2018.* IEEE, 839–844.