Waterfall is too slow, let's go Agile: Multi-domain Coupling for Synthesizing Automotive Cyber-Physical Systems

(Invited Paper)

Debayan Roy
Technical University of Munich, Germany
debayan.roy@tum.de

Michael Balszun
Technical University of Munich, Germany
michael.balszun@tum.de

Thomas Heurung
Siemens Industry Software GmbH, Germany
thomas.heurung@siemens.com

Samarjit Chakraborty
Technical University of Munich, Germany
samarjit@tum.de

Amol Naik
Siemens Industry Software GmbH, Germany
amol.naik@siemens.com

ABSTRACT

For future autonomous vehicles, the system development life cycle must keep up with the rapid rate of innovation and changing needs of the market. *Waterfall* is too slow to react to such changes, and therefore, there is a growing emphasis to adopt *Agile* development concepts in the automotive industry. Ensuring *requirements traceability*, and thus proving functional safety, is a serious challenge in this direction. Modern cars are complex *cyber-physical systems* and are traditionally designed using a set of disjoint tools, which adds to the challenge. In this paper, we point out that *multi-domain coupling* and *design automation* using *correct-by-design* approaches can lead to safe designs even in an Agile environment. In this context, we study current industry trends. We further outline the challenges involved in *multi-domain coupling* and demonstrate using a state-of-the-art approach how these challenges can be addressed by exploiting domain-specific knowledge.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; • **Software and its engineering → Development frameworks and environments**;

KEYWORDS

Agile, traceability, safety, correct-by-design, digital continuity, multi-domain coupling, design automation.

ACM Reference Format:
Debayan Roy, Michael Balszun, Thomas Heurung, Samarjit Chakraborty, and Amol Naik. 2018. Waterfall is too slow, let's go Agile: Multi-domain Coupling for Synthesizing Automotive Cyber-Physical Systems: (Invited Paper). In *IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD '18), November 5–8, 2018, San Diego, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3240765.3243500

ICCAD '18, November 5–8, 2018, San Diego, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3243500

1 INTRODUCTION

Traditionally, automotive systems are developed according to the waterfall process model [23], often represented by one big, or multiple big staggered, V-diagrams. In this model, development proceeds in a sequence of phases (requirements, analysis, design, implementation, testing) and iterations are strongly discouraged. Thus, the requirements defined at the beginning of the development life cycle must remain unchanged throughout. This is not sustainable for the automotive industry in the current scenario [6].

More than 90% of the innovations in modern cars come from electronics and software. And the current rate, at which technology advances in these domains, easily outpaces the manufacturing cycle of a car. With the advent of electric vehicles and even more so, with autonomous driving, there is a growing urge to add the safest and the latest technologies (available in the market) into a newly launched car. Moreover, the preferences and needs of the buyers are also changing rapidly. With waterfall model, it is very time-consuming and expensive to incorporate such changes. Through this paper, we would like to draw attention to Agile development process and recommend its adoption in the automotive industry [13].

One important challenge towards implementing Agile processes is the preservation of functional safety. For autonomous vehicles, safety is of utmost importance from manufacturers' perspective. In case of failures (or accidents), the responsibility mostly lies with them. Typically, automotive software is developed according to ISO 26262 [1], the functional safety standard for road vehicles. ISO 26262 emphasizes on traceability of requirements throughout design, implementation and test phases. This essentially means that any artifact produced during the development process must be traced back to the requirements. However, in an Agile environment, requirements and designs evolve continuously and often asynchronously. Thus, traceability management is an important issue which needs to be addressed.

A modern car is a complex cyber-physical system (CPS) comprising several physical processes being controlled by a software running on a distributed electrical and electronics (E/E) platform. This platform consists of more than 100 electrical control units (ECUs) connected over a network of communication buses and gateways. Design of these systems involves a set of tools provided by different suppliers [29]. The design of control algorithms (or functions) and the subsequent code generation rely on tools like

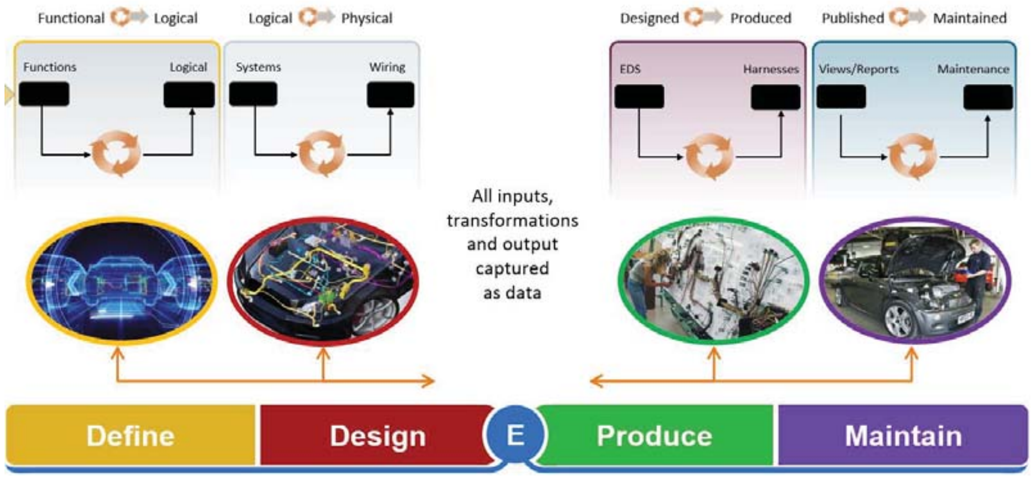

Figure 1: Digital continuity between different levels of abstraction in EDS design [17].

MATLAB/Simulink. Different tools may be used to design and configure different bus systems (CAN, FlexRay, Ethernet). A separate tool is employed to integrate the generated binaries from MATLAB/Simulink with network schedules, operating systems and drivers for ECUs. These tools are mostly disjoint, and thus, it is challenging to maintain requirements traceability across these tools.

We propose multi-domain coupling and design automation to maintain traceability which, together with correct-by-design approaches, can guarantee system safety. Ideally, tools from different design domains should be integrated together and they should co-synthesize different aspects of the overall system like control algorithms, software, hardware, communication and wiring [12]. Towards an integrated framework, there have been efforts from both industry and academia.

In this context, we study an industrial-strength tool suite called Capital [16], provided by Mentor, a Siemens business, recently acquired by Siemens Industry Software (SISW). Capital integrates the physical topology with software and hardware architectures for correct-by-construction electrical harness design. It also uses design automation tools to achieve requirements traceability and to facilitate easy and faster design adaptations.

In the academia, there has been a strong collaboration between control and embedded systems communities in recent years [22]. New techniques are developed to design controllers taking into consideration the details of the underlying platform [3, 9, 10]. Similarly, existing platform design methods are updated based on control-theoretic properties [8, 25, 30]. In addition, works addressing the problem of control-platform co-design are also gaining grounds [2, 24, 26].

In this paper, we review a state-of-the-art control-platform co-design approach [21] and discuss how the challenges of multi-domain coupling can be addressed by exploiting control-theoretic and platform-specific knowledge. We also study an integrated toolchain called Co-Flex [19] which applies the co-design approach in automotive software development. Furthermore, we describe our efforts to integrate Co-Flex with Capital. Developers can model specifications in Capital which can be imported into Co-Flex and then the results from Co-Flex can be communicated back to Capital. This will ensure data coherency between Co-Flex, Capital, and other downstream tools fed with specifications from Capital.

The rest of the paper is organized as follows. Sec. 2 discusses the current industry trends towards digital continuity and design automation using example of an automotive tool suite from SISW. Sec. 3 mentions the need for and the challenges towards correct-by-design approaches for automotive system development. It further outlines how a state-of-the-art control-platform co-design technique for FlexRay-based systems can address the challenges. Sec. 4 states the importance of an integrated framework to maintain traceability and explains one such framework which employs the co-design technique of Sec. 3. Sec. 5 shows how the framework can be used to synthesize software for an automotive case study. Sec. 6 provides concluding remarks.

2 CURRENT INDUSTRY TRENDS

As previously mentioned, automotive systems are developed in several phases and involve different design domains. Traditionally, the interactions between the phases and the design domains mostly happen via plain texts, excel sheets or UML diagrams. This requires manual intervention which makes the whole process slow and error-prone. Towards going Agile, the automotive industry is trying to get rid of these manual processes and adopt a more integrated design approach via digital continuity and design automation [11].

In this context, we study the Capital tool suite from SISW, which aids in automotive system development. This tool suite applies a data-driven approach to design electrical distribution systems (EDSs). It offers a unified framework to specify functional and architectural models and the relationship between the two. Using in-built or customized rules, it is possible to verify the validity of such models and therefore generate correct-by-construction designs. Information from these models can then be used to design the EDS and also generate the associated ordering information using rule-driven design automation tool. These models can also be exported into machine readable XML files which can be used by other tools for hardware, software and network designs.

In Agile development, a development cycle after a change in specification must be fast. It typically involves validation of requirements by assessing the generated design. In Capital, this is achieved using design automation [15]. A change in functional model can automatically affect a change in the corresponding architectural model (if required). Moreover, it is possible to explore different implementation options in negligible time and evaluate them based on different metrics like weight, cost, CPU load, PCB area and volume. For example, if two software components needs to communicate via a signal over FlexRay, then the ECUs on which they are mapped must be connected by a FlexRay bus. However, we can also evaluate the impact of mapping both software components on the same ECU, e.g., CPU load may increase beyond acceptable limits.

Capital also manages traceability between requirements and their implementations to ensure safe design. Each component in functional and architecture models is described by its attributes, thus representing a data object. And the whole system contains hundreds of thousands data artifacts which are constantly changing. Design automation plays a key role in capturing relationships between objects in such a complex and hierarchical data model and maintaining data coherency. These data are fed to the downstream domains such as electrical, hardware, network design and embedded software thus ensuring digital continuity [15], as depicted in Fig. 1. Capital allows comprehensive system modeling, as well as network and EDS designs in a single framework [18]. It can also export models as ARXML which can be used for ECU designs. Digital continuity together with such a comprehensive framework promotes true multi-domain coupling thus enabling traceability. Furthermore, Capital generates correct-by-construction designs meeting the requirements specified in the functional model, thus eliminating manual design errors.

It can be inferred from our study of Capital tool suite that automotive industry realizes the importance of Agile development and slowly moving towards it.

3 CORRECT-BY-DESIGN APPROACHES FOR AUTOMOTIVE CPS

Typically, in a CPS, there is an interplay between the algorithm controlling the physical process and its implementation on a given embedded platform [5]. However, the state-of-practice is to design the controllers and the embedded platform in their isolated design trajectories followed by a long integration and debugging phase [14]. Controllers are tuned in MATLAB/Simulink using closed-loop simulation of the plant model and the control law. The design objective is to meet the requirement on quality-of-control (QoC). The controller is then provided as a blackbox to embedded systems engineer who integrates this into the software. An automotive software is typically partitioned into tasks (or runnables) mapped on a distribued E/E platform. Now, platform design involves scheduling these tasks onto ECUs and the data transmissions between them as messages over the communication buses. There exists a set of tools for generating different parts of platform configuration.

This approach is not correct-by-design as there is no systematic interface between the controller and the platform designs. Controllers are designed with certain assumptions which may be infeasible. These assumptions include negligible sensor-to-actuator delay, zero jitter or continuous range for sampling periods. It is also difficult for embedded systems engineer to evaluate the impact of task and message schedules on QoC. Thus, the synthesized automotive system needs to be tested extensively to guarantee safety. If the test fails then the controllers and platform parameters are redesigned may be with more pessimistic assumptions. However, pessimism can lead to resource inefficiency which is also not desirable for cost-sensitive automotive systems. Thus, there is a need for correct-by-design approaches which co-synthesize controllers and platform parameters in a holistic optimization framework. However, there are several challenges towards this integrated design approach for industrial-sized systems [4, 20].

First, controller and platform designs consider different objectives. A controller is tuned for higher QoC while the platform design aims at making effective use of resources (e.g., computation and communication). Typically, a controller implemented using more resources can give better QoC. Thus, there exists a trade-off between the QoC and the resource usage and it might be possible to explore this trade-off.

Second, controller and platform design problems are very different in their formulations. Controller design involves manual or automatic tuning of design parameters (like system poles) according to certain heuristics. On the other hand, task and message scheduling can be formulated as a constrained programming (CP) model which can be solved using standard solvers like CPLEX and Gurobi. Moreover, controllers for different applications are designed separately while the platform design must consider all the applications mapped on a shared platform. Considering that these two problems have different characteristics, it is non-trivial to integrate them.

Third, the co-design problem consists of large number of dimensions and can become intractable for industrial-sized systems. For each application, tens of parameters need to be identified. For a realistic system, there may be hundreds of unknowns. Considering that the holistic problem is non-convex and non-linear, it is challenging to handle such a huge design space.

In recent years, control-platform co-design has gained importance as a research topic. There have been works addressing integrated modeling and design of automotive CPSs for different platform architectures. In this section, we will review one such technique that synthesizes controllers and the associated task and message schedules together, in a correct-by-design manner, for FlexRay-based automotive systems [21].

3.1 Control-platform co-design for FlexRay-based automotive systems

Typically, automotive E/E architectures comprise different bus clusters. Each cluster consists of several ECUs connected over a communication bus. A FlexRay cluster running number of safety-critical control applications is considered as the problem setting. Each application is composed of one or more software tasks. Distributed implementation of application is also possible. And in such cases, data is transmitted over the static segment of FlexRay. FlexRay static segment offers time-division multiple access (TDMA-) based communication which is appropriate for safety-critical applications due to its inherent determinism.

For the above setting, we study a state-of-the-art control-platform co-design approach. This approach formulates a holistic problem for controller and platform designs from joint specifications. It considers details of the plant models and QoC requirements on the control side and the architectural specifications on the platform side. A novel hybrid optimization technique, exploiting control-theoretic and platform-specific properties, is applied to solve the problem. Here, we will briefly outline how this technique addresses the aforementioned challenges.

First, a multi-objective optimization problem is formulated where QoCs of all applications and resource usage are considered simultaneously. Resource usage U is defined as the percentage of TDMA slots allocated to the control applications in the FlexRay static segment. Minimizing U would imply that slots are not conservatively reserved for control applications and unused slots can be used by other real-time data, thus improving resource efficiency. On the control side, each application can be designed based on different QoC metrics (like settling time or quadratic cost) as per requirements. It must be noted here that QoCs of different applications can be interdependent as these applications share a limited platform resource. A single objective determining the average control performance J of all applications must be formulated to make the problem tractable. Thus, the individual QoCs are first normalized with respect to their required values. And J is given by the weighted sum of the normalized QoCs. Now, for these objectives of U and J, it is desirable to obtain a Pareto front depicting the trade-off.

Second, the problems of controller design and scheduling can be integrated by correctly modeling the interplay between them. The sampling period and the delay with which a controller is designed depend on the task and message schedules and vice-versa. Properties of control theory and FlexRay protocol can be exploited here to formulate a holistic problem efficiently. According to control theory, the impact of delay on the QoC can be mostly compensated by appropriately placing the system poles. Thus, delay can be assumed to be a linear function of the sampling period. In FlexRay [7], message cycle repetition rates can take only limited number of values. The sampling period of a controller can be assumed to be equal to the period with which the corresponding tasks and messages are scheduled. Thus, sampling periods are also restricted to a prescribed set of values. Exploiting these properties, optimal controllers can be predesigned at all possible values of sampling period for each application. The controller design can therefore be integrated into the co-design problem where the solver needs to select one of these predesigned controllers. In addition, application-level constraints (like data dependency, sampling period and delay) and implementation rules (like non-conflicting resource allocations and finite resource capacities) are considered in the CP problem model.

Third, the constrained multi-objective optimization considered here cannot be solved efficiently by existing solvers. A combination of iterative search and integer linear programming (ILP) is applied to solve the problem. Note that only a finite set of values are possible for resource usage U according to its definition. These values are traversed in ascending order. For a given value of U, an optimization problem is formulated with average control performance J as the only objective and an equality constraint on U. To ensure Pareto-optimality, an additional constraint is added, i.e., the J of the current solution must be the best among all the solutions obtained so far. This single-objective optimization problem is solved in two nested layers to improve scalability, as illustrated in Fig. 2. On the outer layer, an ILP is solved with only sampling periods as variables and J as the objective such that the equality constraint on U and the Pareto criterion are satisfied. Inner layer solves a CP problem to

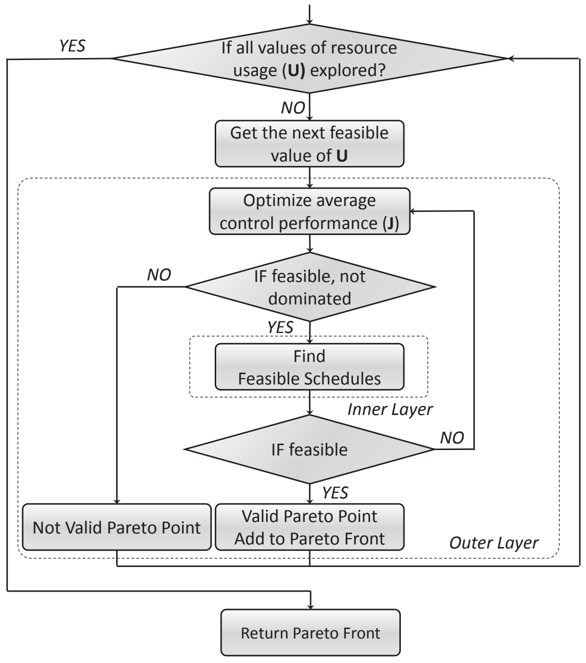
If all values of resource usage (U) explored?
YES
NO
Get the next feasible value of U
Optimize average control performance (J)
IF feasible, not dominated
NO
YES
Find Feasible Schedules
Inner Layer
IF feasible
NO
YES
Not Valid Pareto Point
Valid Pareto Point Add to Pareto Front
Outer Layer
Return Pareto Front

Figure 2: A customized hybrid optimization technique for control-platform co-design.

determine a valid set of task and message schedules satisfying the application-level and implementation-specific constraints. If a valid configuration is found then it represents a Pareto point. Otherwise, the next best set of sampling periods (still satisfying the Pareto criterion) is evaluated for feasibility.

Note that with control-platform co-design, we can do away with the long debugging and integration phase of the traditional approach. This facilitates Agile development. A change in specification at any stage would mean re-calculation of the parameters by just clicking a button. In traditional practice, adding a new application would mean keeping the current configuration intact and incrementally designing the new application. This is not cost-optimal when current configuration prevents adding the new application, while a complete redesign can accommodate all applications without adding any new device.

4 INTEGRATED TOOL SUPPORT FOR DESIGN AUTOMATION

It is established in the last section that we can compute control and platform parameters for FlexRay-based systems in a correct-by-design manner. To apply the co-design technique for industrial-sized systems, it must be integrated with the commercial off-the-shelf (COTS) tools used in the automotive industry. However, this is not trivial as the tools used for controller and platform designs are separate and are products of different suppliers. These tools are developed based on years of domain-specific experience. It is challenging for one supplier to build a holistic framework and maintain the same quality across all domains. The key to go Agile in the real sense is to bring the tool suppliers together for an integrated framework. And it must be possible to maintain traceability automatically across all the tools involved in the development process.

In this context, we study an integrated toolchain [19] employing the control-platform co-design approach discussed in Sec. 3.1. Here, we call this toolchain as Co-Flex. Co-Flex is based on COTS tools for the design and development of FlexRay-based distributed automotive CPS. These COTS tools are MATLAB/Simulink for controller design and SIMTOOLS/SIMTARGET [27, 28] toolboxes (by Elektrobit) for platform configuration and software implementation.

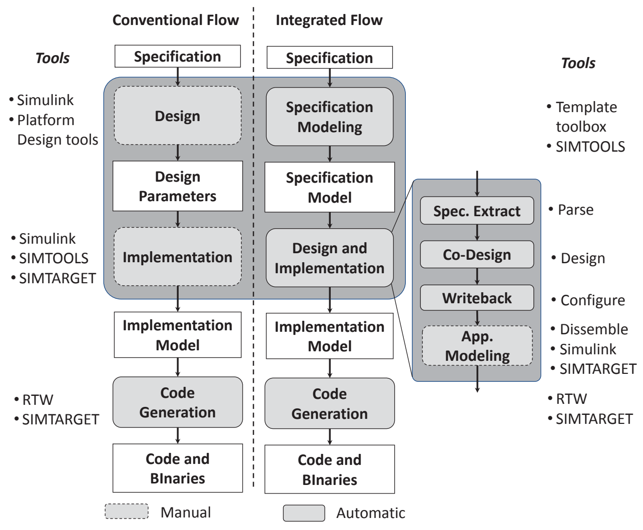

Figure 3: Integrated design flow and the tool support.

With the COTS tools, embedded control software is developed in three distinct phases as illustrated in Fig. 3. (i) In the *design phase*, the controllers and platform parameters are calculated based on specification. (ii) In the *software implementation phase*, the application software is modeled and configured using the parameters obtained from the design phase. (iii) In the *code generation phase*, codes and binary files are generated from the models which are then deployed on the hardware.

The available tools aid in certain steps of the whole flow as follows: (i) SIMTOOLS/SIMTARGET provide specific blocksets which enable modeling of FlexRay network and ECU, configuration of tasks and frames and their schedules, and definition of input and output interfaces. (ii) the Simulink Realtime Workshop and SIMTARGET can be used to generate C-code and binary files. Nevertheless, the design flow also involves manual processes which are tedious, time-consuming and error prone, e.g., formulation of the design problem, modeling of the application tasks and configuration of control and platform parameters.

Towards design automation, Co-Flex offers (i) a template toolbox that can be conveniently used to model automotive control applications through easy parametrization, and (ii) some specific tools to automate the flow between different phases, e.g., specification extraction from partially specified models, re-configuration of the control models with the obtained values of control parameters, and synthesizing the implementation model with correct platform parameters, i.e., task and message schedules.

Here, we study a customized automated tool flow as shown in Fig. 3. It starts with a specification modeling phase where the template blocks can be used to develop a specification model. This model has details of controlled plant dynamics, E/E architecture (ECU and network configuration), software details (tasks and messages) and QoC requirements. Based on the specification model, the design and implementation is carried out in 4 stages. First, a *Parse* tool is used to automatically extract the specifications from the model and store it in a systematic way. Second, a *Design* tool is invoked to co-design control and platform parameters according to the technique in Section 3.1 and generates a Pareto front. The systems engineer can select a Pareto point based on extensibility requirements. Third, the parameter values corresponding to the chosen Pareto point is configured on to the specification model using a *Configure* tool. Fourth, a *Dissemble* tool removes the specification models which were only needed for the design and are not parts of the implementation. In this stage, manual modeling of application-specific details may also be required. After this phase, code generation and hardware deployment are carried out in the same way as in the traditional flow.

With Co-Flex, it is also required to adapt the design and implementation flow. Nevertheless, the adapted flow still exploits the advantages of the existing toolchain, e.g., automated code generation and convenient modeling of ECUs and networks.

Note that Co-Flex only supports ECU software development for safety-critical systems mapped on one FlexRay bus cluster. However, there are other ECUs in the same cluster and their designs are strongly dependent on the results obtained from Co-Flex. The related design domains of wiring and hardware will be impacted as well. As discussed earlier, it is important to ensure traceability across all design tools. In this context, an interface between Co-Flex and Capital have been evaluated. It is possible to export the functional and architectural models specified in Capital as XML files. These files can be used to automatically configure the partially specified Co-Flex models. And the results from Co-Flex can be directly inserted into these XMLs which can then be imported back into Capital. This can ensure data consistency between Co-Flex and Capital. And the integrated framework of Capital can be exploited to maintain data coherency across design tools.

5 AN AUTOMOTIVE CASE STUDY

A case study of 5 control applications, $\mathcal{C} = \{C_1, C_2, C_3, C_4, C_5\}$, is considered. For each application, a plant model is derived from the automotive domain. C_1 to C_5 represent respectively the DC motor speed control (DCM), the car suspension system (CSS), the electronic wedge brake (EWB), and two variants of the cruise control (CC1) and (CC2). Each application C_i is composed of 3 tasks, i.e., sensor task ($\tau_{s,i}$), controller task ($\tau_{c,i}$) and actuator task ($\tau_{a,i}$). The hardware platform consists of 3 ECUs, $\mathcal{E} = \{E_1, E_2, E_3\}$, communicating over a FlexRay bus. The task mappings and the FlexRay bus parameters are given in Table 1.

ECU	Tasks
E_1	$\tau_{s,1}$, $\tau_{c,2}$, $\tau_{a,3}$, $\tau_{a,4}$, $\tau_{c,5}$
E_2	$\tau_{a,1}$, $\tau_{s,2}$, $\tau_{c,3}$, $\tau_{s,4}$, $\tau_{s,5}$
E_3	$\tau_{c,1}$, $\tau_{a,2}$, $\tau_{s,3}$, $\tau_{c,4}$, $\tau_{a,5}$

(a) Task mapping.

Bus Parameters	Values
Cycle length	5 ms
No. of staic slots	25
Slot size	100 μs

(b) FlexRay Bus Configuration.

Table 1: Case Study Configuration.

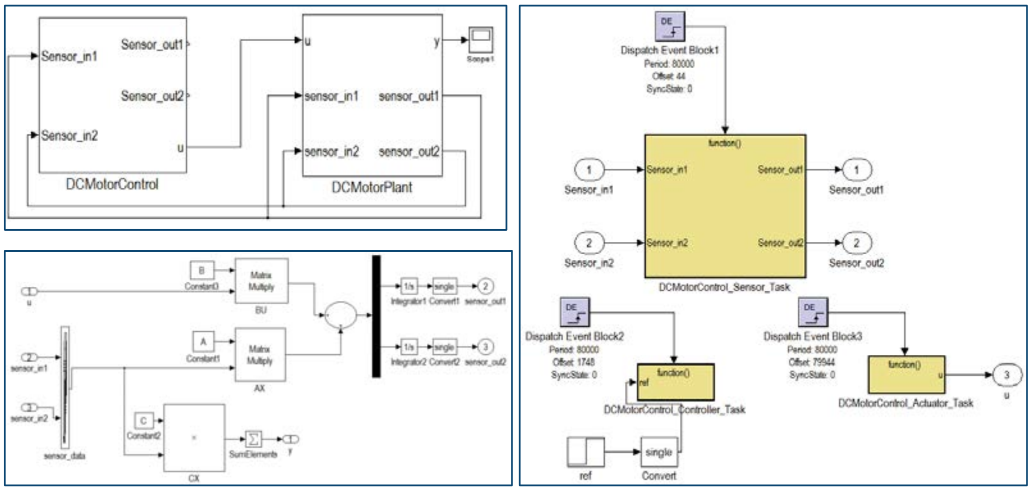

Figure 4: Partially specified model. (i) (left top) Closed-loop system model (ii) (left bottom) Plant model (iii) (right) Distributed controller implementation.

The tool flow discussed in Sec. 4 is used to develop the software model for the system. First, a specification model is built using SIMTOOLS and the offered template toolbox, as shown in Fig. 4. Specifications are then automatically extracted using the *Parse* tool based on which the co-design technique (Sec. 3.1) is applied.

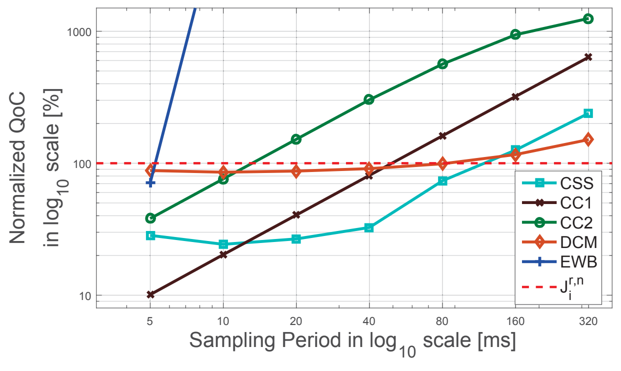

Figure 5: Normalized QoCs of predesigned controllers.

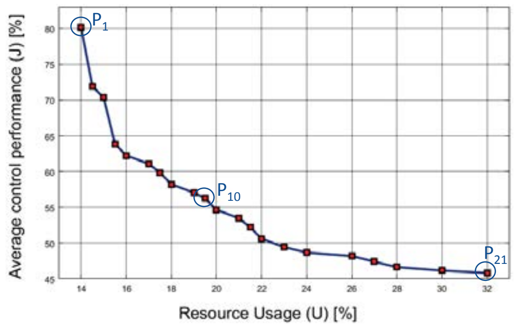

Figure 6: Pareto front for a case study.

The *Design* tool first determines optimal controllers at 7 possible sampling periods for each application. Fig. 5 shows the normalized QoCs of the designed optimal controllers for all the applications. The red dashed line represents 100%, i.e., the required minimum performance. Note that lower the value of QoC, better is the performance. Observe that not all predesigned controllers satisfy the minimum performance requirement. For example, in case of CC2, controllers, designed for sampling periods of 5ms and 10ms, only meet the requirement. Thus, only these sampling periods are feasible. This information can be used for design space pruning. Based on this result, the *Design* tool automatically formulates and solves the co-design problem. The Pareto front thus obtained is depicted in Fig. 6 and consists of 21 Pareto points. The values of U range from 14% to 32% and J varies between 45.82% and 80.14%. It is obvious that there is a large freedom among these viable designs. For larger system size we can expect more trade-off opportunities.

Three well-spaced Pareto points, i.e., P_1, P_{10} and P_{21}, as marked in Fig. 6, are implemented. The specification model is configured successively using the *Configure* tool with parameters corresponding to these Pareto points and tested for plausibility. These fully specified models are simulated according to 4^{th} level of simulation available in SIMTOOLS, where the ECUs and the communication system are simulated based on the timings of application tasks, communication tasks and bus schedules. The system responses are recorded for each of the simulation runs and are shown in Fig. 7.

Note that C_1, C_3, C_4 and C_5 are applied unit step references while C_2 is applied an unit impulse reference. In case of CC2, the system response for P_1 is different from the identical responses corresponding to P_{10} and P_{21}. This is because the synthesized sampling period of CC2 corresponding to P_{10} and P_{21} is the same and is equal to 5 ms while it is 10 ms for P_1. The responses of CC2 are identical for P_{10} and P_{21} despite different task and message schedules. This verifies that in the optimization stage, QoC depends only on the sampling period of a controller. Furthermore, we can observe that the settling times of the system responses of CC1 are respectively 52 ms, 212 ms and 413 ms corresponding to sampling periods of 5 ms, 20 ms and 40 ms and are approximately in the ratio 1:4:8. This can be verified from Fig. 5. Similarly, QoCs of all applications are verified against the calculated values from the simulation results.

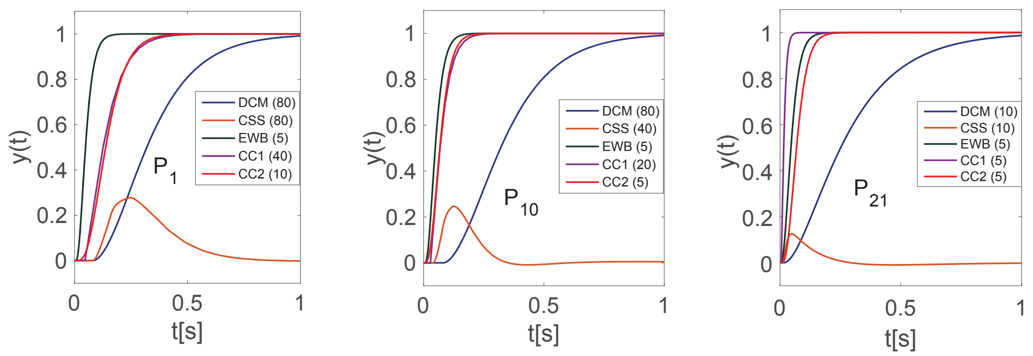

Figure 7: Control responses corresponding to the Pareto points P_1, P_{10} and P_{21} (Sampling period values in ms are given in brackets).

After software simulation, the *Dissemble* tool is used to automatically remove the continuous-time plant models and expand the subsystem blocks. Subsequently, we generated C-code and binary files successfully using SIMTOOLS *Split and Build*, Simulink RTW and SIMTARGET. The binaries can be flashed onto the ECUs.

6 CONCLUSION

Through this paper, we suggest Agile development in the automotive industry. To go Agile, we need to have fast yet safe development cycles for automotive systems. Towards this, while industry is adopting a data-driven rule-based design approach, academic researchers are introducing novel techniques to design safe yet optimal systems. However, a bridge between the two developments is largely missing. We have shown using a case study that it is possible to integrate a state-of-the-art design technique into an industrial toolchain. In the future, we believe different tool suppliers and academia would come together to develop COTS integrated toolchain for high quality future automotive CPSs.

ACKNOWLEDGMENT

This work was partially supported by Deutsche Forschungsgemeinschaft (DFG) through the TUM International Graduate School of Science and Engineering (IGSSE).

REFERENCES

[1] ISO 26262. 2011. Road vehicles – Functional safety. https://www.iso.org/standards.html

[2] A. Aminifar, P. Eles, Z. Peng, and A. Cervin. 2013. Control-quality driven design of cyber-physical systems with robustness guarantees. In *Design, Automation & Test in Europe Conference & Exhibition (DATE).*

[3] M. Balszun, D. Roy, L. Zhang, W. Chang, and S. Chakraborty. 2017. Effectively utilizing elastic resources in networked control systems. In *23rd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).*

[4] W. Chang, D. Roy, L. Zhang, and S. Chakraborty. 2016. Model-based design of resource-efficient automotive control software. In *35th International Conference on Computer-Aided Design (ICCAD).*

[5] W. Chang, L. Zhang, D. Roy, and S. Chakraborty. 2017. *Control/architecture codesign for cyber-physical systems.* Springer Netherlands.

[6] U. Eliasson and H. Burden. 2013. Extending Agile Practices in Automotive MDE. In *Extreme Modeling Workshop (XM).*

[7] FlexRay Consortium. 2010. The FlexRay communications system protocol specification, Version 3.0.1. Retrieved July 27, 2018 from https://svn.ipd.kit.edu/nlrp/public/FlexRay/

[8] G. M. Mancuso, E. Bini, and G. Pannocchia. 2014. Optimal Priority Assignment to Control Tasks. *ACM Transactions on Embedded Computing Systems (TECS)* 13, 5s (2014), 161:1–161:17.

[9] D. Goswami, A. Masrur, R. Schneider, C. J. Xue, and S. Chakraborty. 2013. Multi-rate controller design for resource- and schedule-constrained automotive ECUs. In *Design, Automation & Test in Europe Conference & Exhibition (DATE).*

[10] D. Goswami, R. Schneider, and S. Chakraborty. 2011. Re-engineering cyber-physical control applications for hybrid communication protocols. In *Design, Automation & Test in Europe Conference & Exhibition (DATE).*

[11] T. Heurung. 2015. Bridging Automotive Design Domains with the Latest in Functional Design Technology. In *SAE Technical Paper.*

[12] T. Heurung and S. Walz. 2008. White Paper: Designing and implementing architectures for distributed automotive E/E systems. Retrieved August 13, 2018 from http://go.mentor.com/4gtgc

[13] B. Katumba and E. Knauss. 2014. Agile Development in Automotive Software Development: Challenges and Opportunities. In *International Conference on Product-Focused Software Process Improvement (PROFES).*

[14] L. Ma, F. Xia, and Z. Peng. 2008. Integrated Design and Implementation of Embedded Control Systems with Scilab. *Sensors* 8, 9 (2008), 5501–5515.

[15] H.-J. Mantsch. 2015. White Paper: A holistic approach to vehicle system design. Retrieved August 13, 2018 from http://go.mentor.com/4ness

[16] Mentor, A Siemens Business. 2017. Capital. Retrieved August 10, 2018 from https://www.mentor.com/products/electrical-design-software/capital/

[17] Mentor, A Siemens Business. 2018. Capital: An Integrated Electrical Engineering Environment. Retrieved August 13, 2018 from http://s3.mentor.com/public_documents/datasheet/products/electrical-design-software/capital/chs_overview.pdf

[18] B. Morris. 2018. White Paper: Implementing SAE J1939 in commercial, off-highway & heavy-duty vehicle design. Retrieved August 13, 2018 from http://go.mentor.com/4ZLAH

[19] D. Roy, M. Balszun, T. Heurung, and S. Chakraborty. 2018. Multi-Domain Coupling for Automated Synthesis of Distributed Cyber-Physical Systems. In *International Symposium on Circuits and Systems (ISCAS).*

[20] D. Roy, L. Zhang, W. Chang, and S. Chakraborty. 2016. Automated synthesis of cyber-physical systems from joint controller/architecture specifications. In *Forum on Specification and Design Languages (FDL).*

[21] D. Roy, L. Zhang, W. Chang, D. Goswami, and S. Chakraborty. 2016. Multi-objective co-optimization of FlexRay-based distributed control systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS).*

[22] D. Roy, L. Zhang, W. Chang, S. Mitter, and S. Chakraborty. 2017. Semantics-Preserving Cosynthesis of Cyber-Physical Systems. *Proceedings of the IEEE* 106, 1 (2017), 171 – 200.

[23] W. W. Royce. 1987. Managing the development of large software systems: concepts and techniques. In *9th International Conference on Software Engineering (ICSE).*

[24] S. Samii, A. Cervin, P. Eles, and Z. Peng. 2009. Integrated scheduling and synthesis of control applications on distributed embedded systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE).*

[25] R. Schneider, D. Goswami, A. Masrur, M. Becker, and S. Chakraborty. 2013. Multi-layered scheduling of mixed-criticality cyber-physical systems. *Journal of Systems Architecture - Embedded Systems Design* 59, 10-D (2013), 1215–1230.

[26] R. Schneider, D. Goswami, S. Zafar, M. Lukasiewycz, and S. Chakraborty. 2011. Constraint-driven synthesis and tool-support for FlexRay-based automotive control systems. In *7th International Conference on Hardware/Software Codesign and System Synthesis.*

[27] SIMTOOLS GmbH. 2010. SIMTARGET V5.0.1: C code generation from SIMTOOLS models for CAN, FlexRay, and I/O.

[28] SIMTOOLS GmbH. 2012. SIMTOOLS V5.2.0: Model-Based Design Tools for FlexRay-based Applications.

[29] J. Sztipanovits, T. Bapty, S. Neema, X. Koutsoukos, and E. Jackson. 2015. Design tool chain for cyber-physical systems: Lessons learned. In *52nd Design Automation Conference (DAC).*

[30] H. Voit, R. Schneider, D. Goswami, A. Annaswamy, and S. Chakraborty. 2010. Optimizing Hierarchical Schedules for Improved Control Performance. In *5th IEEE International Symposium on Industrial Embedded Systems (SIES).*