DEVELOPMENT AND EVALUATION OF COLLABORATIVE EMBEDDED SYSTEMS USING SIMULATION.

THARABAI C. V.
Lecturer in Electronics
Women's Polytechnic College Ernakulam, Kochi, Kerala.

Abstract:
Embedded systems are increasingly equipped with open interfaces that enable communication and collaboration with other embedded systems, thus forming collaborative embedded systems (CESs). This new class of embedded systems, capable of collaborating with each other, is planned at design time and forms collaborative system groups (CSGs) at runtime.Traditionally, early-stage validation and verification (V&V) of systems composed of collaborative subsystems is performed by function integration at design time. Simulation is used at this stage to verify system's behaviour in a predefined set of test scenarios.

Keywords:
Simulation, Collaborative Embedded Systems, Automotive, CES, Validation and Verification, Modelling and Simulation,

Introduction:

Modeling and simulation are established scientific and industrial methods to support system designers, system architects, engineers, and operators of several disciplines in their work during the system life cycle. Simulation methods can be used to address the specific challenges that arise with the development and operation of collaborative embedded systems (CESs). In particular, the evaluation of collaborative system behavior in multiple, complex contexts, most of them unknown at design time, can benefit from simulation. In this paper, after a short motivation, we exemplify scenarios where simulation methods can support the design and the operation of CESs and we summarize specific simulation challenges. We then describe some core simulation techniques that form the basis for further enhancements addressed in the individual paper. [1]

Simulation is a technique that aids in the overall design, evaluation, and reliable operation of systems. CESs are a subset of embedded systems that, while designed and developed independently, can form collaborations to achieve collaborative goals during runtime. This new class of systems faces unique design and development challenges that simulation methods can address.

Following the traditional approach of performing early-stage validation and verification (V&V) via simulations, we provide a survey of existing simulation approaches and investigate whether and how simulation approaches should be extended to better address the V&V challenges brought by CES in this paper. We concentrate on simulation methods, which are systematic procedures involving simulations for V&V purposes, and tools, which are technical frameworks and environments that support the simulation methods. [2]

SimulationforVerificationandValidation

Software functions are tested at various levels. Unit tests, integration tests, system tests, and operational acceptance tests are performed on the software. The requirements for individual test levels are increasing as the software's complexity grows. The behaviour of complex systems can be virtually mapped using simulations, allowing for a quick, inexpensive, and efficient analysis of the system. All test levels benefit from the use of simulations.

The use of simulation methods for behavioural evaluation of software systems and system components has numerous advantages, regardless of domain:

- Simulation methods are more exploratory than analytical methods when given a set of concrete scenarios of complex interactions.

- Statistical analysis reveals that an estimated Because simulation models are digital entities, it is possible to automate and parallelize the process of running multiple simulation scenarios.
- Systems of systems that did not exist before are formed by coupling components or systems. As a result, there may be insufficient knowledge and experience with such systems. Simulation of component behaviour can reveal the effects of function interaction for such a system.
- Standards such as ISO 26262[3] explicitly recommend simulation as a quality assurance technique in the automotive domain.

CharacteristicsofCollaborativeEmbeddedSystems

The term Embedded System (ES) is defined by Schlingloff as "a computational system that is a fixed component of a technical system." ESs continuously process input and can provide real-time output; their functionality is limited by the technical context in which they operate. ESs can supervise, control, or acquire data in a fixed context depending on the programmed functionality and context of operation.

Simulation tools and methods can be used to evaluate embedded systems in a variety of contexts. Before delving into how CES are evaluated, we highlight the key differences between behavioural CES evaluation and single embedded system evaluation, and we conclude with a set of CES characteristics that must be verified and validated through simulation. [4-6]

1. New behaviour can emerge when systems meet and form collaborations in real time. This behaviour requires simulation and coupling to real-world scenarios such as road traffic. Additionally, visualisation must be performed intuitively in order to facilitate reasoning about the effects of emergent behaviour.

2. When performing simulation-based evaluation of an embedded system, the system's goals are implicitly considered. Multiple CES in a CSG setting may have different, even conflicting goals. As a result, analysing each CES's goals and their contribution to the overall ecosystem goal (e.g. forming a platoon) is an important first step in evaluating their behaviour. As a result, simulation methods and tools should be designed to support the modelling and analysis of system goals and their relationship to simulation outcomes.

3. In a collaboration, an embedded system learns about its surroundings through the exchange of functions and data. For example, in a vehicle, an embedded system running a control function that processes environmental information will make decisions based on information received from other embedded systems in another vehicle. A set of messages is used to pass the function of sensing the environment to a collaborator. This raises a slew of questions about the accuracy and timeliness of information sensed, interpreted, and delivered by a remote vehicle. As a result, when evaluating a collaboration, uncertainties and their effects must be simulated. One method is to introduce flaws such as delayed communication.

4. Because CSG are open-ended and dynamic systems (the number and potentially collaborators or type of collaborators may change over time as the system runs), it is extremely difficult to test for all possible scenarios at runtime. In addition to design-time simulation, simulation methods and methods after system deployment can be used for runtime validation. As a result, CSG simulation methods would need to be usable at runtime as well.

5. Technically, Embedded Systems in collaboration can communicate over the same shared memory, a bus connection, or the Internet via 5G wireless communication. In terms of communication, ESs can communicate with one another using an event-based protocol that makes efficient use of network resources. This means that information processing is triggered by actions triggered by user input, alarms, or data inputs from other systems. Simulation of communication channels is required in order to perform CES and CSG evaluation.

6. Embedded systems interact with their physical surroundings in order to achieve the system's goals. This implies that, at the level of system analysis, simulation of the physical environment must be taken into account when evaluating CESs.

7. The evaluation of CES and CSG necessitates the coupling of simulation models developed independently. The simulation models can be subsystems or systems communicating with one another, or they can be network communication simulations. Co-simulation platforms are required to assess emergent system behaviour.
8. All of the preceding points demonstrate the complexities of simulating embedded systems that collaborate with one another to provide required services to other systems or end-users. The methodology we used to find answers to the challenges mentioned in this section is presented below.

Review Of Literature:

[Erden et al. 2008] conducted a comparative literature review to investigate different function modelling approaches and their similarities and differences. Functional model ontologies, for example [Chandrasekaran and Josephson 2000], [Umeda et al. 1996], [Umeda et al. 1995], and [Yoshioka et al. 2004] aim to develop frameworks and languages for modelling system functionality from various perspectives [Erden et al. 2008]. None of the proposed functional model ontologies take into account the modelling of complex CES and CSG functions, including the implications of the contexts in which these systems operate. [7]
The authors define two function viewpoints in [Chandrasekaran and Josephson 2000]: "environment-centric viewpoint" and "devicecentric viewpoint." These perspectives correspond to the collaboration and system functions proposed in our work. In the first case, function refers to the external effects that an object or system has on its surroundings. In contrast, in the second viewpoint, functions are associated with the system's internal features and parameters. To some extent, our metamodel incorporates both of the perspectives proposed in [Chandrasekaran and Josephson 2000]: 1) taking into account the functions of systems that have an impact on their environments, specifically the context as the relevant part of the environment in our case; 2) as well as the other systems involved in a collaboration, including their internal system parameters, states, and behaviours. In addition, [Gero 1990] has created a function-behavior-structure model. In his model, a function was viewed as an intermediate step between the system's behaviour and the user's goal. [8] A few frameworks have been proposed in the literature to systematically define a well-formed functional behaviour of the system. FOCUS (cf. [Broy and Stlen 2012], and [Broy 2014]), as previously described in Section 4.3, is an example of a formal framework that provides models and formalisms for the specification and development of distributed interactive and dynamic systems. According to FOCUS, we define a function's behaviour as a stream of messages across its input and output channels, which take up information, material, or energy through its interfaces and transform it before outputting it.

Objectives:

1. Development And Evaluation of Collaborative Embedded Systems Using Simulation.
2. Simulation for Verification and Validation.
3. Define Characteristics of Collaborative Embedded Systems.
4. Overview of collaboration in order-driven production.

Research Methodology:

Close integration of embedded software with physical system processes is possible in this research simulation for single embedded systems. As a result, in a mixed HW/SW simulation environment, effective virtual prototyping of embedded software can be realised. The authors of [9] present an embedded SW simulation with a focus on hardware-dependent software refinements, realised with a toolset that integrates open tools and libraries based on C/C++. Open Dynamics Engine (ODE) can also be used to simulate physical rigid body movement and dynamics. System C/System C-AMS are used to simulate mixed digital/analogue systems in conjunction with the open source software emulator QEMU [10], which provides instruction and register accu rate CPU abstraction of multiple instruction set platforms and is used for fast software emulation. Based on this, [11] employs a simulation environment that can also embed additional simulators in addition to System C/System C-AMS. All of these tools provide open Matlab/Simulink interfaces through which additional design flow of physical components can be modelled.

Result And Discussion:

Order-driven production is a scenario that is frequently described in this context, in which the CESs involved in the production of a product (also known as modules in the factory) form a CSG (also known as production network in the factory) based on the requirements of the product to be manufactured and with the goal of manufacturing the product. The application of the concept described in results in the formation of a production network that is formed to produce a specific production order and then dissolves after completion. [12]

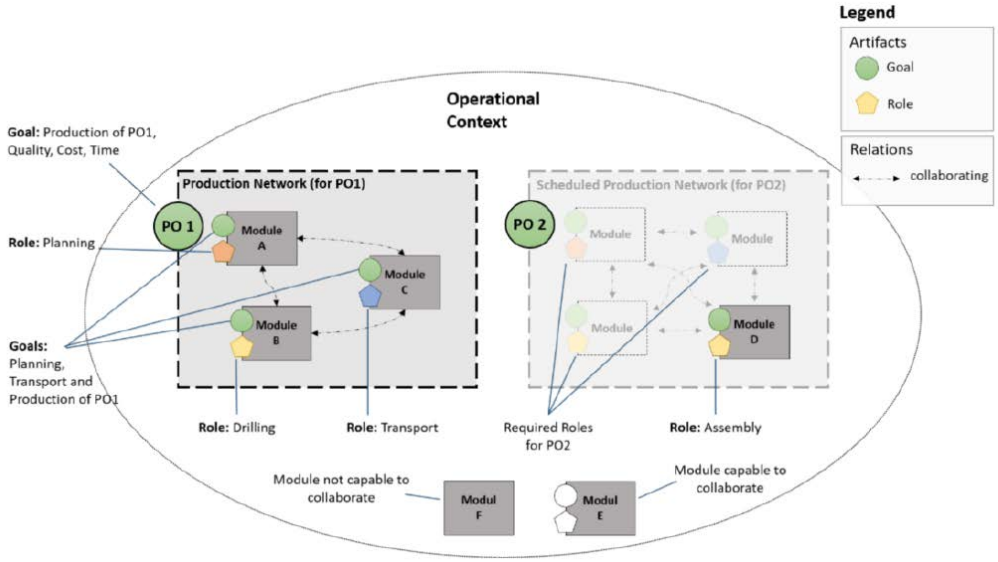

Figure 1: Overview of collaboration in order-driven production

In the case of the adaptable and flexible factory, the interaction of the CESs in the CSG must also be considered and described using appropriate models in order to serve as a foundation for CES development and collaboration during operation. Figure 1 depicts an existing and planned production network for the processing of two orders. [13]

When a CES enters the CSG, for example, its internal structure changes. The operational context of both the entering CES and the CSG changes at the same time. For systems in context, we distinguish between collaborative systems—that is, systems that can enter a CSG due to their architecture and functions—and non-collaborative systems.

CSG at any time.

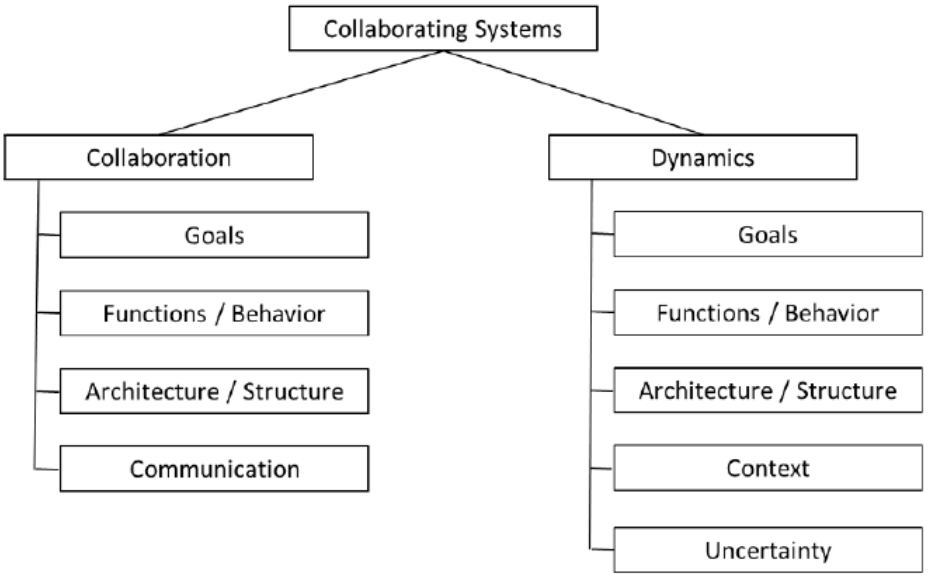

Figure 2: Taxonomy of CrESt challenges

A taxonomy of challenges for collaborative embedded systems can be defined based on these specific challenges, as shown in Figure 2. A number of characteristics were defined for the two superordinate categories "Collaboration" and "Dynamics." [14]

Conclusion:

Collaboration introduces new challenges in the model-based development of embedded systems, necessitating the adaptation and extension of existing modelling languages. In this chapter, we discussed the factors to consider when modelling functions for CESs and CSGs in a metamodel. We then evaluated and illustrated this metamodel using two examples from the adaptable and flexible factory and autonomous transport robots use cases. Specific extensions of modelling languages can be executed based on the metamodel. Methods for applying these extended modelling languages can be developed based on domain-specific requirements. The use case examples presented in this chapter will serve as the foundation for future research.

References:

1. Tuomas W. Sandholm, "Distributed rational decision making" in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Cambridge, MA, USA:The MIT Press, pp. 201-258, 1999
2. M.B. Dias, R. Zlot, N. Kalra and A. Stentz, "Market-based multirobot coordination: A survey and analysis", Proceedings of the IEEE, vol. 94, no. 7, pp. 1257-1270, 2006.
3. Sven Koenig, Pinar Keskinocak and A. Craig, "Tovey. Progress on agent coordination with cooperative auctions", Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence AAAI 2010, July 11–15, 2010
4. M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, S. Koenig, A. Kleywegt, et al., "Auction-based multi-robot routing", Proceedings of the InternationalConference on Robotics: Science and Systems, pp. 343-350, 2005.
5. Burrill, G. (2002). Simulation as a tool to develop statistical understanding. Paper Presented at the International Conference on Teaching Statistics, 2002.
6. Gnanadesikan, M., Scheaffer, J. L., & Swift, J. (1987). The Art and Techniques of Simulation. Quantitative Literacy Series.
7. M.S. Erden, H. Komoto, T. J. van Beek, V. D'Amelio, E. Echavarria, T. Tomiyama: A Review of Function Modeling: Approaches and Applications. Ai Edam 22, no. 2, 2008, pp. 147-169.
8. B. Chandrasekaran, John R. Josephson: Function in Device Representation. In: Engineering with Computers 16.3-4: pp.162-177, 2000.
9. M. Broy, K. Stølen: Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement. Springer Science & Business Media, 2012.
10. Mueller, W., Becker, M., Elfeky, A., DiPasquale, A.: Vir- tual prototyping of cyber-physical systems. In: Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific, pp. 219–226. IEEE (2012)
11. Ilieskou, N., Blom, M., Somers, L., Reniers, M., Bas- ten, T.: Multi-domain virtual prototyping in a systemc sil framework: A heating system case study. In: 2015 In-ternational Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), pp. 286–294 (2015)
12. Grosz: Collaborative Systems. In: AI Magazine, vol 17, no 2, 1996.
13. K. Pohl, H. Hönninger, R. Achatz, M. Broy (Eds.): Model-Based Engineering of Embedded Systems: The SPES2020 Methodology, Springer, Heidelberg/New York, 2012
14. Mark W. Maier: Architecting Principles for Systems-of-Systems. In: Systems Engineering 1(4), 1998, pp. 267-284

.