Why and Where Do We Need Model Execution? (Keynote)

Francis Bordeleau
Ericsson Canada, Town of Mount Royal, Canada
francis.bordeleau@ericsson.com

***Index Terms*—MBE, model execution**

I. Extended Abstract

Model execution is a technique used to validate/test/verify an application, or system, at the model level before an implementation is developed. It has the potential to bring major benefits in many different development contexts and thus strongly contribute to the broader adoption of Model Based Engineering (MBE). Key potential benefits include reduced development time, cost, and risks, and improved overall system quality by enabling execution earlier and more often in the development process. However, not all models can be executed. Model execution requires the use of a modeling language that is based on a precise execution semantics, including behavior specification, and the existence of an execution platform to enable execution of models developed using this language.

Model execution has proven to be very successful in many application domains (aerospace, automotive, industrial automation) to simulate and validate functional behavior with tools like MathWorks Simulink. However, the goals and requirements for model execution are quite different depending on the modeling context. Here, we use software design and system engineering to illustrate some of the main differences between different modeling contexts. Similar analysis can also be done for other contexts like requirements modeling, information modeling, functional modeling, business process modeling, network architecture modeling, and enterprise architecture modeling.

In the context of software design, where modeling is used to design and visualize applications/systems at an abstract level (i.e. at a level of abstraction higher than programming languages), model execution is viewed as a technique that can be used to execute models early in the development process to validate/test/verify an application/system before a detailed design is developed. This allows making abstraction of the specifics of programming languages and deployment platforms, and focus on the design of a platform independent model (PIM) that can later be implemented using different programming languages and deployed on specific platforms to satisfy the needs of different business cases. Model transformation and code generation techniques can then be used to transform the PIM into an implementation. The use of such approach allows increasing portability and enables performance optimization by improving model transformations, code generators, and allocation strategies, instead of by directly modifying the design or implementation.

However, while the development of PIM can provide key benefits in many different contexts, it also imposes important requirements/constraints on modeling tools and their users. One of the main requirements is the need to provide/use an action language, instead of a programming language, to specify the behavior of the components. Many users consider this a main obstacle, as it requires learning a new language and associated integrated development environment (IDE). Moreover, tooling for existing action languages do not provide the level of maturity and functionality that users are used to with popular programming language IDEs, in particular regarding aspects like runtime monitoring and tracing & debug capabilities. Also, while this type of MBE workflow has been a core part of the MBE vision since its inception, very few modeling tools today provide an action language to support model execution. There are several reasons for this, which relate to both technical and business aspects, but these are beyond the scope of this presentation.

Another approach to support model execution consists in integrating modeling techniques and programming languages in a single development environment. In many development contexts, the price to pay for the development of a PIM is perceived to be too expensive for the benefits it provides. For example, in cases where an application is developed for a single programming language, and no business case exists for the development of the application using a different programming language, there is no clear benefit in investing time and effort to develop a PIM. In such context, what users really need is a better integration between the modeling techniques and programming language IDEs to improve the inner design loop and allow for a faster and more agile development process.

In the context of system engineering, where the end goal is not to generate code but rather to develop systems that result from the interaction of a set of heterogeneous components, which can each be implemented/realized using different types of technology, not only software, it is quite different. In this case, modeling is used to design, communicate, and validate the system to people of different background at different stages of the development process, and model execution is used to validate/test/verify the system before it is implemented. In order to be able to execute the model, the behavior of the different components needs to be specified. Since people

involved in system engineering are from different background, and typically not software engineers, it is much easier and natural to introduce a higher-level action language than a programming language, which is usually more complex to learn. So, in this context, the use of an action language doesn't constitute a main issue.

As illustrated by these two different contexts, software design and system engineering, the needs and requirements for model execution are quite different depending on the modeling context. The background and culture of the people involved in the modeling process, which significantly differs between different application domains, also have a strong influence on the type of tooling solution that needs to be developed. Therefore, in order to provide proper modeling tools to support model execution, it is essential to clearly understand the role model execution plays in the overall development process and the type of language and functionality that need to be provided.

The goal of this keynote presentation is to provide an industrial perspective on model execution. We discuss different modeling contexts (information modeling, system modeling, network architecture modeling, and software design) and application domains (telecommunication, Software Defined Radios, cyber-physical systems), and discuss their needs (or not) and requirements regarding model execution. We also discuss what is needed, both from a technical and business perspective, to have first-class support for model execution in modeling tools.

II. Biography

Francis Bordeleau is Product Manager in the Software Development group at Ericsson. His main areas of responsibilities include model-based engineering and modeling tools. In this role, he is responsible for defining product specifications and roadmaps, developing business cases, managing budget, managing open source initiatives, and collaborating with other companies, researchers, and academia.

Francis has over 20 years of experience in MBE and software engineering; researching, working, consulting, and collaborating with numerous companies worldwide. Prior to joining Ericsson in May 2013, Francis was the Founder and CEO of Zeligsoft, a provider of domain specific Model Based Engineering (MBE) tooling solutions for distributed real-time embedded systems. He was also Director of Tooling Business for PrismTech. Prior to found Zeligsoft, he was an Assistant Professor at the School of Computer Science at Carleton University.

Francis holds a B.Sc. in Mathematics from the University of Montreal, a Bachelor of Computer Science from the University of Quebec (UQO), and a Master in Computer Science and Ph.D. in Electrical Engineering from Carleton University.