Success Story: Model-Based Testing at Verified Systems International

Software testing is intrinsically model-based: one needs to tell apart failed and passed test-cases and this calls for a model. Models can further be used to drive test generation and test selection. Model-based testing is a mature field of research dating back to the early work of pioneers such as Edward F. Moore. We have seen several successes in the application of model-based testing to large-scale industrial systems. However, its application to contemporary software and systems is still challenged by the complexity and heterogeneity of such systems.

In this edition of Success Stories, we interview professor Jan Peleska, who took the leap to apply the wealth of research, partly done by himself, into model-based testing of large-scale industrial systems.

Professor Peleska is a professor of computer science at Bremen University in Germany and the co-founder of Verified Systems International GmbH, a company specialised in Validation and Verification services and tools. Under his leadership, we have seen several successful examples of the applications and commercialisation of model-based testing and in this interview, we intend to review these and also focus on the challenges ahead of using model-based testing.

How would you present yourself?

My name is Jan Peleska. I am a Professor at the University of Bremen. I focused my research on Safety critical control systems, verification, validation, testing, and modelling.

In 1998, Cornelia Zahlten (my wife) and I co-founded a company called Verified Systems International specialised in Verification and Validation (V&V) for Safety Critical systems. We now have 25 employees, and we are doing relatively well. Our customer-base includes leading companies such as Siemens or Airbus. We also provide V&V equipment, and hardware for in-the-loop test benches. Our main product is RT-Tester, which consolidates conventional and model-based testing.

Can you define Model-Based Testing in the context of your work?

This is not a trivial question; there are many approaches to Model-Based Testing. In the context of our company, Model-Based Testing works as follows. We analyse the expected behaviour observed of the system under test according to models. From requirements, the product automatically generates tests to cover all cases. By means of temporal logic and formal behavioural specifications, we create concrete test data. The same product, RT-Tester, allows for testing on several levels, from hardware to software. We provide conventional testing and alternatively model-based testing – with an extended license.

Model-Driven Engineering and Model-Based Testing are very powerful. Yet, they do not seem to be as widely adopted as one would expect. How would you explain this?

There is a strong and large international research community around model-based testing. ***However, on the academic side, test automation is not so en vogue. Everybody is talking about AI.***

Model-Based Testing is not an easy sell for industrial applications and there are several explanations for that. Writing good models in a well-defined language is hard. In most companies, system requirements remain textual. Even some well-known aerospace companies failed to introduce Model-Based Testing. ***It seems hard to introduce model-based approaches to large companies.*** There are always some enthusiasts, yet it is still challenging. I believe one of the main reasons is the lack of appropriate modelling languages. For example, SysML is imperfect. The tool support is still weak and expensive.

Programming Integrated Development Environments are much more advanced than modelling Tools. There is a lot of investment in IDEs because it reaches a large community. Companies are not fully satisfied economically by Model-Based Development, that's why there is less investment in building relevant tools. I am waiting to see what will happen to SysML version 2. It will provide a textual and graphical programming language. ***Yet, I think a major issue is its very large syntax.*** SysML has over 400 pages of language description! It makes it very hard to build efficient tools. VSCode supports SysML extension but without semantic checks. For that reason, the code generator is still missing.

I believe Model-Based Programming requires a strong academic base. ***I would say you meet more skilled programmers than skilled modellers.*** It is on a higher level of abstraction. Developers are more willing to learn coding languages, that have direct applications, than modelling that seems too complicated.

It is interesting to note that despite all these challenges you did manage to commercialise this technology and use it with large multi-national companies. Can you tell me about the main challenges you faced to create your product?

After I finished my Ph.D. in mathematics, I went into the industry for a few years. I worked for Phillips, in the control system domain and then as a freelance consultant. In 1995, I decided to go back to University. In 1998, *Cornelia Zahlten* and I created the company. She was the managing director - there are two more managers now. ***Ever since, I worked at the University and for the company simultaneously.***

To go back to the creation of the company, the first version of the product used Communication Sequential Processes (CSP) algebra. Unfortunately, it was not very well welcomed by the customers. They found it too hard to learn. ***Therefore, we had to translate CSP processes to SysML, which was a very challenging task.*** The first major challenge was to understand SysML semantics. Then, we had to create real-time transition relations between the model and the background. It was very hard. From there, we could apply known technics to translate, such as bounded model-checking and tests from witnesses. Every model needs to be semantics at some level. ***The transition from CSP to SysML took approximately 5 years.*** It was supported by research projects and customers.

Airbus hired us as a service. They needed modelling as an effective way to review what was tested. It is a powerful communication tool. We managed to make the requirements traced automatically. We worked on certification: from the requirements, the tool created tests and displayed results automatically. ***They were satisfied with the results, but not sufficiently to build their team of experts.*** Let's say the product did not sell surprisingly well. Also, they always have more pressing things to do, like designing the next generation of planes… And quite a few employees did not want to be qualified for MBSE.

What are the main challenges for MBSE?

I think there is a communication problem between academia and the industry. From academia, we only produce small, well-defined languages like process algebras, which are not appropriate for industry. An industrial-strength modelling language is still missing. It must be more lightweight than SysML but cover a wider spectrum of users than formal languages from academia. ***We know how to define well-design programming languages, but designing a perfect model language is still an open question.***

What is your vision for your company, and Model Based Testing in general?

We are adopting a new strategy for the company. In the future, customers will only write formal requirements. The software will take formalised requirements and generate test models. We will use a much smaller modelling language just to learn models from the requirements and testing in the background. Then, we need to prove the models generated fulfil requirements. ***Apparently nobody likes complex models!***

At the System level, we will adopt the same approach. The software models end-to-end system requirements formalises them and generates all the relevant end-to-end tests. ***Future Model-Based Software Engineering should effectively produce smaller models representing requirements.***

Requirements modelling and automatic model learning are to me things of the future. We are consulting companies to help them introduce this new approach.

In the next 2 years, I believe we can reach automatic requirement modelling. We can produce graphic and textual supports with RT Tester products. People did not like having a case tool and test tool separately. ***We will combine programming, modelling, and testing in one product.***

We will not use SysML version 2. We were disappointed with the applicability of the traditional model-based approach. It was not a huge success. Smaller models are easier to understand and easier to build. For the next 4 to 5 years, we will stick to model learning. We aim to go from model testing to system testing. Generating end-to-end tests is hard. Automatically generating meaningful system tests from requirements is even more complex. ***I think we will need a combination of requirements and expert knowledge on how the system should work***. Systems become more and more complex, exponentially in fact. 10 years from now, I think it will be completely impossible to set up comprehensive models for the whole system because of this complexity. I foresee scenario modelling will play a major role. We can generate a set of scenario models. Then of course we need to prove it is a complete set to tell you what your system does. I know people from Siemens that already tell me they cannot write comprehensive specifications because systems have become too complex. Simulations will also play a major role. Statistical

approaches can calculate that the residual probability that we forgot a model is insignificant. I did not invent this, unfortunately!

This is my vision, I may be wrong of course! Anyway, It is a very exciting time for model-driven engineering. We will see what happens!

Jan Peleska, professor for computer science (operating systems and distributed systems) at Bremen University, interviewed by Avner Bensoussan.