MDENet – the home of model-driven engineering

Newsletter #1

In this issue:

Welcome from the network director

Welcome to this first issue of the MDENet newsletter and thank you for your interest in the network. Software is now at the heart of everything we do, and the problems addressed by modern software are growing increasingly complex—from managing complete product life cycles to environmental sustainability or an increasing demand for efficient and targeted healthcare for an aging population, we will continue to need ever larger and more complex software to support the growing challenges we face as a society. As we emerge from the COVID pandemic, this need for software will only increase, with some observers predicting digital to become the key driver of economic recovery. Not only does the complexity of our software requirements increase, we are also facing a huge gap between the need for software developers and what our education system can supply. As a network, we strongly believe that model-driven engineering (MDE) is a fundamental technology to help us address these big challenges. This is reflected in our vision statement:

> MDENet works towards a future where modelling technology ensures each decision about software is made by the most appropriate stakeholder.

So, what is MDE and why is it such a fundamental technology? MDE focuses on the idea that software should not be developed, as is traditionally the case, using fairly low-level, general-purpose programming languages such as JavaScript, C++, C#, Java etc. Developing software in these languages forces software engineers to constantly consider multiple competing concerns: they must capture the essence of the business logic while also ensuring scalability and security, and building software that can be effectively maintained and extended as requirements change over time.

Instead, MDE advocates the use of higher-level modelling languages, often highly specialised for particular problem domains or stakeholders. This enables true co-creation and collaboration of experts with a wide range of different backgrounds: subject-matter experts and business analysts are provided with languages and tools enabling them to capture the essence of their domain, software engineers can contribute their expertise in scalability and security using dedicated modelling languages of their own and user-experience designers can integrate their designs using notations best suited for these challenges. MDE complements these *domain-specific modelling languages (DSMLs)* with automation and transformation technologies that enable the integration of such different viewpoints and perspectives into a coherent software solution. Because DSMLs directly capture domain knowledge and domain rationale, they also offer opportunities for more powerful validation and analysis—enabling errors to be identified earlier in the development process. Overall, MDE has the strong potential to increase efficiency of software development, reduce error rates and manage complexity.

Of course, there is no free lunch and we still have a long way to go to achieve our vision for MDENet. A lot of technologies and software tools have been developed by the academic community and some industrial players, but more needs to be done to increase the wider uptake in the software development community

– and beyond in domains as diverse as AI, computational biology, data sciences, or robotics. Conversely, there are still a lot of open research challenges in MDE and we will identify more research challenges as MDE sees increased uptake.

MDENet aims to be the key driver for addressing these challenges. Join our community (at community.mde-network.org) to help us:

1. *Drive future MDE research* – learn about the latest technology advances and industrial challenges in our monthly research demonstration workshops, find experts who can help you tackle your challenges with MDE, and build on our seedcorn funding to establish new collaborations.
2. *Train the IT industry* – learn about existing MDE technologies and how to use them for your project by subscribing to our learning resources, learn from the experts in our regular virtual training sessions, and help others use MDE efficiently by contributing to, or creating new, MDENet learning resources.
3. *Tell the story of MDE* – join the conversation on our interactive online platform and events.

This newsletter offers two special feature articles, one by Federico Tomassetti from our partner community strumenta.community, who has been working on a manifesto explaining the vision of model-driven and similar technologies that put subject-matter experts at the heart of software development, and one by Michalis Famelis, who discusses the connection between modelling and uncertainty. You can also find pointers to events, past and future, that cover MDE topics. Future editions of the newsletter will continue to keep you informed about important developments in the community – if you have something you want to contribute, please consider our call for contributions at the end of the newsletter.

Have an enjoyable and relaxing summer,

Steffen Zschaler, director MDENet

Special Feature: A manifesto for subject-matter driven software engineering

Contributed by MDENet member *Federico Tomassetti*.

As a Language Engineering practitioner, I have been living a contradictory experience since I started this journey. On one hand I can see the results that certain practices bring to those who adopt them, while on the other hand it is evident that these practices are ignored by most persons who could benefit from them. It is very frustrating to see something with so much potential being neglected.

But to what practices am I referring?

I am referring to all practices which permit Subject Matter Experts (SMEs) who are not also developers to precisely specify their knowledge, in a way that can be analysed to provide feedback to them and support them in the development process, and then can be processed to obtain some result, like generating a useful application or drive some automated process.

When I try to explain this I look for an existing term I could use, and for lack of a better term I tended to use the term Domain Specific Language (DSL) in recent years. For the most advanced solutions I have designed I had indeed defined one or more DSLs, however I could not avoid noticing how problematic this definition was.

Discussions were often confusing because of the ambiguity of this term, which can indicate a simple internal DSL built in Ruby in a couple of hours, as well as a rich solution designed in MPS, providing specific editors and error checking.

The first source of confusion is the fact that internal DSLs and external DSLs are wrapped under the same term, when they are quite different. In essence internal DSLs are just an attempt to bend as much as possible a certain programming language (e.g., Ruby) to make the code appear almost as if it was written in a new language specifically designed for a certain kind of application. External DSLs are instead proper languages, with their own tooling, so that errors reported and editor suggestions are specific to this new language and not "borrowed" from a general-purpose language. Internal DSLs, in my experience, while being simpler to implement, do not bring the same advantages proper DSLs with proper tooling can provide.

The second reason for confusion was the fact that DSLs can be used to serve different kinds of users. One important distinction is between developers and non-developers. I think both are categories well served by DSLs, but I think it is most important to serve non-developers. The reason is that developers today already have a way to formalize their expertise: they can use formal languages they are familiar with. Yes, under certain circumstances they could benefit from DSLs making them significantly more productive, but for non-developer SMEs the change would be much more fundamental, because without DSLs they have no means to formalize their thoughts.

There is a problem with having the term DSL be used to indicate both DSLs for developers and DSLs for SMEs. When SMEs encounter a DSL intended for developers, let's say, for configuring a virtual machine, they may understand DSLs are not intended for them, as they appear to be very technical languages for software development specialists only.

As I discussed with other practitioners, I found how this frustration was shared. While this was not of great consolation, at least I realized I was not the only one experiencing this. So, I started to reflect on this with some of the practitioners I admire the most. We discussed the necessity to focus on solutions to raise the level of abstraction specifically for SMEs.

We also agreed on the fact that this principle is not tied to any specific technology: years ago I would have used Xtext to build a solution for this, nowadays I would use JetBrains MPS most of the time, a few years from now I could be using different technologies. Even today, in certain contexts I use technologies that are simpler to obtain results with a more limited effort or to integrate with particular tools the SMEs are already familiar with. In other words, we do not see the specifical technical approach (e.g., MDE or DSL) as the one most important to define what we do.

We wanted not to focus on technologies, but to focus on the problems which are preventing the adoption of better tooling for SMEs. The main one is probably the lack of awareness. To address this, we would need to talk to SMEs specifically, instead of just focusing on communicating with other Language Engineering practitioners. There are also organizational issues we need to face, in helping organizations to change how developers and SMEs collaborate.

These motivations led Markus Völter, Sergej Koščejev, and me to create the Subject Matter First Manifesto. This is the result of months of work to refine the text and ensure we were capturing not only our views but also the views of people in the community we hugely admired. Seventeen of them have chosen to be the first signatories of the manifesto, and we are very grateful for their support.

You can find the result of this work at https://subjectmatterfirst.org/. I hope you can share these ideas and work with us to promote them. Signing the Manifesto would be a very appreciated first step. We hope, with your help, to encourage putting Subject Matter Experts at the centre of the stage, focusing on providing them the best support possible for their work.

Special Feature: Modelling different kinds of uncertainty

Contributed by MDENet member *Michalis Famelis*.

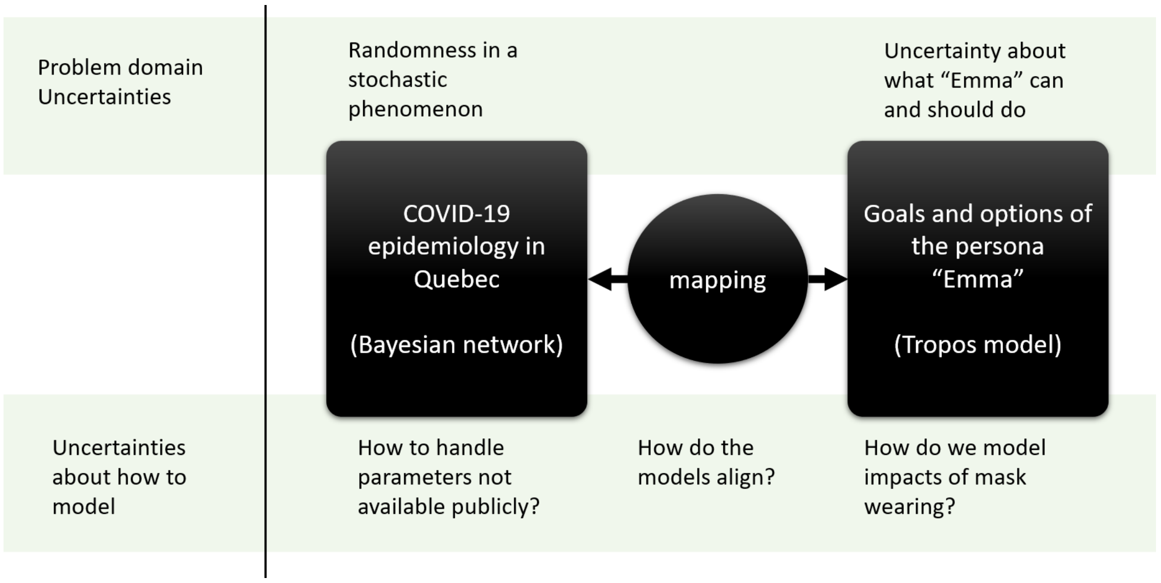
In 2010, David Garlan wrote that "the reality of today's software systems requires us to consider uncertainty as a first-class concern". Eleven years later, the only thing that has changed is that we now live in an even more complex and uncertain world. Uncertainty permeates the systems we build, the requirements for which we create them, the processes by which we develop them, the infrastructure on which we base them, and the conditions under which we operate them. Uncertainty has many meanings and takes many forms. It can arise by lack of knowledge ("epistemic") or due to randomness ("aleatory").

Sometimes new information can help resolve it; other times it can be inherently irreducible. Usually, we think about uncertainty in terms of the context in which our systems need to operate; but we must often deal with uncertainty internal to the workings of our systems. In fact, in 2020, Troya et al. catalogued six types of uncertainty studied by the software modelling research community in more than 120 papers in the last 20 years.

Walker et al. suggested thinking about uncertainty in terms of a spectrum from complete certainty to total ignorance. Towards the "certainty" end of the spectrum, we have good enough models within well-understood *error margins*. As we move away from certainty, we have models that vary with known *probability distributions*. Less certain models cannot rely on such distributions and make do with the *relative likelihood* of possible scenarios. Even more uncertainty means we cannot even rank these alternative scenarios – but at least we know what the *possibilities* are. One more step away from certainty, and we are as wise as Socrates in our *recognized ignorance*. And on the other end of the spectrum we have *total ignorance*, Rumsfeld's "unknown unknowns".

Uncertainty concepts are found in many formalisms and modelling languages. Sometimes modelling uncertainty is as simple as documenting the systematic error of the output of a function or as classifying a bug as "unassigned" on a ticketing system until we figure out who should take on fixing it. People also specifically model uncertainties with purpose-specific models, ranging from simple Markov chains to the sophisticated standardization proposal considered at the Object Management Group. Facing a wide range of flavours of uncertainty, we thus have an equally wide range of modelling approaches. How easy is it to navigate their combinations and interdependencies?

My collaborators and I recently performed a case study to better understand this. We played the role of developers building an app to help people stay safe during the COVID-19 pandemic. As part of early requirements engineering, we created a persona called Emma, that lives in Quebec and wants to use the app to decide how to get dinner. Emma has some dinner options (cooking in, getting take-out) and wants to protect herself and her local community. We modelled Emma's options using Tropos, a language for modelling goals and requirements. We also modelled a publicly available epidemiological model of COVID-19 in Quebec as a Bayesian network. We then connected these models (see figure), to show how Emma could connect her personal decisions with their social impacts.


Problem domain Uncertainties
Randomness in a stochastic phenomenon
Uncertainty about what "Emma" can and should do
COVID-19 epidemiology in Quebec (Bayesian network)
mapping
Goals and options of the persona "Emma" (Tropos model)
Uncertainties about how to model
How to handle parameters not available publicly?
How do the models align?
How do we model impacts of mask wearing?

The Quebec epidemiological model and Emma's goal model are both about uncertainty. The former captures the inherent randomness of a stochastic phenomenon; the latter expresses the different scenarios available to Emma. While modelling, we also documented our own uncertainty as modellers, i.e., about *how to build the models*. We worried about things such as "how should the two models be connected?", "how should we model this decision?", "what should we do with missing values?" etc.

Bayesian networks and goal models were good for capturing the uncertain elements of the problem domain. However, capturing the uncertainty of *the modelling process itself* required a different set of concepts. We captured these uncertainties with DRUIDE, a specialized modelling language which allowed us to decouple modelling uncertainty about the design of the models from uncertainty internal to the problem. The experience made it clear that the different types of uncertainty required very different modelling approaches and treatments.

I have been working for over 10 years on this kind of "design uncertainty". My goal is to mine it from the development context (e.g., developer conversations) and / or to capture it in (semi-)formal representations. I then use these to perform automated reasoning that preserves uncertainty, with the goal to provide feedback to modellers to assist them in building better models. Key to this vision is a more general idea: that understanding the different meanings of uncertainty is necessary if we are to create modelling infrastructure that lets practitioners reason about many kinds of uncertainty at once, depending on how they are modelled, how they interact, and how they affect each other. Achieving this deeper understanding can only happen in a community like MDENet, that allows researchers and practitioners of many different backgrounds and expertise specialties to freely exchange their unique perspectives.

Highlights, News, and Events

MDENet research demonstrations

We have held two highly successful research demonstration workshops in June and July. You can find recordings of the demonstrations on the MDENet YouTube channel. The first demonstration workshop concentrated on MDE on the cloud with a presentation by Dimitris Kolovos from the University of York on the Epsilon Playground, a web-based platform to enable easy experimentation with an advanced model-management platform. The second presentation was given by Panagiotis Kourouklidis and Joost Noppen from BT Research and Development on using MDE to manage machine-learning algorithms in practical cloud-based deployment.

The second demonstration workshop focused on transforming models and tracing change; that is some of the automation support offered by MDE. Artur Boronat from the University of Leicester presented YAMTL a tool for transforming models without the need for an external transformation language. Joe Habgood from the High Integrity Expertise Centre at Capgemini Engineering spoke about their approach towards traceability and how MDE technologies are helping Capgemini Engineering track changes across large, multi-team software development efforts.

Conferences of interest

SiriusCon'21

Sirius is a tool for building graphical domain-specific modelling languages developed by Obeo and the Eclipse community. Obeo regularly organise a short conference with talks about the use of Sirius in various domains and general MDE topics. The program for this year's conference (15 to 17 June) is available here. As in every year, the conference featured an overview of new developments in Sirius, not least the new Sirius Web for building diagrammatic editors on the web. This was complemented by an overview of applications of graphical modelling in domains as varied as low-code software development, systems engineering, infrastructure as code and cloud deployment, and clinical pathway design. Video recordings of the talks are not yet available from the conference website but are usually published some time after the event.

17th European Conference on Modelling Foundations and Applications (ECMFA'21)

ECMFA is one of the key academic conferences on model-driven engineering. This year's edition was held online, organised by the strong MDE research community in Bergen, Norway. As in every year, the conference is embedded in the context of STAF – the Software Technologies: Applications and Foundations federation of conferences – which offered a rich program of complementary software-engineering events, including the Transformation Tool Contest, which this year focused on MDE challenges in translating model queries into SQL and in dynamically adapting workflows. The proceedings of ECMFA are freely available in the Journal of Object Technologies here. Two big topics this year were modelling of uncertainty and modelling for the railway domain.

Looking ahead

In MDENet, we have put together an exciting programme of research demonstrations and we also expect to start our MDE training events as well as run our first thematic workshop on MDE for AI and Data Science (more detail on this will be published soon). There won't be a research demonstration workshop in August as we take a break for the summer, but we will continue on 29 September with a workshop on large models and model optimisation, followed by a workshop on 27 October on model synchronisation and comparison and a workshop on model management on 24 November. We look forward to welcoming you at these research demonstrations – please RSVP on the events on the MDENet member platform community.mde-network.org.

The top academic conference on MDE, MODELS will take place virtually from 10–15 October. Registration will open soon at modelsconference.org. The conference features technical papers as well as tool demos, an educators' symposium, and a broad range of workshops.

Finally, you should check out the funding offered by MDENet. We have recently opened our seedcorn fund, which you can apply for on a rolling basis, and we expect to open our dissemination and commissioning funds very soon. More information about these funds can be found on the member platform in the Funding topic.

Community Call

We are looking for ideas and contributions for future newsletters. If you have something to contribute or want to provide feedback on this newsletter, do get in touch at mdenet@kcl.ac.uk.

MDENet lives through the engagement of its members. We are looking forward to your contributions to the network! If you want to share your views, ask a question, find new collaborators or simply want to learn more about MDE, join our community at community.mde-network.org. There, you can also find and add links to events that might be of interest to the wider MDENet community.