The Relevance of Model-Driven Engineering Thirty Years from Now

Gunter Mussbacher[1], Daniel Amyot[2], Ruth Breu[3], Jean-Michel Bruel[4], Betty H.C. Cheng[5], Philippe Collet[6], Benoit Combemale[7], Robert B. France[8], Rogardt Heldal[9], James Hill[10], Jörg Kienzle[1], Matthias Schöttle[1], Friedrich Steimann[11], Dave Stikkolorum[12], and Jon Whittle[13]

[1] McGill University, Canada
[2] University of Ottawa, Canada
[3] University of Innsbruck, Austria
[4] University of Toulouse, France
[5] Michigan State University, USA
[6] Université Nice-Sophia Antipolis, France
[7] University of Rennes / INRIA, France
[8] Colorado State University, USA
[9] Chalmers University of Technology, Sweden
[10] Indiana University-Purdue University Indianapolis, USA
[11] Fernuniversität Hagen, Germany
[12] Leiden University, The Netherlands
[13] Lancaster University, UK
{gunter.mussbacher,joerg.kienzle}@mcgill.ca,
damyot@eecs.uottawa.ca, Ruth.Breu@uibk.ac.at, bruel@irit.fr,
chengb@cse.msu.edu, philippe.collet@unice.fr,
benoit.combemale@irisa.fr, france@cs.colostate.edu,
heldal@chalmers.se, hillj@cs.iupui.edu,
matthias.schoettle@mail.mcgill.ca, steimann@FernUni-Hagen.de,
d.r.stikkolorum@liacs.leidenuniv.nl, whittle@comp.lancs.ac.uk

Abstract. Although model-driven engineering (MDE) is now an established approach for developing complex software systems, it has not been universally adopted by the software industry. In order to better understand the reasons for this, as well as to identify future opportunities for MDE, we carried out a week-long design thinking experiment with 15 MDE experts. Participants were facilitated to identify the biggest problems with current MDE technologies, to identify grand challenges for society in the near future, and to identify ways that MDE could help to address these challenges. The outcome is a reflection of the current strengths of MDE, an outlook of the most pressing challenges for society at large over the next three decades, and an analysis of key future MDE research opportunities.

Keywords: Model-driven engineering, challenges, research opportunities.

1 Introduction

Model-driven engineering (MDE) is now an established approach for developing complex software systems and has been adopted successfully in many industries including the automotive industry, aerospace, telecommunications, and business information systems [26][27][38]. However, MDE is arguably still a niche technology [51]. It has not been adopted as widely as popular programming languages such as Java and C#, and, whilst some modeling languages like the Unified Modeling Language (UML) have become widespread [19], they are often not used to their full potential [43] and the use of models to automatically generate systems is still relatively rare [51]. One could argue that now is a good time to reflect on the successes of MDE as well as its shortcomings. It is a little over ten years since OMG published the first Model Driven Architecture (MDA; http://www.omg.org/mda/) specification, almost 20 years since it adopted UML, and many decades since the first Computer-Aided Software Engineering (CASE) tools were introduced. In all that time, MDE has *not* become the de-facto way to develop software systems. It is perhaps time, then, to examine the barriers to MDE adoption as well as to look for opportunities where MDE can make a difference in the future.

Towards this end, this paper reflects on the last twenty years of MDE research and practice, makes a candid assessment of where we believe MDE has succeeded and failed, and highlights key research and application opportunities for MDE in the next 30 years. Our intent is to bring fresh impetus to the MDE community and to define a roadmap for future research in this area, particularly in areas that remain largely unexplored by the community. The paper provides an opportunity for MDE researchers to consider their current MDE research within the broader context of grand societal challenges, with the aim to stimulate novel modeling research, techniques, and tools.

To put together this roadmap, we followed an approach loosely based on design thinking [15]. Design thinking is a well-established, brainstorming-oriented approach to problem solving that attempts to understand a problem from diverse perspectives, applies creativity techniques to generate as many solutions as possible without prefiltering, and then down-selects and refines a smaller number of solutions based on well-defined criteria. In essence, design thinking is a process for tackling a problem by first *diverging* (pushing the envelope, envisioning novel ideas) and then *converging* (consolidating the results). Design thinking is based around a number of guiding principles that aim to take a diverse set of participants, each with differing views and experiences, and shape them towards a common and transformative solution: listening is favored over dominating, quantity of ideas over filtering, being positive over saying "No"; participation; seeking wild ideas; and trusting the process.

In our case, the "problem" to which we applied design thinking was how to increase the adoption of MDE and change the perception that MDE might not yet be a solution for the grand societal challenges of today. We brought together 15 junior and senior MDE researchers for a week-long design thinking exercise. Participants performed a series of activities, inspired by design thinking and creativity literature, that promoted thinking outside the box, including external provocations.

The result is a reflection of the current strengths of MDE and an analysis of key future research opportunities for MDE. This exploratory paper first gives an overview of the key accomplishments of the MDE community over the last 20 years (Section 2.1), and then summarizes major current problems in MDE (Section 2.2). Before continuing, the employed methodology based on design thinking is explained in more detail, including a description of specific activities and their rationale (Section 3). The paper then re-examines MDE through the lens of what are the most pressing challenges for society at large over the next three decades (Section 4.1). By focusing on four fictitious future software systems (Section 4.2), the paper unravels how MDE does or does not address the future challenges of society. Based on this analysis, the paper suggests four grand challenges for MDE (Section 5), which we hope will stimulate new research directions in this area. Section 6 concludes the paper and proposes action items that can be initiated immediately.

2 The Last 20 Years

MDE has made significant progress in addressing software engineering challenges over the past 20 years. The major areas of advancement include: modeling languages, model analysis techniques, model-based verification and validation, and model transformations. Each of these areas has developed foundational theories, tool support, bodies of empirical evidence, and, to varying degrees, has been used in industrial settings. For each area, we identify the key research challenges being addressed, highlight the key accomplishments, and give a few representative examples.

2.1 Major Areas of Advancement

Modeling Languages. Researchers working in the area of modeling languages have focused on two key challenges [21]: (i) *Abstraction Challenge:* What kind of modeling constructs and underlying foundation is needed to support the development of domain- or problem-level abstractions that are considered first-class modeling elements in a language? (ii) *Formality Challenge:* What characteristics and/or properties of a modeling language are necessary to enable automated processing and rigorous analysis? Furthermore, what aspects of a language should be formalized?

To address these challenges, a complementary set of strategies has evolved [21].

Extensible General-Purpose Modeling Languages. The abstraction challenge is addressed by providing a general-purpose language that has support for customizing the language to a specific domain. Example customizations are profiles (e.g., UML profiles), domain-specific modeling processes, and, at a fine-grained level, the use of specialized syntactic forms and constraints on specific modeling elements. The formality challenge can be handled by either mapping the modeling language to a formal language, or annotations can be added to the modeling language at the meta-model level to constrain properties that should hold between language elements.

Domain-Specific Modeling Languages (DSMLs). In order to create a modeling language for a given problem domain, meta-metamodeling mechanisms, such as OMG's

MOF [33] and its Ecore implementation have been extensively used. Intelligent textual and graphical editors, together with debuggers and code generators, can now be built (and even modeled) for DSMLs with relatively little effort.

General-purpose modeling languages are relatively more popular in the research, industrial, and educational arenas. Furthermore, the use of modeling has become sufficiently mature such that modeling standards have emerged. UML has been the de facto standard for object-oriented modeling [40]. Furthermore, commercial tools are also available for commonly used modeling languages, such as the Object Constraint Language (OCL), the Systems Modeling Language (SysML), and the Business Process Model and Notation (BPMN). Recently, numerous studies have been performed to study the impact of modeling on various aspects of software development [23][40], for use in specific domains, such as embedded systems [1], and to study the impact of the use of modeling languages in industry [43].

While initially DSMLs were created on a limited basis by individual organizations mostly in the research sector, numerous industrial organizations have witnessed the significant advantages of using DSMLs, particularly when considering automatic code generation and domain-specific analysis as objectives. As such, the field of modeling language engineering has emerged as an important area of research to enable a broader community to systematically develop DSMLs for their respective domain and organization. Example frameworks to support DSML development include MOF (http://www.omg.org/spec/MOF), EMF (http://www.eclipse.org/modeling/emf), VisualStudio [14], JetBrains/MPS (http://www.jetbrains.com/mps), Kermeta [29], GME (http://www.isis.vanderbilt.edu/Projects/gme), Epsilon [31], and Xtext (http://www.eclipse.org/Xtext). A popular DSMLs in relatively wide use is MATLAB's Simulink (http://www.mathworks.com/products/simulink).

Model Analysis. While the process of modeling facilitates a better understanding of system requirements, design constraints, and user needs, the value of models increases significantly with the ability to automatically process the models and analyze the models for various properties. Significant progress has been made to formally analyze models for behavioral properties [21][34][37]; analyze models for structural properties [6], both within a given diagram type [12], and across multiple types of diagrams [8]. Within the embedded systems domain, model analysis is achieved by executing models in simulation environments, such as Simulink [21][30] or USE [35]. Also models may be queried using standardized model query languages [32]. Finally, model understanding can be achieved through animation and visualization [13][22][44][46]. In some cases, production-quality tools have been built from research tools (e.g., Microsoft's Static Driver Verifier is based on a model-based approach to find errors in device drivers using the model checker SLAM [4]).

Model-Based Analysis. Models have also been an enabling technology used to facilitate numerous software and systems development tasks. For example, model-based testing has long been used in industry [7][17][18][48], and for specific domains, such as reactive and embedded systems [9]. Enterprise architecture models are among the modeling approaches settled in practice ([28], also see The Open Group Architecture Forum at http://www.opengroup.org/subjectareas/enterprise/togaf). The major goal of these models is to document actual elements of an organization's IT infrastructure,

and to interrelate these elements as a basis for further analysis and decision support. Since enterprise architecture models usually get very large, visualization aspects have been considered for several years [10]. Similarly, business process and workflows models have been adopted by industry for many years. Business process models are both applied in a pure organizational context and within IT management to analyze organizational processes and their IT support [45]. In addition, workflow models are used to configure workflow engines, thus they have been precursors of using models at runtime [49]. Model-based testing applies implementation-independent models and code generation to the area of testing [41]. The manifold approaches in this area have not only addressed theoretical considerations of generating models and test cases, but yielded also practice-oriented methods and tools [48] and standardization efforts [3].

Model Transformations (Management). A model transformation establishes a relationship between two sets of models, and itself may even be model-based. Several categories of model transformations have been defined, as well as the intent for model transformations [39]. An operational transformation takes a source set of models to produce a target set of models that are a refinement, abstraction, or refactoring of the source. Emerging techniques in this category focus on the composition of multiple views to form a single integrated model, the decomposition of a single model into multiple models, each representing a different aspect of a system, and the translation of models to a format amenable to automated analysis, including static analysis, model checking, and other types of behavioral and performance analyses.

Synchronization transformations enable model traceability and synchronization between a model and its related artifacts. Examples include transformations to support code generation and code updates to reflect changes to models. The OMG Query/View/Transformation (QVT) standard [32] defines several languages that can be used to define such transformations at different levels of abstraction. To facilitate their use and development, Czarnecki and Helsen developed a survey of features used for transformation languages [16]. Successful tools for model transformations include ATL (http://www.eclipse.org/atl/), Epsilon (http://www.eclipse.org/epsilon/), and several tools based on Triple Graph Grammars [25].

Other contributions have gained traction, but are not as well established as the above, including model repositories (e.g., REMODD: http://www.remodd.org), patterns, aspects, features, models at run-time, and MDA/MDE processes.

2.2 Major Current Problems in Model-Driven Engineering

While much progress has been made in MDE over the last 20 years, the MDE community also has recognized many problems that it still must face. This subsection summarizes the main current problems in the field of MDE (in no particular order) as identified by the workshop participants.

Shortcomings of MDE to Address Increasing Demands on Software. Software has to respond to an ever-increasing number of demands. The explosion of stringent functional requirements and qualities is complemented by the ever-increasing need to customize and tailor software to specific usage contexts. Many software systems are tightly connected with their environment, are distributed, need to support heterogeneous

platforms, and/or are open in nature. Software needs to adapt to rapidly changing hardware and implementation platforms, and is developed in a context that requires developers to shorten time-to-market to a minimum. In such context, the inherent complexity of the problems that we are trying to solve with software keeps growing.

Current modeling approaches, techniques and tools do not live up to the challenge. Often, mature tools provide techniques that can successfully cope with software systems that we were building a decade ago, but fail when applied to model complex systems like the ones described above. Some academic techniques propose interesting ways of addressing these shortcomings, but the prototypical nature of academic tools often prohibits their application to the development of real-world software systems.

Obstacles for Tool Usability and Adoption. The proliferation of modeling languages, tools, and techniques makes it hard for users to commit to using MDE. Even after a suitable language and tool have been identified, the users face significant usability challenges [24][42][52], e.g., steep learning curves, arduous user interfaces, and difficulty with migrating models from one version of a tool to the next. Despite the fact that software development is a team activity, there is little effective tool support for collaborative modeling. In general, tools do not support the fundamentally creative side of the modeling process due to their inflexibility and complexity. Far fewer MDE community or interactive forums on the web can be consulted to find solutions to problems when compared to programming-based forums. Finally, model transformations, which are essential in order for MDE to be effective, are difficult to maintain and adapt to changing requirements and implementation platforms [50].

MDE Is Not Considered "Cool". Even though MDE has been around for over 10 years, it is currently not as widespread in industry as the modeling community has hoped for [51]. As bluntly illustrated in Table 1, MDE is simply not considered cool. Why this is the case needs to be investigated. Maybe, the bad experience with CASE tools decades ago still casts a dark shadow on MDE. Maybe, the effects of the so-called UML Fever [5] are continuing to hurt the perception of MDE by people outside the community. Some even argue that there is a stronger need to investigate people's perception of MDE than to research new MDE technologies [11].

Table 1. Results of six queries (with quotes) on Google Search, February 12, 2014

	"Agile"…	**"MDE"…**	**"Model-Driven Engineering"…**
..."is cool"	4,250	10 (*)	0
..."is not cool"	1	41	10

(*) The first two results actually linked to Cabot's article entitled "Model driven engineering is not cool" [11], and 7 links had nothing to do with MDE.

Inconsistencies between Software Artifacts. A number of companies are using software modeling languages such as UML in their architecture development. The problem is that these models are often ignored as soon as one moves on to coding. Changes are made in the code but not in the models, leading to inconsistencies between software models and code. Synchronization of models between different levels of abstraction is not the norm. Good tool support is lacking to keep these models in

sync today. A complicating factor is that often a system is modeled with multiple views using different models and modeling notations, thus further increasing the likelihood of introducing inconsistencies between these models. Even when additional information is overlaid onto an existing view (as is the case, for example, in UML, when stereotypes define non-functional properties), there are no guarantees that the resulting system is consistent or correctly functioning.

Models Are Still Not Valued as Much as Code. The advantage of code is that it is a product on its own. It is often quite motivating to work directly on the product. It permits a software engineer to point out, e.g., that this part is due to her programming. In addition, one can obtain constant feedback when programming by executing the code, allowing one to easily experiment with the code and test its behavior. Unfortunately, for many people, modeling is considered a superfluous activity that becomes an activity in itself not necessarily for the benefit of the software development. This concern makes it hard to see both the short and long time benefits of using models to specify the product and creates a lack of trust in the technology.

Lack of Fundamentals in MDE. Unlike most other fields of engineering, model-driven engineering does not have a Body of Knowledge (BoK) as such. Some recent initiatives such as SWEBOK (http://www.computer.org/portal/web/swebok) and SEMAT (http://semat.org/) aim at filling this gap, but the required effort is huge. This deficiency also hampers the support for reuse. Programming languages have libraries. Modeling libraries are emerging (e.g., [2], also see REMODD) but the lack of common representations, query mechanisms, and critical mass pose obstacles.

Education Issues. There is a large mismatch between modeling examples found in books and the ones used in the real world. For example, small and unrealistic state-chart diagrams are often used. It is relatively easy to teach the syntax of a modeling language such as UML, but we still struggle with how to teach design principles using modeling. For students, it is difficult to learn to use their abstraction abilities [36][47], which have been shown to closely relate to software design skills. For effective teaching, students need to be motivated by the benefits of modeling (e.g., solution complexity can only be managed by models instead of simple coding problems).

Uncertainty in Environments, Requirements, and Systems. It is not that hard to create a model if the problem fits one's mental picture. That usually depends on the modeler's domain knowledge and well-defined, stable domain abstractions. However, nowadays software more and more adapts (and sometimes self-adapts) to its environment. The inherent uncertainty in many such problem domains (such as human science, social issues, etc.) and environments makes software very complex. In the face of uncertainty, actual modeling techniques neither ease the integration of multiple concerns nor support problem domain modeling.

Lack of (Industrial) Evidence of Benefits. There have been a number of empirical papers in the last few years that address the lack of industrial proofs of benefits [20][50]. These papers give a good status of the use of MDE in industry, but they do not let us understand why MDE projects fail or succeed. We are still lacking knowledge on factors that make MDE successful, also considering that model-based approaches are regularly used in the hardware industry (e.g., model checking to analyze hardware designs instead of testing).

3 Methodology

In this section, we describe our methodology for defining the MDE roadmap presented in this paper. The method is loosely based on principles of design thinking [15], which aims to approach a problem from as many angles as possible (i.e., problem understanding), generate as many ideas as possible (i.e., ideas generation), and then only finally consolidate those ideas into a small number of workable solutions (i.e., ideas selection). This section describes how we applied design thinking in terms of the concrete activities (see Table 2) that our participants undertook.

Table 2. Activities of the Design Thinking Workshop

(Phase) Activities	Rationale
(1) Put Aside Personal Interests. Each participant was given an opportunity to talk about their own research agenda.	The aim was NOT to look for research overlaps or to build on existing research strengths. Since the workshop aimed at getting people to think differently, participants needed to put aside their own research interests for the week. By providing a forum to air their research first, participants feel content that their research has been articulated, and also feel happy to step outside their boundaries.
(2) Think Beyond MDE and Software Engineering. Participants were asked to identify the grand challenges of the population at large in the next 30 years. This was done by asking participants to describe two futuristic scenarios: a perfect day and a hellish day in 2030. (*)	One way to reach genuinely novel and different ideas is to change context completely. By asking participants to temporarily not think about software engineering, but instead think about societal challenges, we created an environment in which new thinking could blossom and participants could engage with issues that they feel passionate about.
(3) External Influences. Two external speakers from outside the software engineering community were invited to talk to participants: one was an expert on environmental sustainability, the other was an expert on robotics in marine environments.	External speakers were introduced at key points as a nudge to make sure participants continued to think differently: these speakers were introduced to provide inspiration from a perspective traditionally not considered in MDE research.
(4) Ideas Generation. Participants self-organized into small groups and generated ideas for future systems that could address grand challenges identified in phase 2.	Participants were provided with a safe, supportive environment to generate ideas. Ground rules were put in place to ensure that any idea could be heard. Participants were told to value listening over dominating in conversations, quantity of ideas rather than pre-filtering, being positive over being critical, to seek wild ideas, and to fully engage with the process.

Table 2. (*Continued*)

(5) Consolidation of Ideas. The participant groups were taken through a series of iterative cycles where they presented their ideas to an external mentor and their peers, received constructive feedback, and then were asked to re-present the evolving idea at regular intervals.	Through this process, the most promising ideas were nurtured to ensure that they satisfied the criteria of: novelty, feasibility (within 30 years), relevance to MDE, and a different way of thinking.
(6) Documentation of Results. During the workshop, the participants began writing this paper, which was then completed after the workshop.	The ideas developed in phase 5 were used as driving exemplars to identify new areas for MDE research, which are the ultimate result of the design thinking exercise.

(*) Not included for space reasons, see http://www.cs.mcgill.ca/~joerg/SEL/motb-day.html.

Setup and Participant Selection. We brought together 15 participants for a week-long design thinking workshop at McGill University's Bellairs Research Institute. Participants were required to devote themselves fully to the workshop for the whole week so that outside distractions could be minimized. The approach to participant selection was largely "curated" in that the organizers made a prioritized list of potential participants with the aim of maximizing diversity in terms of seniority, gender, and research area. In the case that an invited potential participant could not attend, the organizers went down the prioritized list trying to maintain diversity. The final set of 15 participants included 2 women and 13 men, who came from thirteen academic institutions from across 8 countries and covered research in a wide spectrum of the software lifecycle from early requirements to implementation. (It was of course disappointing to not maintain a better gender balance. Despite best efforts, we were constrained by the heavy skew towards male MDE researchers.)

Activities. Table 2 summarizes the activities of the design thinking workshop. Activities were designed to avoid tunnelled thinking so that genuinely fresh ideas could emerge. This was achieved in a number of ways: (1) Move people away from their own research areas so that they are open to fresh ideas; (2) Move people away from software engineering by having them discuss grand challenges of society today; (3) Introduce external speakers at key points to inject fresh ideas from a completely different perspective; (4) Encourage unfettered ideas generation, where "anything goes" and pre-filtering of ideas is discouraged; (5) Consolidate ideas by down-selecting and/or refining them according to well-defined criteria; and (6) Document the results. All of these activities are tried and tested, and are based on well-accepted techniques in design thinking and/or creativity theory.

Rationale. Table 2 gives the rationale for each phase of the design thinking exercise. Each phase was carefully designed so that, taken as a whole, the phases would lead to new ways of thinking about MDE and MDE research. It is important to understand that many of the activities in phases 1-5 are not an end by themselves, but are either ways of moving the group of participants towards the end goal, or ways of generating useful by-products. The ultimate end-result comes in phase 6, where the

roadmap is defined. The ideas selected in phase 5, for example, were example futuristic systems where MDE could have an influence. Rather than proposing that the community should start developing these systems, we see these systems as useful driving ideas to help understand where the current gaps are in MDE research.

4 Grand Challenges for the Next 30 Years

Through several iterations of group brainstorming activities, prioritization activities, and the perfect/hellish-day-in-2030 session (phase 2 of our methodology), we identified six grand challenges for society at large to be addressed over the next three decades. They are introduced here in Section 4.1, in no particular order of importance. We also provide examples of futuristic software systems in Section 4.2 to illustrate potential solutions to some of these challenges and highlight the characteristics of such systems that may have to be addressed by new modeling solutions (phases 4 and 5 of our methodology).

4.1 Six Grand Challenges

1) Resource Affordability and Availability. Many kinds of resources exist, including health, food, knowledge, and energy. Yet, they are not available and affordable to all in equitable ways, even for primary needs. There is a need to substantially improve the management of resources, including their Creation, Access, Distribution, Usage, and Disposal (collectively referred to as CADUD), in order to improve resource availability and affordability for everyone. Additional threats to mitigate include costs, corruption, greed, wrong incentives, lack of basic infrastructures and data, and the use of local optimizations instead of more sensible global optimizations.

2) Sustainability. Many resources such as energy and food are not easily renewable without control and efforts, and we are now facing many sustainability issues that demand more precise, trustable, and timely information for decision making. In particular, there is much room for better trade-offs between economic growth and responsible use of resources, for education and understanding of cause and effects of CADUD-like resource management, and for ways to avoid misinformation of sustainability factors by special interest groups. There is a vicious cycle where the need for comfort often leads to growth, which in turn requires more energy, leading to pollution (and global warming) that stresses our level of comfort. Attitudes need to change at all levels of granularity (from the individual level to city-wide, regional, national, continental, and planetary levels).

3) Disaster and Crisis Management. There is a strong need to improve the predictability of natural disasters such as storms and earthquakes, as well as of human-triggered crises related to economy, health, and social tensions. Where predictions are impossible or fail, societies should be enabled to react in a timely way.

4) Steady-State Economy. Global and local economies are still based on a growth model that cannot be sustained forever. Mechanisms are needed to bring economies of

any scale to a "steady-state" that would no longer rely on continuous growth, exploiting resources and people in all areas of the world.

5) Life Balance. Individuals are subject to many extrinsic factors such as peer pressure and demand for performance that are difficult to balance with real intrinsic motivation and a sustainable lifestyle. They are also bombarded with an ever-growing amount of information that strains the individual's abilities to cope with life's challenges. Support is needed to help understand, manage, and control extrinsic factors and information to avoid getting caught in a pernicious "rat race" and, instead, to achieve a healthy balance in life.

6) Common Sense. Current governance structures are often subject to bureaucracy and abusive lobbying. There is an opportunity to bring back common sense in governance and better balance the weight of individual/common needs versus the interests of special interest lobbying groups.

4.2 Four Examples of Futuristic Systems

1) Model-Experiencing Environments (MEEs). Facing the vicious cycles that hamper sustainable solutions development and effective resource management, we believe that any person, community, decision maker, or company should be able to play, analyze, and "experience her personalized *Model-Experiencing Environments (MEEs)*. Those *MEEs* are very "sophisticated and highly tuned *"what-if"* impact models, but with a simplified and adaptive user interface. Each *MEE* consists of combinations of interconnected models based on open data, enabling one to play, run, and see evidence on the impact over resource consumption chains.

We envision different kinds of *MEEs*. In any *MEE*, the user can adjust the level or amount of different properties she is interested in, e.g., impact on health, employment, economy, amount of waste, gas, water, or even taxes. Then, she is able to specify and assemble certain criteria in a *do-it-yourself* way for the scenario she is interested in. The selection is automatically propagated to the outcome view where the different impacts are shown. The impacts are visualized in graphs, charts, or any adapted interfaces such as personalized virtual reality ones. Finally, deployed *MEEs* feed back into underlying open models to improve accuracy or user confidence.

The following list overviews the different kinds of *MEEs*:

- *MEE* for Game-Based Learning: allows children to learn about impacts in a playful way (tackling challenge 5 (life balance)).
- Crowd-Sourcing *MEE* Use: permits several people to see impacts if they do something together (tackling challenges 1 (resources affordability and availability), 2 (sustainability), and 5).
- *MEE*-Enabled Community Decisions: informed decisions can be made by community members (tackling challenges 1, 2, 5, and 6 (common sense)).
- *MEE*-Driven Policy Analysis: enables policy makers to understand impacts of their decisions (tackling challenges 1, 2, 3 (disaster and crisis management), 4 (steady-state economy), 5, and 6).

In addition, *MEEs* are likely to be useful in broader domains than just resource impact models. Any kind of experiencing can benefit from *MEEs*: personal health

companion, family expenses habits monitoring, etc. This ultimately allows models to be part of everybody's life and usage, making them trusted daily objects that enable everyone to learn, think, and act on her own.

2) Making Zense. Imagine relaxing yoga music… "Do you feel like you are in a rat race? Do you need to make personal decisions, but can't evaluate their short term and long term impact? Do you have trouble balancing work, family, and personal activities?" *Making Zense* helps you find a healthy balance. Humans are unique; modeling a human being is too complex. A human story captures important events and facts, accomplishments, failures, health records, nutrition history, sleep patterns, and social connections. A human story is completely personal and confidential, i.e., it is not possible to identify a living person from her human story, but there is a way to assess happiness levels throughout a person's life.

Billions and billions of human stories make up the *Human-esZense* – a vast collective wisdom, the essence of the human race. It does not stop there. Cities and countries are also unique, and their stories are also found in the *Human-esZense*. From time to time, a role model emerges from human stories. It is shaped by societal forces at play. A role model displays characteristics that are beneficial to achieve happiness.

Making Zense feeds your human story continuously into the *Human-esZense* and compares it with similar human stories. Based on this collective knowledge, personal trajectories are continuously presented, possible outcomes of one's life with varying probabilities and happiness levels, and role models are used to characterize these possible paths along your road to happiness. Once a role model you would like to aspire to is selected, the *Human-esZense* enables the assessment of what-if scenarios by comparing the proposed changes to your life based on the role model against the *Human-esZense*, addressing challenge 5 (life balance)).

3) Models4 ∆○⌂∴ (Modeling for the Illiterate). Most modeling languages and tools target highly-educated experts. Yet, many complain that models are difficult to create and use. One reason is that we have not yet fully understood what modeling is, and the intuition needed to make it effective. In addition, a global trend nowadays is to invite the population at large to learn programming (e.g., see the `code.org` effort), as programming and configuration will become pervasively required. *Models4∆○⌂∴* is an application that enables anyone to create and use models needed to configure their daily lives and long-term goals, for example with the *MEE* and *Making Zense* systems. It is so intuitive that illiterate people can use it as effectively as domain experts, hence confronting the education portion of challenge 1 (resources affordability and availability). Note that by targeting illiterate people as a primary audience, the development of *Models4∆○⌂∴* helps us truly understand what modeling really is, which in turn enables us to transfer this knowledge to a much broader set of modeling approaches, including those for software and systems development.

4) Have You Thought of … (HYTo). Too often, projects are cancelled at very late stages. Political, cultural, or other factors can play a role in these decisions. For example, after the election of a local government, the political leaders decide to stop the creation of a promised 'very green' park, because of the intense lobbying of other parties, such as influential contractors who plan to develop the space into lucrative

real-estate projects. 'Have you thought of the coming elections, which could possibly be won by the opposition party?', the *HYTo* application will ask you – along with 'Have you thought of the increase in tourist revenue because of the park?' and 'Have you thought of the increased CO_2 emissions because of reduced green space and increased traffic to the real-estate project?'. *HYTo* helps you make decisions while taking the predictions of other (external) factors into account. Consequently, *HYTo* is applicable to all challenges identified in the previous sub-section.

5 Grand Challenges of Model-Driven Engineering

As a community, we have made substantial progress in the areas of modeling languages, processes, quality, and automated integration of models (across domains and at different levels of abstraction). In addition, we now have, or are very close to having, good modeling techniques for tackling complexities related to scalability, forecasting/predictions, data/knowledge-awareness, personalized adaptation, usability, real-time, and perceived intelligence. However, these techniques are currently not capable of supporting the modeling needs required to realize the kinds of systems outlined in the previous section. While the MDE community tends to cope well with only one of these dimensions of complexity at a time, existing and future systems will face many of these dimensions at the same time, for example:

- Real-time, knowledge-aware forecasting at the personal level (as exemplified by *MEE*, *Making Zense*, and *HYTo*, which all try to predict future events and behaviors based on gathered knowledge),
- Personalized, ubiquitous access for uneducated users (obviously applicable to *Models4*Δ○⌂ but also to a certain degree to all other identified systems as they are deployed on a massive scale to users without expert knowledge),
- Ultra-large scale, intelligent, near-future predictions (which is an essential part of *MEE*, *HYTo*, as well as *Making Zense* not just at the personal level but also at the level of whole communities or even countries), and
- Knowledge-aware shaping of usable models, i.e., the tailoring of models to stakeholders' immediate needs based on knowledge accumulated at the individual, community, and global levels (as required for *Making Zense*, *HYTo*, and *MEE* because the power of these systems lies in the fact that contextual models are provided at the right level of abstraction for each stakeholder).

In the final phase of our design thinking-inspired methodology, we more closely looked at the grand societal challenges, futuristic systems and scenarios, and iterative refinements of our ideas through the lens of the MDE community and distilled them into common threads to highlight the following four grand modeling challenges and new research areas for MDE for the next 30 years.

I) Cross-Disciplinary Model Fusion. One grand challenge is to better take advantage of modeling knowledge *across* disciplines. Our MDE community has focused much on software and systems modeling, without much interaction with modeling activities in areas such as artificial intelligence, databases, the semantic web, or human-computer interactions. This lack of interaction and awareness is even worse

when we consider entirely different fields, e.g., biology, economics, arts, law, medicine, and social sciences. We need to study more rigorously what other communities do and learn from their modeling experience and challenges. This will help us improve our modeling approaches to better deal with multiple dimensions of complexity, while at the same time enabling us to provide modeling approaches that better fit the needs of other disciplines. The MDE community has a lot to offer in terms of language, process, quality, and automation expertise that can be leveraged in these other disciplines. All of the challenges identified in the previous section require models from different disciplines to be *fused* into models that are usable by stakeholders. Solutions to challenges 2 (sustainability), 3 (disaster and crisis management), 4 (steady-state economy), and 6 (common sense) must, for example, make use of models from economics, physics, biology, and politics to adequately address these problems. Consequently, solution systems for these challenges (e.g., the highly sophisticated what-if scenarios of *MEE* and the context-aware questions of *HYTo*) rely on cross-disciplinary model fusion.

II) Personal Model Experience. A second grand challenge of MDE is to make modeling and the use of models directly benefit the *individual*. Nowadays, access to sophisticated models and model analysis is restricted to a select few. We need to find ways to provide individual end users with straightforward access to models that encode global information relevant to their particular situation. Furthermore, individuals must be allowed and able to customize these models to their particular context and needs, and feel confident that the customization is trustworthy and accurate. While some default models may be used as starting points, the high individuality of these personalized models presents new challenges for model reuse. Furthermore, innovative model analysis algorithms and tools have to be developed, that based on the global information and the individual's personal context, can produce valuable, timely insight, which the individual can then use to make decisions on a local scale in accordance to personal beliefs. Solutions to challenges 1 (resource affordability and availability), 2 (sustainability), and 5 (life balance), and hence solution systems for these challenges (i.e., *Making Zense*, *MEE*, and *HYTo*), depend on such a personal model experience to demonstrate to the individual the consequences of local/global and individual/communal actions. *Models4*Δ○⌂;;; on the other hand, highlights the need to pay close attention to non-expert users.

III) Flexible Model Integration. An additional grand challenge is to determine how software models should be structured to provide value when developing systems that *flexibly* address *many* concerns *simultaneously*, as seen in the four types of systems described at the beginning of this section. This challenge is complementary to the first one that seeks for cross-fertilization with radically different disciplines to MDE, but it focuses on heterogeneous concern integration *within* an application field. This integration is already happening in the industry today. For example in the automotive industry, mechanical parts, electronic parts, and software are now extremely integrated. Furthermore, telecommunication plays an important role, as cars start to communicate more and more with each other and the surrounding environment. Software modeling can play an important role in integrating these large and complex systems. Tackling this integration challenge also means to be able to dynamically use

and reuse models as well as integration strategies with better confidence in and predictability of the result. To this aim, means must be devised that allow modelers to specify assumptions and limitations of models explicitly, as well as the contexts in which a model can successfully be applied, and how to apply it. Solutions to any of the identified societal challenges require flexible model integration, which can be observed most prominently in the *MEE* and *HYTo* solution systems where models for differing concerns need to be assembled on the fly depending on the user's context.

IV) Resemblance Modeling – From Models to Role Models. Last but not least, modeling, and object-oriented modeling in particular, has traditionally adopted an Aristotelian view according to which individuals (objects) are classified by universals (classes). These classes introduce a very convenient level of abstraction in that they allow forgetting the myriads of individuals that, from the viewpoint of the modeler, are all more or less the same. In particular, the introduction of classes allows the reduction of a potentially infinite domain to a finite (and usually also rather small) one.

However, this abstraction is not without a price. In complex systems, the differences between objects may be more important than their commonalities, and if traditional class-based modeling is used, one quickly ends up with one class per object. While this is not a problem per se, it does question the usefulness of class-based modeling in these contexts. The real problem surfaces however when the number of significantly differing individuals becomes so vast that mapping them to classes boosts models to an ultra-large scale. In that case, it may make sense to resort to a *prototype-based* classification of individuals, defined by the similarity and differences of one individual from another. Certain individuals, the prototypes, then serve as *role models* for others, which characterize themselves by stating their role models and the differences from them. Interaction between individuals is first defined at the prototype level; individuals may choose to override wherever deemed apt. Models of this kind may never reach perfect accuracy; yet, they may trade precision for manageability which, at the ultra-large scale, may be the higher good.

The need for highly individualized models is most obvious in the *Making Zense* system (with its billions of unique human, city, and country models) and for the grand societal challenges where the individual is key (e.g., challenges 1 (resource affordability and availability), 2 (sustainability), 5 (life balance), and 6 (common sense)).

6 Conclusion and Proposed Action Items

This paper formulates a roadmap by describing four grand MDE challenges that need to be addressed by the MDE community over the next 30 years. *Cross-Disciplinary Model Fusion* highlights the need to investigate modeling in radically different disciplines. *Personal Model Experience* points out that the power of modeling and model analysis needs to be made available at an individual's level. *Flexible Model Integration* advocates looking inward at software models to find ways to capture and consolidate heterogeneous application concerns. Finally, *Resemblance Modeling* questions the applicability of class-based modeling for systems with large numbers of highly unique individuals.

The six societal and four MDE challenges are an opportunity for the reader to put her modeling research into the perspective of the broader context of grand societal challenges, possibly stimulating her to apply modeling research, techniques, and tools to new areas, to different disciplines, or to bridge the gap and connect fields that have traditionally been isolated. The intermediate workshop results (summary of MDE success stories, current MDE problems, six pressing challenges for society at large, the perfect/hellish day in 2030, and the examples of futuristic systems) provide a rich frame of reference that allows the reader to look at the relevance of her research, and the research of the MDE community as a whole, from a different angle.

The roadmap intends to inspire the MDE community. It is our hope that the ideas presented here will incite new research directions and new technologies, which eventually partake in the creation of systems, similar to the ones envisioned in this paper, that considerably improve the quality of life of mankind.

While the main purpose of this paper is to explore an MDE research roadmap for the next 30 years, there are two immediate action items that emerged through intense discussions throughout the workshop. First, there is a need in the MDE community to more actively look outward instead of inward and invite other disciplines to join the dialog. Perhaps, a cross-disciplinary or extra-disciplinary track at the MODELS conference (e.g., a *Models OUtside Software Engineering* (MOUSE) track) may be a promising start. Second, the Artificial Intelligence, Analytics, and Natural-Language Processing communities had a coup d'éclat when IBM's Watson won Jeopardy. The MDE community should look for a similar demonstration of MDE capabilities that helps solve a significant societal problem, captivates informed insiders and general audiences, and makes everyone understand the value of modeling.

References

1. Agner, L.T.W., et al.: A Brazilian survey on UML and model-driven practices for embedded software development. J. of Systems and Software 86(4), 997–1005 (2013)
2. Alam, O., Kienzle, J., Mussbacher, G.: Concern-Oriented Software Design. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) MODELS 2013. LNCS, vol. 8107, pp. 604–621. Springer, Heidelberg (2013)
3. Baker, P., Dai, Z.R., Grabowski, J., Haugen, Ø., Schieferdecker, I., Williams, C.: Model-Driven Testing – Using the UML Testing Profile. Springer, Berlin (2008)
4. Ball, T., Cook, B., Levin, V., Rajamani, S.K.: SLAM and Static Driver Verifier: Technology Transfer of Formal Methods inside Microsoft. In: Boiten, E.A., Derrick, J., Smith, G.P. (eds.) IFM 2004. LNCS, vol. 2999, pp. 1–20. Springer, Heidelberg (2004)
5. Bell, A.E.: Death by UML Fever. Queue 2(1), 72–80 (2004)
6. Berenbach, B.: The evaluation of large, complex UML analysis and design models. In: 26th International Conference on Software Engineering. IEEE Computer Society (2004)
7. Briand, L., Labiche, Y.: A UML-based approach to system testing. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 194–208. Springer, Heidelberg (2001)
8. Briand, L.C., Labiche, Y., O'Sullivan, L.: Impact analysis and change management of UML models. In: IEEE International Conference on Software Maintenance (2003)
9. Broy, M., Jonsson, B., Katoen, J.-P., Leucker, M., Pretschner, A. (eds.): Model-Based Testing of Reactive Systems. LNCS, vol. 3472. Springer, Heidelberg (2005)

10. Buckl, S., et al.: Generating Visualizations of Enterprise Architectures using Model Transformations. Enterprise Modelling and Information Systems Arch. 2(2), 3–13 (2007)
11. Cabot, J.: Model driven engineering is not cool (November 08, 2012), http://modeling-languages.com/mde-is-not-cool/
12. Cheng, B.H.C., Stephenson, R., Berenbach, B.: Lessons learned from automated analysis of industrial UML class models (an experience report). In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 324–338. Springer, Heidelberg (2005)
13. Combemale, B., et al.: Introducing simulation and model animation in the mdetopcased toolkit. In: 4th European Congress Embedded Real Time Software, ERTS (2008)
14. Cook, S., Jones, G., Kent, S., Wills, A.C.: Domain-specific development with visual studio DSL tools. Pearson Education (2007)
15. Cross, N.: Design Thinking: Understanding How Designers Think and Work. Berg, Oxford UK (2011)
16. Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: 2nd OOPSLA Wksh. on Generative Techniques in the Context of the MDA (2003)
17. Dalal, S.R., Jain, A., Karunanithi, N., Leaton, J.M., Lott, C.M., Patton, G.C., Horowitz, B.M.: Model-based testing in practice. In: 21st ICSE, pp. 285–294. ACM (March 1999)
18. Dias Neto, A.C., et al.: A survey on model-based testing approaches: a systematic review. In: Wksh. on Empirical Assessment of Softw. Eng. Lang. and Techn., pp. 31–36. ACM (2007)
19. Dobing, B., Parsons, J.: How UML is Used. Communications of the ACM 49, 109–113 (2006)
20. Farias, K., Garcia, A., Whittle, J., Lucena, C.: Analyzing the Effort of Composing Design Models of Large-Scale Software in Industrial Case Studies. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) MODELS 2013. LNCS, vol. 8107, pp. 639–655. Springer, Heidelberg (2013)
21. France, R., Rumpe, B.: Model-driven development of complex software: A research roadmap. In: Future of Software Engineering. IEEE Computer Society (2007)
22. Goldsby, H.J., Cheng, B.H.C., Konrad, S., Kamdoum, S.: A visualization framework for the modeling and formal analysis of high assurance systems. In: Wang, J., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 707–721. Springer, Heidelberg (2006)
23. Grossman, M., et al.: Does UML make the grade? Insights from the software development community. Information and Software Technology 47(6), 383–397 (2005)
24. Hill, J.H.: Measuring and reducing modeling effort in domain-specific modeling languages with examples. In: IEEE Eng. of Computer Based Systems (ECBS), pp. 120–129 (2011)
25. Hildebrandt, S., et al.: A Survey of Triple Graph Grammar Tools. Electronic Communications of the EASST 57, 1–17 (2013)
26. Hutchinson, J., Rouncefield, M., Whittle, J.: Model Driven Engineering Practices in Industry. In: ICSE 2011, pp. 633–642 (2011)
27. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical Assessment of MDE in Industry. In: ICSE 2011, pp. 471–480 (2011)
28. Inmon, W.H., Zachman, J.A., Geiger, J.G.: Data Stores, Data Warehousing, and the Zachman Framework: Managing Enterprise Knowledge. McGraw-Hill (1997)
29. Jézéquel, J.-M., Barais, O., Fleurey, F.: Model driven language engineering with Kermeta. In: Fernandes, J.M., Lämmel, R., Visser, J., Saraiva, J. (eds.) GTTSE 2009. LNCS, vol. 6491, pp. 201–221. Springer, Heidelberg (2011)
30. Kawahara, R., et al.: Verification of embedded system's specification using collaborative simulation of SysML and simulink models. In: MBSE 2009, pp. 21–28 (2009)
31. Kolovos, D.S., Paige, R.F., Polack, F.A.: Eclipse development tools for epsilon. In: Eclipse Summit Europe, Eclipse Modeling Symposium, vol. 20062 (2006)

32. Kurtev, I.: State of the art of QVT: A model transformation language standard. In: Schürr, A., Nagl, M., Zündorf, A. (eds.) AGTIVE 2007. LNCS, vol. 5088, pp. 377–393. Springer, Heidelberg (2008)
33. Lédeczi, Á., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J., Karsai, G.: Composing domain-specific design environments. Computer 34(11), 44–51 (2001)
34. Lilius, J., Paltor, I.P.: Formalising UML state machines for model checking. In: France, R.B. (ed.) UML 1999. LNCS, vol. 1723, pp. 430–444. Springer, Heidelberg (1999)
35. Martin, G., Büttner, F., Richters, M.: USE: A UML-based specification environment for validating UML and OCL. Science of Computer Programming 69(1), 27–34 (2007)
36. Mayart, F., Bruel, J.-M.: Psychological Requirements for Software Engineers: A Reverse Engineering Approach. In: IEEE C3SEE, pp. 137–146 (2004)
37. McUmber, W.E., Cheng, B.H.C.: A general framework for formalizing UML with formal languages. In: 23rd ICSE, pp. 433–442. IEEE Computer Society (2001)
38. Mohagheghi, P., Dehlen, V.: Where is the Proof? – A Review of Experiences from Applying MDE in Industry. In: Schieferdecker, I., Hartman, A. (eds.) ECMDA-FA 2008. LNCS, vol. 5095, pp. 432–443. Springer, Heidelberg (2008)
39. Moussa, A., et al.: Towards a model transformation intent catalog. In: First Workshop on the Analysis of Model Transformations, pp. 3–8. ACM (2012)
40. Nugroho, A., Chaudron, M.R.V.: A survey into the rigor of UML use and its perceived impact on quality and productivity. In: ESEM 2008, pp. 90–99. ACM (2008)
41. Offutt, J., Abdurazik, A.: Generating Tests from UML Specifications. In: France, R.B. (ed.) UML 1999. LNCS, vol. 1723, pp. 416–429. Springer, Heidelberg (1999)
42. Pati, T., Feiock, D.C., Hill, J.H.: Proactive modeling: auto-generating models from their semantics and constraints. In: Wksh. on Domain-Spec. Modeling, pp. 7–12. ACM (2012)
43. Petre, M.: UML in practice. In: 2013 International Conference on Software Engineering (ICSE 2013), pp. 722–731. IEEE Press (2013)
44. Radfelder, O., Gogolla, M.: On better understanding UML diagrams through interactive three-dimensional visualization and animation. In: AVI. ACM (2000)
45. Scheer, A.-W., Nüttgens, M.: ARIS Architecture and Reference Models for Business Process Management. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 376–389. Springer, Heidelberg (2000)
46. Sol, E., Harel, D., Cohen, I.R.: Reactive animation: Realistic modeling of complex dynamic systems. Computer 38(1), 38–47 (2005)
47. Stikkolorum, D.R., Stevenson, C.E., Chaudron, M.R.V.: Assessing software design skills and their relation with reasoning skills. In: EduSymp 2013. CEUR, vol. 1134, paper 5 (2013)
48. Utting, M., Pretschner, A., Legeard, B.: A taxonomy of model-based testing approaches. Journal on Softw. Testing, Verification & Reliability 22(5), 297–312 (2006)
49. van der Aalst, W.M.P., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
50. Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., Heldal, R.: Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) MODELS 2013. LNCS, vol. 8107, pp. 1–17. Springer, Heidelberg (2013)
51. Whittle, J., Hutchinson, J., Rouncefield, M.: The State of Practice in Model-Driven Engineering. IEEE Software 31(3), 79–85 (2014)
52. Wu, Y., Hernandez, F., Ortega, F., Clarke, P.J., France, R.: Measuring the effort for creating and using domain-specific models. In: Wksh. on Domain-Spec. Mod., p. 14. ACM (2010)