# A Hitchhiker's Guide to Model-Driven Engineering for Data-Centric Systems

Benoit Combemale[*], Jörg Kienzle[†], Gunter Mussbacher[†], Hyacinth Ali[†], Daniel Amyot[‡]
Mojtaba Bagherzadeh[§], Edouard Batot[¶], Nelly Bencomo[‖], Benjamin Benni[**], Jean-Michel Bruel[††]
Jordi Cabot[‡‡], Betty H.C. Cheng[x], Philippe Collet[**], Gregor Engels[xi], Robert Heinrich[xii]
Jean-Marc Jézéquel[xiii], Anne Koziolek[xii], Sébastien Mosser[xiv], Ralf Reussner[xii], Houari Sahraoui[¶]
Rijul Saini[†], June Sallou[xv], Serge Stinckwich[xvi], Eugene Syriani[¶], Manuel Wimmer[xvii]
[*] Univ. Toulouse & Inria, France, benoit.combemale@inria.fr
[†] McGill University, Canada, joerg.kienzle,gunter.mussbacher@mcgill.ca, hyacinth.ali,rijul.saini@mail.mcgill.ca
[‡]University of Ottawa, Canada, damyot@uottawa.ca
[§]Queens University, Canada, mojtaba@cs.queensu.ca
[¶] Université de Montréal, Canada, batotedo,sahraouh,syriani@iro.umontreal.ca
[‖] Aston University, United Kingdom, nelly@acm.org
[**] Université de la Côte d'Azur, I3S CNRS, France, benni,collet@i3s.unice.fr
[††] Université de Toulouse & IRIT, France, bruel@irit.fr
[‡‡] ICREA & UOC, Spain, jordi.cabot@icrea.cat
[x] Michigan State University, USA, chengb@msu.edu
[xi] Paderborn University, Germany, engels@upb.de
[xii] Karlsruhe Institute of Technology, Germany, heinrich,koziolek,reussner@kit.edu
[xiii] Université de Rennes, Inria, CNRS, IRISA, France, jezequel@irisa.fr
[xiv] Université du Québec à Montréal, Canada, mosser.sebastien@uqam.ca
[xv] Université de Rennes, Inria, CNRS, IRISA & Géosciences Rennes, OSUR, France, june.benvegnu-sallou@univ-rennes1.fr
[xvi] United Nations University Institute in Macau, Macau SAR, China, stinckwich@unu.edu
[xvii] Johannes Kepler University Linz, Austria, manuel.wimmer@jku.at

## I. Introduction

A broad spectrum of application domains are increasingly making use of heterogeneous and large volumes of data with varying degrees of humans in the loop. The recent success of Artificial Intelligence (AI) and, in particular, Machine Learning (ML) further amplifies the relevance of data in the development, maintenance, evolution, and execution management of systems built with model-driven engineering techniques. Applications include critical infrastructure areas such as intelligent transportation, smart energy management, public healthcare, and emergency and disaster management; many of these systems are considered socio-technical systems given the human, social, and organizational factors that must be considered during the system life-cycle [1]. This article introduces a conceptual reference framework – the *Models and Data* (MODA) framework – to support a data-centric and model-driven approach for the integration of heterogeneous models and their respective data for the entire life-cycle of socio-technical systems.

While system development within these diverse application domains makes use of both models and data, it differs in the types and uses of models and data, and the degree and role of humans in the loop. *E.g.,* ML models are applied to independently-collected large (training) datasets to produce decision models (*e.g.,* image classification for automotive obstacle avoidance). In contrast, software engineers use extensive data (*e.g.,* domain knowledge) to develop engineering models (*e.g.,* domain and design models), and then evolve them incrementally. Furthermore, scientific modelling typically starts with a mathematical model of a given physical phenomenon (*e.g.,* Navier-Stokes equations to describe fluid motions) and use application-specific data to calibrate this general model for a particular application (*e.g.,* a flood management system). While the combination of data and models originated from diverse application domains, we are starting to see attempts to integrate them into large socio-technical systems that involve these types of models and data coming from different communities [2], [3]. Such an integration requires the individual strengths of models and data as well as their complementing natures to be harnessed.

MODA supports the system life-cycle of socio-technical systems, and is intended to handle a broad range of stakeholders and community groups. The framework provides a vision for how to explicitly integrate the three roles played by models – prescriptive, predictive, and descriptive [4] – as well as their respective data sources and highlights related actions to integrate them.

An overarching objective of this article is to emphasize and illustrate the *complementarity* and *duality* of models and data in socio-technical, likely software-intensive, systems. In addition to reviewing the types of models used in the scientific, engineering, and artificial intelligence communities, highlighting their common roles and complementarities, we demystify the ways such models and data can be synergistically used in the system life-cycle of socio-technical systems with the MODA framework. While MODA was developed to capture emerging socio-technical systems with different model types, it is sufficiently expressive to characterize and generalize existing complex engineering practices and technologies in the system life-cycle, which we demonstrate with thirteen well-known processes, technologies, and systems. We also identify several key research challenges with proposed follow-on studies for realizing MODA.

We expect the MODA framework to be used *(i)* as a guide for educational purposes to clarify roles of models, data sources, and related actions during a wide variety of system life-cycles; *(ii)* to organize and compare complex engineering processes and technologies to support critical engineering choices involving large landscapes (*e.g.,* smart grid systems); and *(iii)* to situate existing research approaches or initiate new research agendas to improve the integration of different types of models and data sources (*e.g.,* air quality management, climate change).

## II. On Models and Data

The word *model* is used in many communities, for a good reason: they share a common definition. A model is an abstraction of an aspect of reality for a given purpose. Models can be used to answer questions with responses that are sufficiently close to reality. Beyond this common definition, different types of models have commonalities, but also notable differences, which are not yet fully understood. Existing work has discussed various types of models [5] and the roles they can play [4]. Here we study these notions of model types and roles with respect to their interplay with the available data. As a result, we provide a conceptual framework that demonstrates the relevant combinations of the different roles of models in engineering and scientific processes.

In this article, we concentrate on three main types of models: *Engineering*, *Scientific*, and *Machine Learning* models.

### *A. Types of Models*

*1) Engineering Model:* Models in engineering disciplines are devoted to support the definition and representation of a targeted system [5]. Engineering models represent concerns ranging from onboard control in autonomous vehicles for braking and obstacle avoidance, to traffic management models, information systems, business rules, etc. They are meant to drive, possibly with some degree of automation, the development of the system-to-be.

Engineering disciplines often use systematic processes and methods in addition to well-defined notations for their models. Those formalisms can be domain-specific (*e.g.,* BPMN or BPEL) or more generic (*e.g.,* UML or SysML). With these formal processes and languages, validation of the models includes the use of formal techniques, simulations, and tests.

Engineering models can represent a means to develop a physical system for a specific purpose that obeys physical laws, or a software-based system (including behavior, structure, intentions, and/or configuration), or both (*e.g.,* cyber-physical systems). As such, we also consider engineering models to be those that use decision logic to manage a system based on inputs from environment sensors to determine an appropriate response. In many cases, a feedback loop is used to govern how to adjust the response to account for different types of uncertainty (*e.g.,* errors, changing operational conditions).

During the design of a feedback loop, engineering models must keep track of key elements of the system (*e.g.,* physical and logical elements), which requires processing large volumes of data. *E.g.,* when engineering models are used to control their environment, they have to be able to handle continuous data as opposed to discrete data.

*2) Scientific Model:* A scientific model is a representation of some aspects of a phenomenon of the world [6]. It is used to explain and analyze the phenomenon (*e.g.,* define, quantify, visualize, or simulate), based on established scientific knowledge defining a theory. A theory provides a framework with which models of specific phenomena and systems can be constructed. Models are validated or rejected by experiments or known theories. Upon validation, these models are typically used to predict future behavior of the system through simulation or mathematical calculus. Different types of models are used for different aims: conceptual models to improve shared understanding, operational models to refine measurement, mathematical models to quantify a subject, or graphical models to visualize the subject. A holistic view of a phenomenon or a system is assumed and different models can be used at different time- or space-scales.

Scientific models encompass a wide range of representations, such as climate change models, electromagnetic models, protein synthesis models, or metabolic network models. Scientific models typically involve equation-based continuous formalisms such as differential equations, as well as discrete models (*e.g.,* state-based, event-driven, or agent-based models).

Data can be numerical or symbolic (*e.g.,* DNA nucleotides). Data is collected, produced, manipulated, and exploited in several ways at different points of the scientific method life-cycle. *E.g.,* observation data can be curated and then used in a calibration phase to set the parameters of a model; data can be directly processed by a model; or data can be produced as a result of model simulations.

*3) Machine Learning Model:* ML models are produced by automated learning algorithms out of sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to perform the task. They can be seen as an approximation of the conceptual relationship between a particular input and the expected or a priori unknown target output.

ML models are used for a wide range of applications, such as image classification, feature extraction, defect density prediction, language translation, or motion planning of robots. Common formalisms include neural networks, Bayesian classifiers, statistical models (*e.g.,* linear regression), and many others.

ML models are obtained according to the inductive reasoning principle, *i.e.,* generalization from specific cases. This approach implies a certain degree of uncertainty as to whether the specific cases sufficiently represent the rules and principles an ML model is intended to capture. The kind of data used in ML is mostly numeric for regression problems, but also symbolic for classification problems.

### *B. Complementarities and Synergies*

By definition, a model has a purpose, and thus plays one or more roles with respect to that purpose. A model plays:

- a *descriptive* role if it documents some current or past aspect of the system under study (which can be a software-intensive system or a natural system), facilitating understanding, and enabling analysis.
- a *prescriptive* role if it is a description of the system to be built, driving the constructive process, including runtime evolution in the case of self-adaptive systems (aka models@runtime).
- a *predictive* role if it is used to predict information that one cannot or does not want to measure (which creates new knowledge and allows decision-making and trade-off analyses to be performed).

Each model type can play more than one role. A scientific model is first descriptive, but its main objective is to become predictive supporting what-if scenarios [7]. Embedded into a socio-technical system, it becomes prescriptive. *E.g.,* consider a prescriptive model of a decision-making tool for climate change using a predictive simulator based on a descriptive scientific model of the earth's water cycle. An engineering model typically starts by being descriptive (*e.g.,* a domain model describing key concepts and relationships), and then at design time is refined/transformed into a prescriptive model. But once the system is built as prescribed, the model becomes descriptive again as a form of documentation [8]. An engineering model can also be used as a predictive model: *e.g.,* an architecture model could be used to predict the performance of a specific configuration. An ML model is mostly used in a predictive role with the objective to infer new knowledge given some hypothetical input data. It might also be descriptive of a current or past relationship, or prescriptive if the results are used to make decisions. *E.g.,* consider a prescriptive model of a smart farm where a predictive ML model is used to decide on irrigation plans based on descriptive historical data.

To create a model of any of the above types, knowledge *and* data are needed as input. The proportion of required knowledge or the importance of the availability of the required data to build the models are highly specific to each model type. *E.g.,* in ML models, we need problem-specific knowledge to choose the adequate ML technique(s), choose the ML meta-parameters (*e.g.,* different kinds of layers and how they connect in a neural network), choose the input variables and the output variables, and then derive a specialized model from the data. In scientific models, knowledge formulates a hypothesis while data parameterizes the model. In engineering models, we mainly use knowledge about the domain and possibly improve or tune the models with data.

Along with the respective importance of knowledge and data in the process of building the models, the order in which models and data are considered is specific to the type of models. Descriptive engineering models primarily start with data, including external data (*e.g.,* expert/domain knowledge expressed in requirements or constraints) or measured data (*e.g.,* exploitation data from previous systems). Engineering models are then used to prescribe the way the future system will be built. With knowledge about which algorithms are best suited for a problem, ML typically starts with input/output system data or measured data for training, and iteratively (*e.g.,* with feedback loops) revises the model to address the problem at hand, where the resulting models are the main output of the process. In scientific models, the external data (*e.g.,* real-world observations) plays important roles while off-the-shelf models aim to describe existing phenomena and hence are regularly updated and improved.

## III. The MODA Framework

Life-cycle support for current and future complex socio-technical, likely software-intensive, systems requires us to synergistically combine this range of models through well-founded techniques that leverage their overall benefits to satisfy many different purposes. In order to support this integration through engineering processes, we describe a conceptual *Models and Data* (MODA) framework that explicitly relates the different roles of the model types according to three kinds of data: input/output data, measured data, and external data. The MODA framework provides insight into the integration of the different roles that various model types play, including their data sources and related actions, resulting in a generalized view of common software development processes, technologies, and systems.

The systems we consider range from software-intensive systems (where software is the predominant component, *e.g.,* e-commerce applications) and cyber-physical systems (where software controls physical components, *e.g.,* smart grids) to more general socio-technical systems (where humans are in the loop with software-based systems, *e.g.,* crisis management systems). These systems are data-centric. Data is not only provided to and produced by the running system, but data about the software itself and its surrounding environment is collected (*e.g.,* performance data). All this data is processed by descriptive, predictive, and prescriptive models in order to adapt the system to handle the evolving data.

Figure 1 presents the MODA framework. The running software is depicted in red, different kinds of data are shown

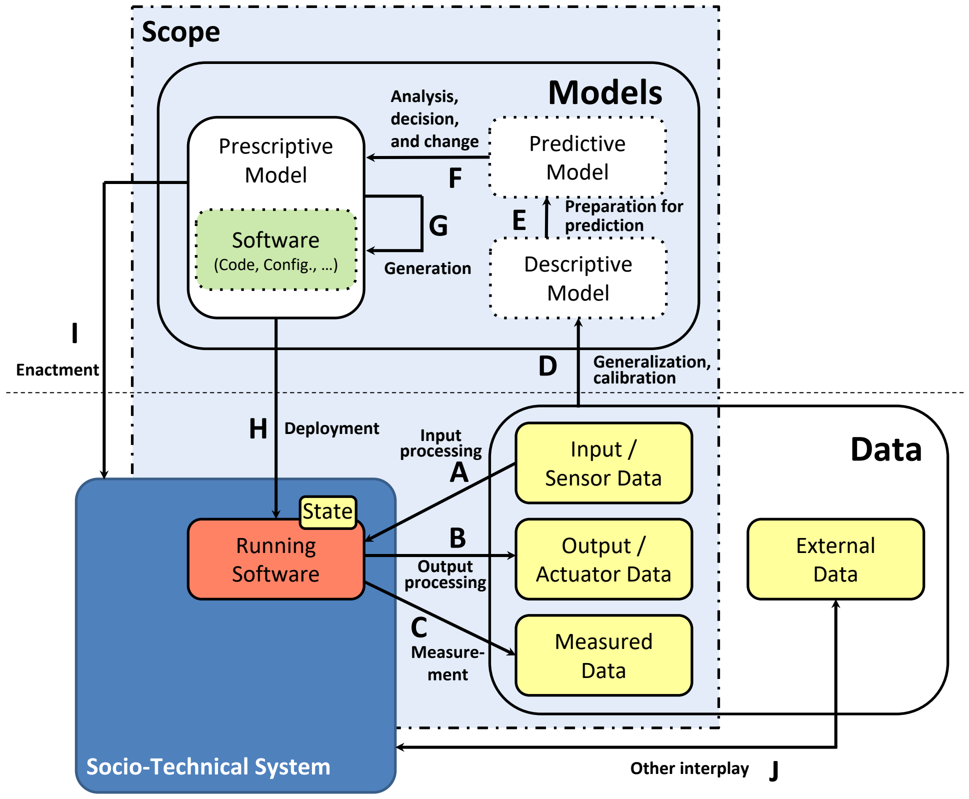

Figure 1. The MODA Framework (dotted boxes are optional)

in yellow, different model roles are represented in white, and the arrows represent actions related to the models and data. Consider, *e.g.,* a *Crisis Management System* (CMS) intended to provide a responsive means to systematically detect crises and deploy resources to mitigate them (*e.g.,* traffic accidents). In this case, the running software typically comprises a distributed system with one or multiple backends and databases, sensors, and applications running on the smart devices of the different CMS participants (first responders, drivers, etc.), and vehicles.

The running software processes input data and generates output, depicted by the arrows $A$ and $B$ labelled *Input processing* and *Output processing*, respectively. For CMS, input data includes information gathered from phone calls (*e.g.,* number of vehicles involved, affected area, fire), and information gathered from sensors (*e.g.,* GPS data gathered from workers, vehicles, cameras, weather information). Output data includes resource assignments and mission-related information communicated to activate emergency personnel, as well as information and requested actions sent to other systems (*e.g.,* the police).

The $C$ arrow labelled *Measurement* represents the gathering of metadata or metrics about the running software. Gathering such data requires additional effort, ranging from code instrumentation and logging to human auditing. The gathered data might be filtered or aggregated in real-time, as well as stored for offline use. For CMS, possible metadata could include performance measurements, resource usage (*e.g.,* used network bandwidth), reliability data (*e.g.,* time-to-failure, noise in communication), and detected intrusions.

The last kind of data we distinguish, external data, is any kind of information that is not explicitly within the scope of the software in the current version of the system. For CMS, *e.g.,* historical data about past emergencies or social media data could be considered to improve crisis management ($J$ arrow).

The $D$ arrow labelled *Generalization, calibration* represents techniques that generalize from the different kinds of data to yield a descriptive model. These techniques include conceptual generalization approaches such as abstraction, synthesis, and type induction, but also statistical approaches, regression, differential equation inference, complex event processing (*e.g.,* aggregating many small events into semantically meaningful ones), mining, as well as natural language processing and advanced machine learning techniques. The generalization can happen in real-time (*e.g.,* for adaptive systems), or offline.

Building a CMS descriptive model includes matching the received data about an event with generic crisis templates to classify the unfolding crisis according to well-known crisis types. Subsequently, the templates are parameterized with specific event information, *e.g.,* the number of victims and vehicles on fire. Models of historical information about crises could be generated by mining historical data. Scientific models, *e.g.,* fire propagation modelling, modelling of physical roadway condition in response to different weather conditions, can be built to help assess the situation. ML techniques can be applied to analyze traffic patterns.

The $E$ arrow (*Preparation for prediction*) refers to preprocessing techniques that combine data and descriptive models to build models that can be used to make predictions. Sometimes descriptive models can be used directly to make predictions (*e.g.,* fault tree analysis). More often, additional processing is required, *e.g.,* applying techniques for inter- and intrapolation, using statistical techniques (regression), preparing for simulation and training of ML models, *e.g.,* neural networks. For CMS, one might prepare a queuing network and then choose and parameterize a simulator to make traffic predictions. Neural networks could be trained to discover hidden behavioral patterns that can be used to prevent potential future accidents or shorten the emergency response time.

The $F$ arrow (*Analysis, decision, and change*) represents decision support activities (*e.g.,* what-if analysis) and the application of consequent changes to the prescriptive model. For CMS, prescriptive models would include mission workflows, safety regulations that should be enforced, UML descriptions of the software solution, algorithms supporting communication, etc. What-if scenarios can be run manually or in an automated fashion (*e.g.,* using hill-climbing / optimizing searches, genetic algorithms). Different approaches can be used for enacting a decision. *E.g.,* in self-adaptive systems, a decision might require a reconfiguration that can be achieved by making changes to the prescriptive architecture model (*e.g.,* by means of model transformations), or by updating configuration files. In a software product line setting, foreseen adaptations can be achieved by selecting features that describe previously-designed alternatives, and then adapting the prescriptive model, *e.g.,* by model weaving or merging. For CMS, one could predict that the network noise will become significant in the near future and then trigger the decision to switch to a more robust communication protocol at runtime. Offline extrapolation of current crises data predicting future needs of the CMS might lead to the decision to develop new features for the next version of the CMS software.

The $G$ arrow labelled *Generation* represents the typical software development activities that use high-level prescriptive models (*e.g.,* requirements models) to produce lower-level prescriptive models (*e.g.,* design models or executable code). The techniques used here include model transformations, model instantiation, and compilers. Recently, AI techniques have also been used in this step to optimize the generation process.

The $H$ arrow (*Deployment*) involves deploying and executing or interpreting the low-level, executable models (*e.g.,* code). Here as well, AI techniques are beginning to emerge, *e.g.,* to optimize node configurations in cloud deployments. For CMS, a new architecture model could be distributed to all system nodes to switch to a new communication protocol.

Finally, the $I$ arrow (*Enactment*) represents actions accomplished or enforced in a socio-technical system based on prescriptive models involving human/social dimensions (*e.g.,* policies, laws, standards). For CMS, new driving regulations could be devised that force truck drivers to adhere to stronger safety requirements. Such regulations would have to be enforced by legal means. Similarly, transparent information dissemination policies could be put in place to strengthen the population's feeling of security.

Besides socio-technical systems like CMS, MODA generalizes state-of-practice processes, technologies, and other systems. Figure 2 highlights MODA's broad applicability with representative instantiations of MODA. We show three different software development processes (waterfall, iterative/agile, and test-driven development; Fig. 2a-c), as well as business process modelling and mining approaches (Fig. 2d). The generic nature of MODA even enables its use as an underlying structure for explaining business modelling approaches (Fig. 2e). In Fig. 2f-h, we consider workflows in scientific computing [9], commonly-used machine learning pipelines in software development, and the autonomic computing MAPE-K loop [10]. Note that recommender systems are similar to machine learning pipelines in that a *predictive model* is included in a *prescriptive model* to provide recommendations automatically. Finally, we consider the development of four systems (Fig. 2i-l), differing in terms of complexity, availability of data, and requirements volatility: *(i)* a simple mobile app, *(ii)* a control and command system in an airplane, *(iii)* a digital twin application, and *(iv)* a smart power grid application. Exemplified by the above processes, technologies, and systems, we have illustrated how the MODA framework is a common reference to guide the use of models, data sources, and their implied actions to improve the integration of different model roles and data sources.

## IV. Conclusion and Perspectives

A key objective of software engineering (SE) research in the near future is to enable an engineering-based approach to support rigorous processes and techniques for model and data integration for the increasingly complex and dynamic socio-technical systems of tomorrow. To date, SE researchers have used AI as a tool to support SE tasks (*e.g.,* to improve testing techniques) or applied SE to AI (*e.g.,* to test AI software). Beyond this bidirectional use, this article focuses on relating the *fundamental role* of models obtained through AI to the fundamental roles of both scientific and engineering models.

We introduce the *MODA framework* in this article as a conceptual reference framework that provides the foundations for identifying the various models and their respective roles within any model-based system development life-cycle. It is intended to be a guide to organize the various models in data-centric socio-technical systems.

Such a framework also facilitates the identification of open challenges that need to be addressed in the near future. We mention some of these challenges in the following and organize them according to the framework's arrows $C$ to $F$ in Fig. 1. This list is not an exhaustive treatment of all challenges (*e.g.,* quality attributes in general, and ethical considerations of ML in particular, are not discussed). In general, to make

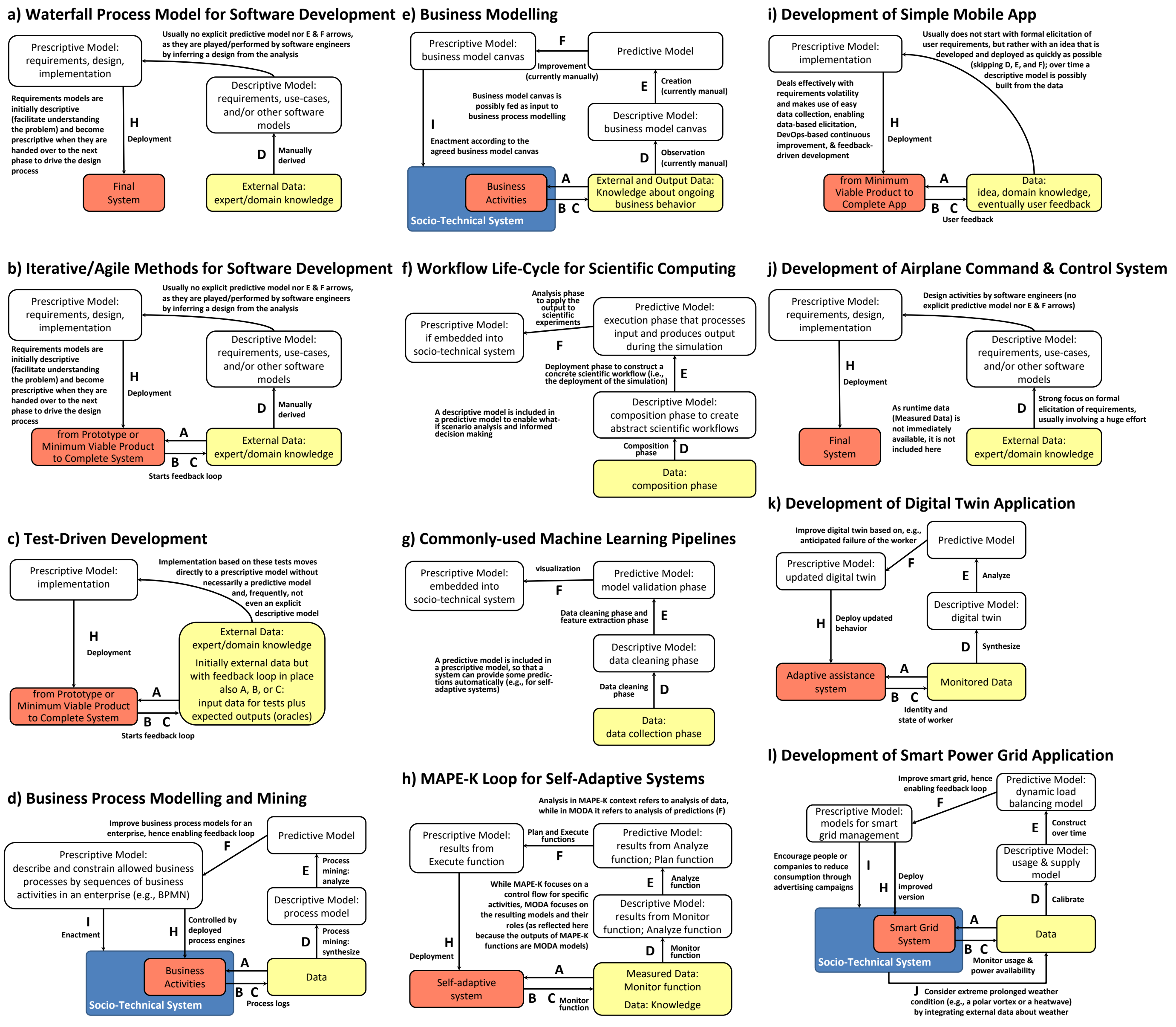
a) Waterfall Process Model for Software Development
Prescriptive Model: requirements, design, implementation
Usually no explicit predictive model nor E & F arrows, as they are played/performed by software engineers by inferring a design from the analysis
Descriptive Model: requirements, use-cases, and/or other software models
Requirements models are initially descriptive (facilitate understanding the problem) and become prescriptive when they are handed over to the next phase to drive the design process
H Deployment
D Manually derived
Final System
External Data: expert/domain knowledge
b) Iterative/Agile Methods for Software Development
Prescriptive Model: requirements, design, implementation
Usually no explicit predictive model nor E & F arrows, as they are played/performed by software engineers by inferring a design from the analysis
Descriptive Model: requirements, use-cases, and/or other software models
Requirements models are initially descriptive (facilitate understanding the problem) and become prescriptive when they are handed over to the next phase to drive the design process
H Deployment
D Manually derived
from Prototype or Minimum Viable Product to Complete System
A
B C
External Data: expert/domain knowledge
Starts feedback loop
c) Test-Driven Development
Prescriptive Model: implementation
Implementation based on these tests moves directly to a prescriptive model without necessarily a predictive model and, frequently, not even an explicit descriptive model
H Deployment
External Data: expert/domain knowledge
Initially external data but with feedback loop in place also A, B, or C: input data for tests plus expected outputs (oracles)
from Prototype or Minimum Viable Product to Complete System
A
B C
Starts feedback loop
d) Business Process Modelling and Mining
Improve business process models for an enterprise, hence enabling feedback loop
F
Predictive Model
Prescriptive Model: describe and constrain allowed business processes by sequences of business activities in an enterprise (e.g., BPMN)
E Process mining: analyze
Descriptive Model: process model
I Enactment
H Controlled by deployed process engines
D Process mining: synthesize
Business Activities
Socio-Technical System
A
B C
Data
Process logs
e) Business Modelling
Prescriptive Model: business model canvas
F
Predictive Model
Improvement (currently manually)
E Creation (currently manual)
Business model canvas is possibly fed as input to business process modelling
Descriptive Model: business model canvas
I Enactment according to the agreed business model canvas
D Observation (currently manual)
Business Activities
A
B C
External and Output Data: Knowledge about ongoing business behavior
Socio-Technical System
f) Workflow Life-Cycle for Scientific Computing
Analysis phase to apply the output to scientific experiments
Prescriptive Model: if embedded into socio-technical system
F
Predictive Model: execution phase that processes input and produces output during the simulation
Deployment phase to construct a concrete scientific workflow (i.e., the deployment of the simulation)
E
A descriptive model is included in a predictive model to enable what-if scenario analysis and informed decision making
Descriptive Model: composition phase to create abstract scientific workflows
Composition phase
D
Data: composition phase
g) Commonly-used Machine Learning Pipelines
Prescriptive Model: embedded into socio-technical system
visualization
F
Predictive Model: model validation phase
Data cleaning phase and feature extraction phase
E
A predictive model is included in a prescriptive model, so that a system can provide some predictions automatically (e.g., for self-adaptive systems)
Descriptive Model: data cleaning phase
Data cleaning phase
D
Data: data collection phase
h) MAPE-K Loop for Self-Adaptive Systems
Analysis in MAPE-K context refers to analysis of data, while in MODA it refers to analysis of predictions (F)
Prescriptive Model: results from Execute function
Plan and Execute functions
F
Predictive Model: results from Analyze function; Plan function
While MAPE-K focuses on a control flow for specific activities, MODA focuses on the resulting models and their roles (as reflected here because the outputs of MAPE-K functions are MODA models)
E Analyze function
Descriptive Model: results from Monitor function; Analyze function
H Deployment
D Monitor function
Self-adaptive system
A
B C Monitor function
Measured Data: Monitor function
Data: Knowledge
i) Development of Simple Mobile App
Prescriptive Model: implementation
Usually does not start with formal elicitation of user requirements, but rather with an idea that is developed and deployed as quickly as possible (skipping D, E, and F); over time a descriptive model is possibly built from the data
Deals effectively with requirements volatility and makes use of easy data collection, enabling data-based elicitation, DevOps-based continuous improvement, & feedback-driven development
H Deployment
from Minimum Viable Product to Complete App
A
B C
User feedback
Data: idea, domain knowledge, eventually user feedback
j) Development of Airplane Command & Control System
Prescriptive Model: requirements, design, implementation
Design activities by software engineers (no explicit predictive model nor E & F arrows)
Descriptive Model: requirements, use-cases, and/or other software models
H Deployment
As runtime data (Measured Data) is not immediately available, it is not included here
D Strong focus on formal elicitation of requirements, usually involving a huge effort
Final System
External Data: expert/domain knowledge
k) Development of Digital Twin Application
Improve digital twin based on, e.g., anticipated failure of the worker
Predictive Model
Prescriptive Model: updated digital twin
F
E Analyze
Descriptive Model: digital twin
H Deploy updated behavior
D Synthesize
Adaptive assistance system
A
B C
Monitored Data
Identity and state of worker
l) Development of Smart Power Grid Application
Improve smart grid, hence enabling feedback loop
Predictive Model: dynamic load balancing model
Prescriptive Model: models for smart grid management
F
E Construct over time
Descriptive Model: usage & supply model
Encourage people or companies to reduce consumption through advertising campaigns
I
H Deploy improved version
D Calibrate
Smart Grid System
Socio-Technical System
A
B C
Data
Monitor usage & power availability
J Consider extreme prolonged weather condition (e.g., a polar vortex or a heatwave) by integrating external data about weather

Figure 2. Instantiations of the MODA Framework

MODA effective, efforts are required in *foundational support* for data and model interplay (*e.g.,* protocols and interfaces).

One challenge deals with questions on *guidelines regarding when, what, and how (i) to observe* (*e.g.,* systematic methods to derive (arrow $D$) descriptive models out of observed data) or *(ii) to measure running software* (*e.g.,* need metrics that take into account the measured data (arrow $C$) such as data related to performance, load, and execution time). These issues are similar to the monitoring issues faced by the self-adaptive systems community [11] and the models@runtime approach [12]. However, MODA provides a more accurate explanation of the different types of models, and thus of the questions to be answered: what are the methods needed to systematically design the data processing pipeline from observations to decisions? How can we control data quality through the entire processing pipeline? How can established ML techniques be used to support design decisions? What are the optimal uses and constraints related to online-training and offline-training? How can ML techniques be used in data processing that purely runs online (*e.g.,* observation process), as measurement overhead needs to be kept low?

It is also important to *help determine when the different types of models must be made explicit within the process* (*e.g.,* when is it beneficial to have an explicit scientific model – in addition to an engineering model – as a descriptive model?), and *elaborate semi-automated model transformations that assist the developers in accomplishing arrows D, E, and F*. This need requires a deep understanding of what kinds of factors affect the interplay between a descriptive model and a predictive model (*i.e.,* certain descriptive models make the prediction easier, but their creation requires considerable effort), and how to learn complex models with ML techniques (*i.e.,* models that process complex inputs and/or produce complex outputs). In this context, it is crucial to identify the limitations of the different types of models used in a system. Techniques need to be defined to mitigate these limitations and still be able to provide high-level guarantees. A plethora of model integration work exists mainly for engineering models [13]. Recent work has attempted to integrate ML and SE in Differentiable Programming [14], and ML and scientific models to define a Theory-Guided Data Science [15]. MODA goes further, though, as integration of the three different kinds of models requires a common understanding, and we envision that new kinds of model interfaces will need to be developed to address this heterogeneity challenge.

Finally, systematic methods are needed for the *operationalization of decision making that apply predictive models to improve prescriptive models of the system* (arrow $F$), while ensuring important prescribed properties (*e.g.,* safety, security). MODA can again be used to pose relevant research questions. How can we combine useful knowledge extracted from observations of varying nature (traceability information, quality measures, structural/environment constraints) with previous/external knowledge in order to refine the predictive model and enhance or adapt the prescriptive model? How can we systematically deal with data uncertainty (either coming from uncertain data in the descriptive model or from the predictive model)?

We envision the MODA framework to be used as a hitchhiker's guide to explain, organize, and compare complex engineering processes, software development life-cycles, system life-cycles, and technologies, while creating a research momentum to address the open challenges. Finally, to all those who wonder whether the MODA framework is a descriptive, predictive, or prescriptive model, the answer is... 42!

## References

[1] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interacting with computers*, vol. 23, no. 1, pp. 4–17, 2011.

[2] R. Ashri, "Building AI software: Data-driven vs model-driven AI and why we need an AI-specific software development paradigm," https://bit.ly/2X6vu9H, 2018, hacker Noon. Last access: 2019-02-06.

[3] R. Yang and G. Rizzoni, "Comparison of model-based vs. data-driven methods for fault detection and isolation in engine idle speed control system," in *PMH*. Prognostics and Health Management Society, 2016.

[4] T. Kühne, "Unifying explanatory and constructive modeling: towards removing the gulf between ontologies and conceptual models," in *MODELS 2016*. ACM, 2016, pp. 95–102.

[5] E. A. Lee, "Modeling in engineering and science," *Commun. ACM*, vol. 62, no. 1, pp. 35–36, Dec. 2018.

[6] P. Gerlee and T. Lundh, *Scientific Models*. Springer, 2016.

[7] J.-M. Bruel, B. Combemale, I. Ober, and H. Raynal, "MDE in Practice for Computational Science," in *ICCS 2015*, Jun. 2015.

[8] R. Heinrich, R. Jung, C. Zirkelbach, W. Hasselbring, and R. Reussner, *Software Architecture for Big Data and the Cloud*. Morgan Kaufmann, 2017, ch. An Architectural Model-Based Approach to Quality-aware DevOps in Cloud Applications, pp. 69–89.

[9] J. Liu, E. Pacitti, P. Valduriez, and M. Mattosa, "A survey of data-intensive scientific workflow management," *Grid Computing*, vol. 13, pp. 457–493, 2015.

[10] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan 2003.

[11] R. De Lemos, H. Giese, H. A. Müller *et al.*, "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, pp. 1–32.

[12] N. Bencomo, R. B. France, B. H. C. Cheng, and U. Aßmann, *Models@run.time: foundations, applications, and roadmaps*. Springer, 2014.

[13] J. Kienzle, G. Mussbacher, B. Combemale, and J. DeAntoni, "A unifying framework for homogeneous model composition," *Software and Systems Modeling*, vol. 18, no. 5, pp. 3005–3023, 2019. [Online]. Available: https://doi.org/10.1007/s10270-018-00707-8

[14] G. Baudart, M. Hirzel, and L. Mandel, "Deep probabilistic programming languages: A qualitative study," *CoRR*, vol. abs/1804.06458, 2018.

[15] A. Karpatne, G. Atluri, J. H. Faghmous *et al.*, "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE Trans Knowl Data Eng*, vol. 29, no. 10, pp. 2318–2331, Oct 2017.