# Example-Driven Modeling
Model = Abstractions + Examples

Kacper Bąk, Dina Zayan, Krzysztof Czarnecki, Michał Antkiewicz, Zinovy Diskin
GSD Lab, University of Waterloo, Canada
{kbak,dzayan,kczarnec,mantkiew,zdiskin}@gsd.uwaterloo.ca

Andrzej Wąsowski
IT University of Copenhagen, Denmark
wasowski@itu.dk

Derek Rayside
University of Waterloo, Canada
drayside@uwaterloo.ca

***Abstract*—We propose Example-Driven Modeling (EDM), an approach that systematically uses explicit examples for eliciting, modeling, verifying, and validating complex business knowledge. It emphasizes the use of explicit examples together with abstractions, both for presenting information and when exchanging models. We formulate hypotheses as to why modeling should include explicit examples, discuss how to use the examples, and the required tool support. Building upon results from cognitive psychology and software engineering, we challenge mainstream practices in structural modeling and suggest future directions.**

I. Introduction

Domain modeling is one of key activities in requirements engineering. It is about transferring knowledge from a Subject Matter Expert (SME) to Business Analyst (BA) and stating it explicitly in the form of documents, models, and code. Getting requirements right is hard because much of the knowledge is difficult to elicit and communicate [1]. Let us imagine a scenario where Alice is an SME and Bob is a BA. Alice hires Bob to build a system for booking meeting rooms in her organization. Bob is new to Alice's organization and needs to understand the domain first. The conversation goes as follows:

ALICE: Members of our organization often book rooms for meetings. Each meeting is organized by a chair and has at least one other participant. Each room has a maximum capacity it can hold. Depending on the room different equipment is available.

BOB: Please, give me examples of room bookings.

ALICE: The mid-year review meeting always takes place in June. It is organized by Steven, our chair, and is held in the meeting room C. The room provides a whiteboard and audio-conferencing equipment to include online participants.
Another example is an on-demand meeting organized within work hours. Joana, a team-leader, sometimes meets other team members in room D. They use a whiteboard and have no online participants.
[BOB *writes down the examples, learns the domain and abstracts concepts to come up with a model*].

BOB: So we have 4 main concepts: meeting, meeting room, equipment, and a member. The participants are on-site members or join remotely. There is one distinguished member: the chair. I have one question. Are all types of the equipment available in each meeting room? For example, can there be a meeting room without audio-conferencing equipment?

ALICE: Well, each room provides either a traditional or an electronic whiteboard. All other equipment is optional (such as projectors and computers). Only if the meeting includes remote participants, must the room offer at least audio-conferencing equipment.

BOB: Aha, an important constraint must be added!

The dialog shows how Alice transferred her domain knowledge to Bob by describing the concepts and giving examples. Bob created an initial model and validated it with Alice. Examples allowed Bob to clarify the requirements, and discover (together with Alice) an important domain constraint.

II. Example-Driven Modeling

Systematic use of models provides many benefits in software engineering, such as, improved communication, understanding of problems, productivity, and software quality [2]. Despite these benefits, models have failed to gain widespread adoption. The main reason for limited adoption is because they do not serve a wider audience. They are mostly used in narrow domains by highly trained experts [3] who work with abstractions. The majority of stakeholders, who would benefit from modeling, however, are not modeling experts. Yet they can understand and work with models via examples [2], [3].

The use of examples is critical to provide benefits of models to a wider population of stakeholders as it makes modeling more accessible to non-experts. We propose EDM, an approach where modeling is fundamentally based on examples. EDM relies on systematic use of explicit examples for eliciting, modeling, verifying, and validating domain knowledge. In contrast to current structural modeling practices, examples are as important as abstractions. In EDM, models comprise both *abstractions* and *examples*, where *abstractions* are mental representations dissociated from concrete objects, and *examples* are concrete instances of abstractions. Although abstractions or examples alone can represent concepts [4], the redundancy is useful in comprehension and verification.

EDM distinguishes two non-trivial activities that relate examples with abstractions: 1) Abstraction Inference (AI) – for synthesizing abstractions from a set of examples, 2) Example Derivation (ED) – for generating examples from abstractions. In the session between Alice and Bob, Bob performed AI to build models from the examples given by Alice. He used ED to verify the models on examples. Neither activity is given a higher priority, both are equally important and first class. Depending on the modeler and the situation either of the activities may be preferred to support modeling.

Common sense tells us that examples are useful in structural modeling. In fact, modelers use examples informally for building and explaining models. We are not aware, however, of a systematic approach and empirical studies that investigate the usefulness and the use of examples. We would like to address the following questions: *How useful examples are for building models? What is the impact of examples on comprehension of models? How to use examples systematically? What examples are more useful than others? What kind of tool support is needed to work with examples?* We assert that they should be studied on the conceptual basis of cognitive science, rather than addressed intuitively on the basis of common sense alone. Answering these questions may lead to new methods, languages, and tools that make modeling more accessible. The remainder of this paper explores why and how to use examples in EDM, with each subsection presenting a hypothesis, followed by a discussion and a short conclusion.

III. Why Base Modeling on Examples?

A. Constructing models with the aid of explicit examples improves the quality of models.

The session with Alice and Bob showed how Bob created a model from examples, and then validated it with Alice. He gave an example of a room without audio-conferencing equipment. Alice clarified that rooms hosting online meetings must have audio-conferencing equipment. Thanks to examples they were able to refine the initially underconstrained model.

Humans learn from examples by generalizing information and building mental models. Gick and Holyoak [5] showed that humans learn abstractions and solve analogical problems more effectively if the new material is presented with *multiple* examples. They also showed that multiple examples improve the quality of constructed abstractions. By analogy, we hypothesize that explicit examples help to build better models by capturing relevant details, scoping the models, and serving as test cases. They should have positive impact on a model's validity and the modelers' confidence that the model is valid.

Explicit examples should improve correctness and precision of models just as test cases lead to better software. Similar to test cases, examples naturally express corner cases. One of the corner cases that Bob touched upon was a room without audio-conferencing equipment hosting an online meeting. Thus, examples determine the scope of models and explicitate their applicability. They also capture the relevant details that SMEs recall from the real world. By investigating the examples a BA can question some details and generalize them. In the dialog, Alice mentioned Steven as a chair. Bob abstracted that information and distinguished *some* participant as a chair.

Black-box software testing uses test cases to verify software's observable behavior. Traditionally software testing was the last step of software development. Recent approaches, such as Test-Driven Design (TDD) [6] and Behavior-Driven Development (BDD) [7], encourage starting development by writing tests. TDD prescribes creating automated test cases before writing the actual code. BDD is based on TDD and focuses on providing a ubiquitous language among stakeholders to specify examples. Despite these advances in software testing, testing of structural models is a less common practice.

The importance of examples is well recognized and acknowledged formally in behavioral modeling methodologies. Examples typically show traces of execution, as in Spin [8], or are used as test cases [9], [10]. Some behavioral modeling methodologies are examples-based [11]. Structural modeling seems to underestimate the role of examples as compared to behavioral modeling. UML [12] provides object and class diagrams. Object diagrams are instances of class diagrams, thus they encode examples of the latter. In practice, class diagrams are much more frequently used than object diagrams [13]. The tool support for systematic use of explicit examples is limited to academic tools, such as Dresden OCL [14], MMUnit [15], and the USE tool [16]. The USE tool can verify if an object diagram is an instance of a class diagram, only recently providing ED, but has no support for AI. Modal Object Diagrams (MODs) [17] extend object diagrams with positive/negative examples. In contrast to UML, Alloy [18] and Formula [19] are designed to validate models via examples. Only recently was the need for an instance notation in Alloy recognized [20]. While Alloy goes from abstractions to examples, Modeling By Example (MBE) infers abstractions from examples [21]. However, it deals only with simple structural models.

Conclusion: Explicit examples have been successfully used in software testing and behavioral modeling. They help to find bugs and fix software and models. Yet they are rarely used in structural modeling. The structural modeling world needs evidence that the systematic use of examples improves the quality of models. It needs empirical studies showing defect reduction and maintenance gains as compared to modeling without explicit examples. Empirical evidence will guide the design of methods, languages, and tools. We envision tools that derive examples and infer rich structural models.

B. Augmenting models with explicit examples improves model comprehension among various stakeholders.

Stakeholders also need to understand models after they were created by others. Let us imagine that Alice sends the model to another SME who was not trained to deal with abstractions but understands the domain and examples very well. We hypothesize that if stakeholders know the phenomena shown by the examples, they can understand models more effectively.

The best knowledge transfer occurs when humans learn abstractions with examples [5], [22], [23], i.e., presenting just an example or an abstraction alone is insufficient for comprehension. The use of examples lowers mental effort required to understand problems [24]. While abstractions capture domain knowledge more completely than examples, they are difficult to comprehend as many business rules and constraints are implicit. Examples, on the other hand, capture intuition [23].

We hypothesize that relating abstractions to examples fosters 1) domain learning when the model is created and 2) model comprehension when the model is explained to others.

The structural modeling world has not looked at examples from the comprehension viewpoint. Although models are often meant to facilitate communication among shareholders [2], they are mostly understandable to a relatively small group of experts [3]. Many other stakeholders can comprehend models only by analyzing examples [2], [3], and, indeed, this is how explanations are provided to non-experts. The rare use of object diagrams [13] suggests that if examples are used at all, they are used informally. Examples are missing in structural methodologies and practices, which hinders comprehension. As a result (or a cause), the tools and languages do not seamlessly integrate abstractions and examples.

Conclusion: It is instrumental to empirically verify whether examples improve the comprehension of models among various stakeholders. A variety of studies should explore the use of examples while learning the domain and while explaining models to others. The results of studies may help to answer the questions *When? How much?* and *For whom?* the examples are mostly useful. If they confirm our hypothesis, it would be clear that new modeling notations should integrate examples into models. Furthermore, methodologies and practices would have to be redesigned to accommodate examples.

IV. How to Use Examples in EDM?

A. EDM starts either with abstractions or examples. A modeler typically goes back and forth between the two.

In the dialog, Alice gave examples of how the organization manages bookings. Bob learned the domain concepts and inferred the initial abstractions. Later they found out that rooms that host online participants must offer audio-conferencing equipment. Bob added the constraint to the initial abstraction.

Going back and forth between examples and abstractions is unavoidable. It is virtually impossible to construct and understand a non-trivial model without validating it on examples. Also, understanding a large set of examples without abstractions is equally hard. The process of going back and forth is incremental: abstractions and examples evolve together.

The use of examples varies between novices and experts: while novices need examples right-away, experts need them only for clarification [25], [26], [24]. The difference comes from the level of expertise, which in turn comes from the knowledge. According to Kalyuga [26] examples compensate for the missing or partially available knowledge. We hypothesize that BAs familiar with the domain start with abstractions and seek examples only for clarifications; BAs new to the domain elicit examples first and then infer abstractions.

Going from examples to abstractions requires tools that infer abstractions from a set of examples. There are techniques for inferring grammars [27], programs [28], data schemas [29], behavioral [30] and structural models [21]. They typically take examples and an initial abstraction. A major challenge is to infer constraints imposed by negative examples. None of the tools can handle constraints that represent complex business rules. Furthermore, co-evolution of examples and abstractions, although studied [31], [32], is not well supported in tools.

Support for partial examples is needed to enable flexible modeling and rapid verification and validation. Partial examples are underspecified model instances, i.e., they either have some variability (e.g., optional attributes) or uncertainty (e.g., undefined values). BAs should not have to fill in all the details of examples, but should specify only the properties that matter in a given context. AlloyPI [20] and Clafer [33] offer first-class support for expressing and completing partial examples.

Conclusion: EDM proposes to use examples in a new way as compared to mainstream structural modeling methods. The requirements for new tools push the boundaries of what is currently available and well supported. First, EDM assumes that different tools are required by BAs familiar and unfamiliar with the domain. The former need tools to derive examples from models (for validation and explanation), while the latter need tools that infer models from examples. EDM also requires support for partial examples, which are currently handled mostly by academic tools. Furthermore, simultaneous evolution of models and examples calls for tools that offer seamless, interactive support for such co-evolution during modeling.

B. A variety of generated examples leads to more effective model construction, comprehension, and validation.

In the dialog, Alice gave two examples of room booking. Both examples had a significant common part but also were slightly different. Both bookings included a chair (*Steven* and *Joana*, respectively), specified the room (*C* and *D*), and listed the equipment. The concrete values, such as chair names, differed. Those abstract commonalities and specific differences allowed Bob to infer a model that covers the two examples. Lack of critical differences between examples results in too concrete abstractions (e.g., room name always being *C*).

The choice of examples is important for the effectiveness of comprehension and knowledge transfer. According to Gick and Paterson, the most effective are *near-miss* contrasting examples [34]. They emphasize the critical differences, which helps humans build flexible abstractions. Too contrasting examples, however, are ineffective in spotting the critical differences and negatively impact knowledge transfer. We hypothesize that a specific, yet to be defined, variety of examples is needed to build, understand, and validate models effectively.

While positive examples specify correct model instances, negative examples represent disallowed instances, i.e., they can be abstracted as model constraints. Constraints reduce models' variability and uncertainty, thus making them more precise. A pair of near-miss examples may include a positive and a negative one. Understanding such pairs gives additional insight and helps to reason why and when the model is correct [35].

The need for a specific variety of examples has been recognized in software testing and grammar inference research. In the former, the selection of test cases impacts the effectiveness of the testing process [36]. Significantly different test cases provide better code coverage with fewer cases. While randomly-generated test cases do not guarantee that they are significantly different, antirandom testing does [37]. Similarly, significantly different examples lead to more effective discovery of model errors [38]. Grammar inference uses multiple examples and machine learning to infer a finite state automaton that defines the grammar [39]. The set of examples includes positive and negative examples. In modeling, object diagrams, as defined by UML, specify only positive examples.

Conclusion: Besides many technical improvements, there is still little known about examples in modeling from a user's viewpoint. We need empirical studies showing the impact of using a variety of examples on model construction, comprehension, and validation. The studies should investigate properties of individual examples (*What examples are useful? What are the benefits of adding negative examples? How to derive near-miss examples? How to find minimal examples?*), properties of the whole population (*Is it representative? Is it diverse enough? Is it minimal?*), and the ordering of example presentation to stakeholders (*How to effectively explore examples? How to show differences among examples?*).

V. Conclusion and Future Work

Developers have been using examples for decades to test software. Traditionally they used test cases to verify and validate the existing code. Modern approaches prescribe writing tests before writing the code. Examples are also used for verification and validation of behavioral models. The use of examples in structural modeling, however, is not prevalent. EDM emphasizes the importance of examples in such models and gives guidelines on how to apply them.

We see EDM as a promising way of improving the quality of structural models—such as, domain, data, variability, and component models and meta-models—and the effectiveness of communication among stakeholders. Even though our discussion focussed on structural modeling, an empirically-validated theory of examples will also likely help improve behavioral modeling. The next step is to define a method for evaluating each hypothesis and to design empirical studies. We encourage other researchers to join the effort to investigate the effectiveness of explicit examples. If the presented hypotheses are confirmed, this may lead to redefining models as inherent combinations of abstractions and examples, i.e., *Model = Abstractions + Examples*. The studies are likely to uncover weaknesses of the existing methods, languages, and tools, and also determine new requirements and improvements.

Acknowledgements

Thanks to Dan Berry and Tom Maibaum for their comments.

References

[1] M. A. Jackson, "Problems & Requirements," in *RE*, 1995.
[2] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen, "Empirical assessment of MDE in industry," in *ICSE*, 2011.
[3] B. Dobing and J. Parsons, "How UML is used," *CACM*, vol. 49, no. 5, 2006.
[4] E. E. Smith and D. L. Medin, *Categories and concepts*. Harvard University Press, 1981.
[5] M. L. Gick and K. J. Holyoak, "Schema induction and analogical transfer," *Cognitive Psychhology*, vol. 15, no. 38, 1983.
[6] D. Janzen and H. Saiedian, "Test-driven development concepts, taxonomy, and future direction," *Computer*, vol. 38, no. 9, 2005.
[7] D. North, "Introducing bdd," *Better Software*, vol. 12, 2011.
[8] G. J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*, 2004.
[9] J. Dick and A. Faivre, "Automating the generation and sequencing of test cases from model-based specifications," in *FME*, 1993.
[10] J. Offutt and A. Abdurazik, "Generating tests from UML specifications," in *UML*, 1999.
[11] D. Harel and R. Marelly, *Come, Let's Play: Scenario-Based Programming Using LSC's and the Play-Engine*. Springer-Verlag, 2003.
[12] OMG, *OMG Unified Modeling Language*, 2009.
[13] J. Whittle, "What do 449 MDE Practitioners Think About MDE?" in *MODELS*, 2011.
[14] B. Demuth, "The Dresden OCL Toolkit and its Role in Information Systems Development," in *ISD*, 2004.
[15] D. Sadilek and S. Weißleder, "Testing metamodels," in *ECMDA-FA*, 2008.
[16] M. Gogolla, F. Büttner, and M. Richters, "USE: a UML-Based Specification Environment for Validating UML and OCL," *SCP*, vol. 69, no. 1-3, 2007.
[17] S. Maoz, J. Ringert, and B. Rumpe, "Modal Object Diagrams," in *ECOOP*, 2011.
[18] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.
[19] E. K. Jackson, E. Kang, M. Dahlweid, D. Seifert, and T. Santen, "Components, platforms and possibilities: towards generic automation for MDA," in *EMSOFT*, 2010.
[20] V. Montaghami and D. Rayside, "Extending Alloy with Partial Instances," in *ABZ*, 2012.
[21] L. Mendel, "Modeling by example," Master's thesis, MIT, 2007.
[22] Z. Chen and M. W. Daehler, "Positive and negative transfer in analogical problem solving by 6-year-old children," *CD*, vol. 4, 1989.
[23] R. L. Goldstone and J. Y. Son, "The transfer of scientific principles using concrete and idealized simulations," *JLS*, vol. 14, no. 1, 2005.
[24] T. van Gog, L. Kester, and F. Paas, "Effects of worked examples, example-problem, and problem-example pairs on novices' learning," *CEP*, vol. 36, no. 3, 2011.
[25] M. T. H. Chi, P. J. Feltovich, and R. Glaser, "Categorization and representation of physics problems by experts and novices," *Cognitive Science*, vol. 5, no. 2, 1981.
[26] S. Kalyuga, "Knowledge elaboration: A cognitive load perspective," *Learning and Instruction*, vol. 19, no. 5, 2009.
[27] R. Parekh and V. Honavar, "An incremental interactive algorithm for regular grammar inference," in *Grammatical Interference: Learning Syntax from Sentences*, ser. LNCS, L. Miclet and C. de la Higuera, Eds. Springer Berlin / Heidelberg, 1996, vol. 1147.
[28] S. Gulwani, W. R. Harris, and R. Singh, "Spreadsheet data manipulation using examples," *CACM*, vol. 55, no. 8, 2012.
[29] G. J. Bex, F. Neven, and S. Vansummeren, "Inferring XML schema definitions from XML data," in *VLDB*, 2007.
[30] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *VMCAI*, 2006.
[31] B. Gruschko, D. S. Kolovos, and R. F. Paige, "Towards synchronizing models with evolving metamodels," in *ECSMR*, 2007.
[32] A. Cicchetti, D. D. Ruscio, R. Eramo, and A. Pierantonio, "Automating co-evolution in Model-Driven Engineering," in *EDOC*, 2008.
[33] K. Bąk, K. Czarnecki, and A. Wąsowski, "Feature and meta-models in Clafer: mixed, specialized, and coupled," in *SLE*, 2010.
[34] M. L. Gick and K. Paterson, "Do contrasting examples facilitate schema acquisition and analogical transfer?" *CJP*, vol. 46, no. 4, 1992.
[35] R. M. Seater, "Core extraction and non-example generation: Debugging and understanding logical models," Master's thesis, MIT, 2004.
[36] R. DeMillo, R. Lipton, and F. Sayward, "Hints on test data selection: Help for the practicing programmer," *Computer*, vol. 11, no. 4, 1978.
[37] Y. K. Malaiya, "Antirandom testing: getting the most out of black-box testing," in *SRE*, 1995.
[38] S. Weißleder, "Test models and coverage criteria for automatic model-based test generation with UML state machines," Ph.D. dissertation, 2010.
[39] R. Parekh and V. Honavar, "Grammar inference, automata induction, and language acquisition," in *Handbook of Natural Language Processing*, 2000.