On Understanding Teaching Modeling in Computer Science as an Ecosystem

Martin Gogolla
University of Bremen, Germany
gogolla@informatik.uni-bremen.de

***Abstract*—This contribution views teaching modeling as an ecosystem. We identify the factors that make up the ecosystem and establish some relationships between the constituing factors. We discuss some of the factors that need further work and improvement.**

I. Introduction

According to the wikipedia an ecosystem is characterized as follows: An ecosystem is a community of living organisms (e.g. animals, plants) in conjunction with nonliving components of the environment (e.g. water, air, soil) interacting as a system. In an ecosystem there are biotic factors as well as abiotic factors. An ecosystem is self supporting, and the components are linked through nutrient cycles and energy flows. It is a network of interactions among organisms, and between organisms and their environment that can be of any size, but usually encompasses limited space.

Our view on teaching modeling is that of an ecosystem as well. Let us first turn to the factors (or dimensions as we also call them) and then to the interplay of the factors. We identify nine factors due to our subjective view and counting. Thus, the influencing factors for teaching modeling are manyfold, and these factors are highly connected.

II. Nine Factors in Teaching Modeling by Example

We identify nine main dimensions and factors and explain them first by naming them and by giving a short, sketchy example.

(1) Teaching *content* which has three subfactors: the taught modeling *technique*; example: use case modeling; the taught model *qualities*; example: abstraction; the used modeling *style*; example: a domain-specific modeling style.
(2) Used teaching *medium*; example: Excel.
(3) The *demonstrations* employed for teaching; example: an industrial case study.
(4) The *actors* involved in teaching; example: a tutor.
(5) Teaching *timing*; example: teaching undergraduates in the third year.
(6) Teaching *form*; example: a student project with 10 ECTS points.
(7) Teaching style; example: a style involving gamification.
(8) Involved related Computer Science *areas*; example: artificial intelligence viewed from a software engineering teaching activity.
(9) Teaching *environment*; example: legal requirements.

III. Detailing Factors in the Teaching Modeling Ecosystem

Let us go through the nine factors one by one and explain them in some more detail.

Teaching *content* is a highly important and structured factor. There are many modeling *techniques* like class diagrams or state charts that may concern model structure or model behavior; the taught techniques may regard model organisation like handling of profiles, and the techniques may concern model relationships, in particular model transformations. Model *qualities* and model *characteristics* are captured by general, admittedly vague notions like classification, abstraction, structuredness, appropriateness, clearness, and understandability (of a model) as well as by terms like verifiability and executability. Modeling *styles* can be caught by contrastive pairs as textual..graphical, universal..domain-specific or informal..formal.

The teaching *medium* may be a blackboard or a sheet of paper. It may be a combination of a modeling language used in a particular tool like UML [1] or OCL [2] in MagicDraw, USE [3], [4], Umple [5], EMFtoCSP [6] or ATL [7]. The teaching medium may also be some non-mainstream modeling tool like Excel, PowerPoint, or yEd, or the medium could be a programming or database language like Java or SQL.

Demonstrations in form of examples or case studies are essential for teaching. The spectrums may be classified by contrastive pairs like tiny..large, vague..detailed, concrete..abstract, positive..negative, academic..industrial. Demonstrations may be classified by their suitability for the taught modeling technique. Demonstration cases can come from different software development stages and refinement levels.

Actors in teaching may take roles as teacher, lecturer, researcher or tutor in a pro-active position or student in the first place in a more reactive position. One may consider project level-specific roles as programmer or designer and also more organisational positions as examination administrators, technicians or curriculum deans.

Timing in teaching classically decides on wether a teaching activity concerns undergraduate, graduate, or phd students, but one could also consider education for professionals in order to improve skills needed in the job [8]; teaching modeling to professionals is currently under-represented in the teaching modeling literatue, like e.g. the MODELS Educators Symposium. The author has longer experience in teaching the university courses 'Basics of Databases' for the 2nd year, 'Database Systems' for the 3rd year, and 'Design of Information Systems' for the 4th and 5th year (in the context of university Computer Science curricula like Diploma, Bachelor, and Master).

As the teaching *form* one can distinguish between lectures, exercises, courses, seminars, homeworks, projects, and various forms of theses in Bachelor, Master, and PhD curricula. Regarding PhD students one may even consider a research paper as a teaching form trying to transport methods for developing ideas and presentations from a teacher to a student.

The teaching *style* may vary from a classical teacher-in-front option to group-work-oriented styles. Contrastive pairs as traditional..gamified or research-oriented..trainee-oriented styles are important. A research-oriented style views teaching in the first place as representing research results. A trainee-oriented style looks at teaching from the perspective of the trainee and tries to satisfy the trainee's needs respecting and taking into account the trainee's abilities in the first place.

Teaching modeling does not only involve the software engineering realm, but many other Computer Science *areas* teach modeling in some form, e.g. models are used in teaching programming languages, information systems, networks, theoretical computer science, or in artificial intelligence.

Teaching always takes place in a certain *environment* in which a curriculum performing institution is located. There are always related curricula and related institutions at the teaching institution and in the geographical neighborhood. The present teaching colleagues and laws expressing curricula frame requirements also take serious influence.

IV. Relationships between Teaching Modeling Factors

Figure 1 graphically places the discussed factors and dimensions on a circle that allows to establish relationship connections in the inner part of the circle being able to link a single relationship with many factors.

The figure shows prototypically two relationship connections. There are much more similar relationships than the shown ones. It is open future work to explore what the most important relationships are.

The two shown relationships are Teaching-Unit-Core and Teaching-Unit-Neighborhood. Teaching-Unit-Core connects content, medium, demonstrations, and form. It establishes essentials of what makes up a traditional teaching unit. Teaching-Unit-Neighborhood connects actors, areas, and environment. It brings together the neighborhood of a teaching unit, in particular it puts emphasis on the fact that teaching actors are connected to related Computer Science areas.

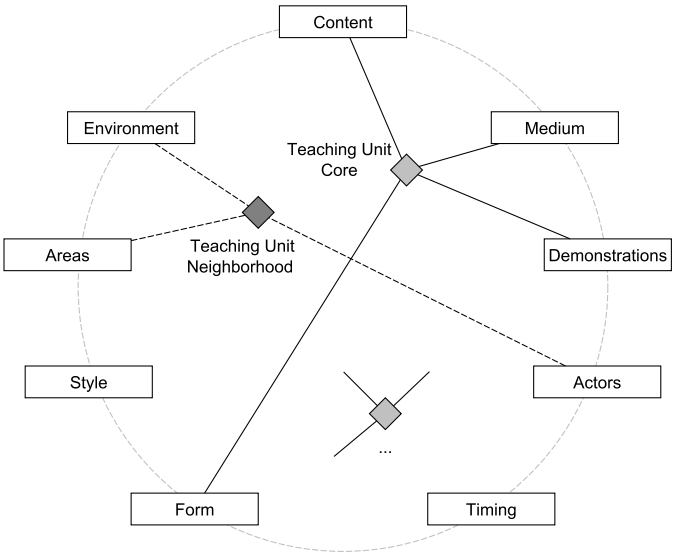

Fig. 1. Factors and dimensions of the teaching modeling ecosystem.

A possible 'know-how flow' relationship (similar to the energy flow in a natural, biotic ecosystem) that could be added to the figure is a reflexive relationship on actors because a student can become a phd student who can become a researcher who can become a lecturer teaching again to a student: a 'know-how flow' in the teaching modeling ecosystem.

V. Factors Needing Attention and Improvement

We here postulate a working thesis that there are currently six factors that need special attention and improvement: Content (in particular model qualities), demonstrations, actors, timing, areas, and environment.

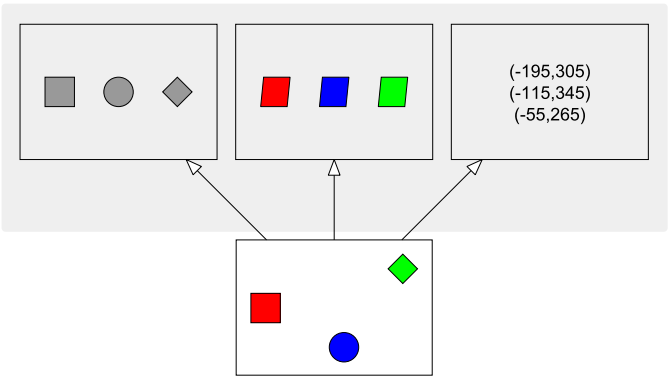

Fig. 2. Attempting to exemplify the process for finding abstractions.

The first factor needing improvement is 'Content'. Content is about 'Model qualities and characteristics' concentrating on notions like abstraction, classification, executability, verifiability, and appropriateness. We think that modeling currently underestimates the role of 'traditional' modeling languages like flowcharts, automata, ER modeling, first order predicate

calculus, or graphs. The modeling community puts the emphasis on 'modern' modeling languages (e.g. UML, OCL, SysML, EMF, ATL, QVT). And the focus is on teaching techniques, not on teaching model qualities like abstraction or classification. It is much easier to teach techniques than to teach qualities. A qualitiy is something that you cannot take into your hands. Figure 2 is an attempt to explain what teaching 'abstraction' basically looks like. The lower part shows the system that needs to be handled and to be abstracted from. The upper part shows different ways how this abstraction can be done: by shape, by color, by coordinate. However in the moment a teacher presents these abstracting solution options, the teacher has done already the creative part of the abstraction process, and the students could not look into the teachers head to see how this process of finding the abstractions was done. Finding abstractions is hard to teach! [9] Another point for improvement concerns the teaching characteristics: models in their representing medium should give feedback to students with respect to verifiable or measurable qualities, in particular in terms of executing and analysing models. Modeling tools should give more feedback in terms of executing typical modeling scenarios or analyzing essential model properties like consistency.

The second factor needing improvement is 'Demonstrations'. Good demonstrations for abstract classes (in the technical sense) covering structure (attributes and operations) as well as behavior (e.g. operation contracts and protocol state machines) could realize the demand for teaching the principles of abstraction. Furthermore, demonstration cases at different development stages and refinement levels should not serve only for presentation of perfect, non-developing models. The discusssion of development steps and their rationals by presenting models at various levels of detail could benefit teaching modeling. Also negative examples that show how students should not model have to be taken into account [10].

The third factor needing improvement is 'Actors'. There is the typical teaching researcher that presents newest results and tools. But for teaching models we would like to see more professional teachers that present material from (neutral, not self-authored) professional textbooks in a professional way. Teaching is typically done from our research perspective through teaching our own research results or at least by considering material that leads to our results because as researchers we are interested in advancing our research results. This research-oriented teaching method should be complemented by trainee-oriented teaching methods that help students in a way they understand and that are inline with their abilities.

The fourth factor needing improvement or at least discussion is 'Timing'. To bring up a very simple question: could modeling be taught before programming is taught? Modeling comes in curricula quite late, and thus students often use modeling techniques as a way of 'documenting the code'. So what would happen if we ban programming from the first year of Computer Science education and teach modeling instead?

The fifth factor needing improvement is 'Areas'. As mentioned above, modeling is done in many, if not all areas of Computer Science, and not only in software engineering. The identification of confederate Computer Science areas who have interest in modeling and their integration and establishing a connection to our field is worthwhile. The modeling community looks from the outside sometimes like to re-invent the wheel. Identification of partners and confederates and relating 'their' modeling to our view on modeling would be a fruitful task.

The sixth factor needing improvement is 'Environment'. As part of the environment one can regard the MODELS conference and its internal structuring. The MODELS Educators Symposium and the MODELS Tutorials are separated, non-communicating events at MODELS. There is no common strategy or plan for these events. Tutorials could be a place for MODEL education and teaching, with concepts being developed hand in hand from both MODELS events. For example, 'bridge' tutorials lying thematically between other Computer Science areas and core modeling techniques could establish connections. One could also discuss to systematically offer basic modeling tutorials, not only specialized tutorials on newest research directions and tools.

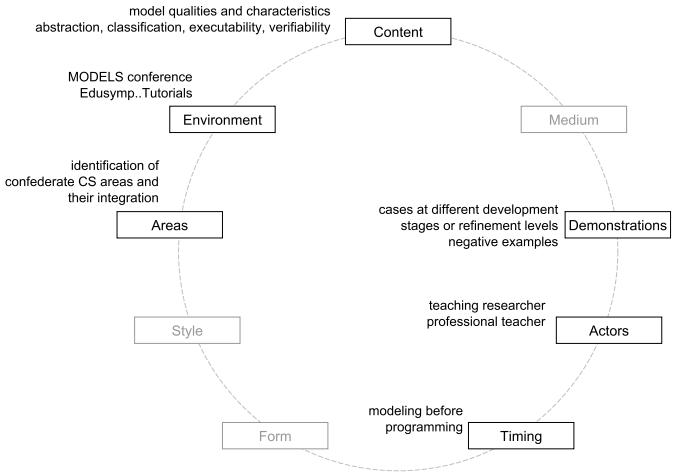

Fig. 3. Teaching factors to be improved.

Figure 3 displays the six factors that in our view need improvement and surveys the items to be worked on.

VI. Conclusion

This contribution has identified the factors that make up a 'teaching modeling' ecosystem. It has opened the option to establish relationships between the constituing factors. Much more workon identifying the relationships is needed. Lastly we have discussed some of the factors that need further work and improvement.

References

[1] J. E. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manuel - Covers UML 2.0, Second Edition*. Addison-Wesley, 2005.

[2] J. Warmer and A. Kleppe, *The Object Constraint Language: Getting Your Models Ready for MDA*. Reading/MA: Addison-Wesley, 2003.

[3] M. Gogolla, F. Büttner, and M. Richters, "USE: A UML-Based Specification Environment for Validating UML and OCL," *Science of Computer Programming*, vol. 69, pp. 27–34, 2007.

[4] M. Gogolla and F. Hilken, "Model Validation and Verification Options in a Contemporary UML and OCL Analysis Tool," in *Proc. Modellierung (MODELLIERUNG'2016)*, A. Oberweis and R. Reussner, Eds. GI, LNI 254, 2016, pp. 203–218.

[5] T. C. Lethbridge, V. Abdelzad, M. H. Orabi, A. H. Orabi, and O. Adesina, "Merging Modeling and Programming Using Umple," in *Leveraging Applications of Formal Methods, Verification and Validation: Proc. 7th Int. Symposium ISoLA 2016*, ser. LNCS 9953, T. Margaria and B. Steffen, Eds., 2016, pp. 187–197.

[6] C. A. González, F. Büttner, R. Clarisó, and J. Cabot, "EMFtoCSP: A Tool for the Lightweight Verification of EMF Models," in *Proc. 1st Int. Workshop Formal Methods in Software Engineering*, 2012, pp. 44–50.

[7] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, "ATL: A model transformation tool," *Sci. Comput. Program.*, vol. 72, no. 1-2, pp. 31–39, 2008.

[8] D. Ratiu, V. Pech, and K. Dummann, "Experiences with Teaching MPS in Industry - Towards Bringing Domain Specific Languages Closer to Practice," in *Proc. 20th Int. Conf. Model Driven Engineering Languages and Systems (MoDELS'2017)*. IEEE, 2017.

[9] M. Simonot, M. Homps, and P. Bonnot, "Teaching Abstraction in Mathematics and Computer Science - A Computer-supported Approach with Alloy," in *Proc. 4th Int. Conference Computer Supported Education*, M. Helfert, M. J. Martins, and J. Cordeiro, Eds. SciTePress, 2012, pp. 239–245.

[10] R. F. Paige, F. A. C. Polack, D. S. Kolovos, L. M. Rose, N. D. Matragkas, and J. R. Williams, "Bad Modelling Teaching Practices," in *Proc. ACM/IEEE MODELS 2017 Educators Symposium*, ser. CEUR Workshop Proceedings, B. Demuth and D. R. Stikkolorum, Eds., vol. 1346. CEUR-WS.org, 2014, pp. 1–12.