bee-up – A teaching tool for fundamental conceptual modelling

Patrik Burzynski[1] and Dimitris Karagiannis[1]

Abstract: This work presents the *bee-up* tool, which has been developed with teaching about conceptual models and modelling in mind and used as such for several semesters. It is an implementation which integrates several modelling languages that gained wide popularity and further extends them in one overarching meta-model. Among others the included modelling languages are the Business Process Model and Notation (***b***PMN), Event-driven Process Chains (***e***PC), Entity-Relationship models (***e***R), the Unified Modeling Language (***u***ML) and ***p***etri Nets, giving *bee-up* its name. Besides allowing the creation of models, bee-up also provides functionality for gaining the value of models and means to extend the available functionality by the user.

Keywords: modelling language integration, use of models, model value, bee-up

1 Motivation

Gaining knowledge and skills about conceptual models and their creation is an essential part of computer and information science at university level [JL17]. We teach our students these skills and knowledge in the Conceptual Modelling course on undergraduate level, where they learn how to "use abstraction to reduce complexity in a domain for a specific purpose". Depending on the purpose different modelling languages may be needed. For example when realising a complex enterprise information system the relevant processes may be described with BPMN, the application designed with UML and the database structure specified using ER and simulate complex behaviour with Petri Nets. Therefore the Conceptual Modelling course covers the foundations of conceptual modelling, teaching the factual knowledge of conceptual modelling by emphasising its procedural and semantic aspects, and communicating the theory of several fundamental conceptual modelling languages (BPMN, ER, EPC, UML and Petri Nets).

However, the aim is not only to teach how to create syntactically correct models in a particular language, but also to show how models can be used to gain value. Therefore, once the students become accustomed to creating conceptual models, their practical application is taught and their value is not only shown in its traditional role of supporting communication, but also as machine-readable artefacts on which various functionalities can be used. Additional details about the teaching can be found in [St19].

[1] University of Vienna, Faculty of Computer Science, Research Group Knowledge Engineering
{ patrik.burzynski | dk }@dke.univie.ac.at

To cover all this we have created the free to use modelling environment bee-up[2] which allows for the creation of models in the above mentioned languages, but also provides some additional languages (Working environment, DMN, Flowcharts, …), integration features allowing the semantic extension and linking of different models in multi-perspective approaches and enables the use of models to derive value (see [Ka16]). For example an ER model can, on one hand, be linked as a detailed view on a BPMN data object and also can generate the corresponding SQL CREATE statements for actually creating that data schema. This demonstrates to students that the modeller has a direct influence on the model value and that both knowledge about the used modelling method, e.g. ER and derivation of SQL statements, and knowledge about the domain that is modelled, e.g. accounting, car production or aviation, are necessary to achieve the desired purpose. Another benefit of bee-up in this context is that it covers the necessary languages for the course in one environment, allowing to focus on Conceptual Modelling tasks and content.

2 Extracting model value

One of the main benefits the bee-up modelling environment provides to educators is the support for shifting perception - from seeing models as a form of graphical documentation ("drawings") to seeing them as machine-processable knowledge artefacts - an educational design problem highlighted in [Bu19]. Following are some examples of features supporting this.

- Meta²-model specific – functionalities relying on a specific meta²-model structure.
  - Generation of RDF structure – An RDF representation of the models can be generated. This allows the integration of model content with ontologies and Linked Data and for processing with semantic technology.
  - Voice-control – Model content can be edited with voice commands based on existing speech recognition services. The functionality is currently in an early prototype stage.
- Abstraction specific – functionalities relying on the semantics on a certain level of abstraction that covers multiple types of models.
  - Simulation of processes – Models depicting processes using different languages (BPMN, EPC, and UML Activity diagrams) can be uniformly simulated, allowing to perform a path analysis, listing all paths and their quantitative details like execution times, or a workload analysis in conjunction with a working environment model, showing the estimated workload on individual people.

[2] https://www.omilab.org/bee-up

- Language specific – functionalities relying on the semantics of a specific language or its adaptation.
  - Derivation of code – It is possible to derive SQL statements for database creation out of an ER model, if additional details like the datatype of attributes are provided. Additionally a simple prototype which can transform parts of a UML class diagram into code (C++, Java or Python) is used in the lectures.
  - Execution of models – Going one step beyond simulation, elements in Petri Nets and Flowcharts models have execution attributes which are used with a simple integrated execution engine to run the defined code or call web services from model elements directly in bee-up.

One specific teaching case employing such features is the querying of BPMN process models using SPARQL, based on the RDF generation. Examples of model queries are provided by analogy to the more familiar data queries, e.g.: 1) determining all the decisions of the process based on the gateways diverging the flow with special conditions or 2) using the process in conjunction with a working environment model to find all people needing access to a specific document.

Another more elaborate scenario uses models to control a robotic arm to perform, with some imagination, a "coffee brewing" process driven by the knowledge captured in the form of a Petri Net or Flowchart and relying on model-robot interoperability. The robotic arm[3] is part of the OMiLAB[4] infrastructure at the University of Vienna and provides a micro-service interface to execute simple operations, like moving to or grabbing on, which are abstracted in the elements of process models. For example a Flowchart model is used to combine these simple operations into more case specific ones, like picking up, placing or dropping objects and considering safety distances, while another Flowchart utilizes these to place the necessary components in the cup. This demonstrates the benefit of models in reducing the complexity of the execution layer (i.e. the robotic arm moving) to the scenario layer (i.e. create coffee using a robotic arm) through several levels of abstraction and that the models can be directly used to execute and validate the realised case. A similar case with the same robotic arm is also modelled as a Petri Net, allowing to execute the relevant operations step by step. Additional details and instruction on how to perform this scenario can be found at the bee-up homepage[2] under "Scenarios".

3 Implementation

The entire bee-up environment is implemented through different artefacts and resources. At its heart is the modelling tool, which is realised on the ADOxx platform[5] due to its free availability and ease of use. Thus bee-up uses the ADOxx platform's meta²-model to

[3] https://austria.omilab.org/psm/content/omiarm1/info
[4] https://www.omilab.org
[5] https://www.adoxx.org

define the available concepts and inherits the architecture of the ADOxx platform for deployment as a standalone tool, parts of which are shown in Fig. 1.

Fig. 1: bee-up architecture as an ADOxx based tool

The specific types of models are realised through extending the ADOxx platform's core meta-model with the necessary classes, relations and attributes, leading to an integration of the languages through shared super-classes. The current bee-up implementation has around 200 classes and around 50 relations in total and an even larger number of attributes. By having one combined meta-model it is possible integrate the modelling languages. An example with different types of models and how they can be connected in the tool using cross-references like "Used system", "Referenced subprocess", "Automation details" and others, as is seen in Fig. 2.

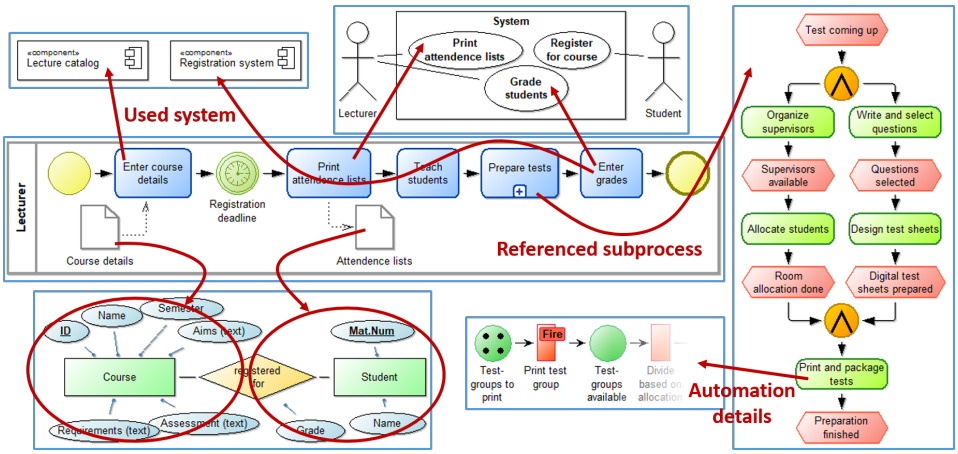

Fig. 2: Example where different types of models are connected

It is also possible for the user to extend the functionality directly available in the bee-up tool. These can be realised in different ways, just like the already available functionalities. Some, like the derivation of code, are implemented solely in the platform-specific

scripting language AdoScript, for which additional support is available[6]. Other functionalities use dynamically linked libraries or use Java applications, which is the case for the generation of RDF. There are also functionalities available that use external web services[7].

4 Conclusion

From our experience in teaching, parts of which are also covered in [St19], it is important to focus on purpose-driven model value to impart the understanding that modelling is not just for the sake of modelling, but to create value and achieve a specific purpose. A single tool that covers all the needs and is freely available further facilitates participation and learning. The bee-up tool also allows to directly witness the result of the modelling work by employing the available functionalities. Additionally, materials are provided to allow anyone interested for self-study in the form of a case study (see [Ka17]).

This experience is based on using the tool ourselves and on feedback provided from multiple sources. bee-up has been downloaded over 6000 since 2016, including different educational and research institutions, and since then used by students in the Conceptual Modelling course among others.

5 References

[Bu19] Buchmann, R.A.; Ghiran, A.; Döller, V.; Karagiannis, D.: Conceptual Modeling Education as a "Design Problem". Complex Systems Informatics and Modeling Quarterly, no. 21, pp. 21–33, 2019.

[JL17] Jung, R.; Lehrer, C.: Guidelines for Education in Business and Information Systems Engineering at Tertiary Institutions. In Business & Information Systems Engineering, no. 59, pp. 189-203, 2017.

[Ka16] Karagiannis, D.; Buchmann, R.; Burzynski, P.; Reimer, U.; Walch, M.: Fundamental Conceptual Modeling Languages in OMiLAB. In (Karagiannis, D.; Mayr, H.C.; Mylopoulos, J.) Domain-Specific Conceptual Modeling. pp. 3-30, 2016.

[Ka17] Karagiannis, D.; Burzynski, P.; Miron, E.T.: The IMKER case study. http://vienna.omilab.org/repo/files/Bee-Up/The_IMKER_Case_Study.pdf accessed Jan. 2020.

[St19] Strecker, S.; Baumöl, U.; Karagiannis, D.; Koschmider, A.; Snoeck, M.; Zarnekow, R.: Five Inspiring Course (Re-)Designs. In Business & Information Systems Engineering, no. 61, pp. 241-252, 2019.

[6] For example the official documentation on https://www.adoxx.org or a code-highlighting extension for Visual Studio Code (available at https://marketplace.visualstudio.com/items?itemName=ADOxxorg.adoxx-adoscript accessed Jan. 2020).

[7] For example using https://www.adoxx.org/live/adoxxweb-verification-details