Supporting Collaborative Decision Making in Software Engineering

Peter Forbrig
University of Rostock
Rostock, Germany
peter.forbrig@uni-rostock.de

Anke Dittmar
University of Rostock
Rostock, Germany
anke.dittmar@uni-rostock.de

ABSTRACT

Smart factories or Industry 4.0 are names of domains of assistive systems. Such systems become more and more important and ask for new technologies in software engineering. They provide support for decision making of users. Human-centered software engineering and subject-oriented modeling seem to be promising approaches. However, decisions have also to be made during software development. The awareness of modelling and discussing alternative solutions have to be teached and tool support has to be developed. The paper discusses aspects of using heterogeneous modeling for specifying applications and collaborative activities.It is asked for education in different paradigms, Domains-specific textual specification languages can be used for this purpose. Additionally, work practices in collaborative design of software are analyzed and corresponding tool support is presented. Task migratability is discussed and characterized as success factor for assistive software systems of the future.

CCS CONCEPTS

• **Software and its engineering** → **Software notations and tools**; • **Software notations and tools** → *General programming language*; • **General programming languages** → Context specific languages; • **Context specific languages** → Domain specific languages;

KEYWORDS

Heterogeneous Models, Domain-Specific Languages, Composition of Languages, Task migratability, Industry 4.0, Business-Process Modeling

ACM Reference Format:
Peter Forbrig and Anke Dittmar. 2018. Supporting Collaborative Decision Making in Software Engineering. In *Proceedings of The 2018 Workshop on PhD Software Engineering Education: Challenges, Trends, and Programs (SWEPHD2018).* ACM, New York, NY, USA, 6 pages.

SWEPHD2018, September 17th, 2018, St. Petersburg, Russia

1 INTRODUCTION

The goal of software applications is supporting users in performing their tasks. New technologies allow interactive systems automatically to adapt to a changing environment. Often such changes are based on conclusions from rules that are triggered by sensed data. The rules specify the solution space of the corresponding applications. These technologies are characterized as smart. Smart meeting rooms, smart houses, smart factories, and even smart cities have been developed. Automatic decision support is provided or users get support for their decisions. We will call this run-time decision support. This is in contrast to design-time support that assists software developers in their work. Supportive applications for this domain range from programming tools, programming environments, computer-aided software engineering tools to integrated environments. However, those tools rarely support adaptation and they often do not allow the elaboration and discussion of alternative solutions. Nevertheless,a lot of decisions have to be made by software developers. This starts with the decision about the importance of requirements, is followed by the decision about the applied software architecture, the decisions during design, the decisions during implementation, etc. The challenge of software engineering and software engineering education lies in strengthening the support of methods for supporting the discussion of alternative solutions and providing computer-supported assistance for that. In this paper, the problem space of decision making is used to discuss different specification methods. Domain-specific textual languages are used to show the application of heterogeneous modeling. This is reached by a Meta mode unifying the concepts of different languages. In this way different paradigms can be used together in one specification and can have references to each other. Additionally, based on task migratabilty the role decision migratability is discussed.

2 CHALLENGES IN DECISION MAKING

Recently, there are a lot of discussions about Ambient Assisted Living (AAL). Related systems are designed to help people (e.g., elderly, children, handicapped, etc.) in having an independent and monitored life with the use and assistance of technology. Additionally, concepts of smart meeting rooms, smart homes, smart factories, and even smart cities exist. Like most software systems, such systems are designed to support users in performing their tasks and their decision making (e.g. what to do next). Assistive software focuses on support for end users in different domains. This can be called decision support during run-time and will be discussed within the next section of the paper. However, software development should be supported by software tools as well. Computer-aided software engineering (CASE) tools have been used for several decades already. Support is provided during design of software. Therefore, after focusing on decision support during runtime decision support for software developers during design time will be discussed.

Decision Making At Runtime

The problem of decision making during runtime will be discussed from three different perpectives. The problem can be tackled in a data-centric or human-centered way. Additionally, it can be specified with one framework or with heterogeneous model. The paper will focus on the second aspect.

Data-Centric Software Engineering. Data are an important resource of the digital world. Data come from different sources, have to be computed as Big Data and change the behavior in large extend software applications. The management of smart applications is a success factor of industry and the whole society. Software Engineering provides processes, models, tools, and principles for constructing and managing high quality software with limited costs. Additionally, software engineering methods should provide explanations to users about the results of deep learning algorithms and big data analysis. This is especially important for complex applications in context of autonomous driving, adaptive systems, and applications for industry 4.0 [18]. Industry 4.0 refers to a current trend of automation and data exchange in manufacturing technologies. The number stands for the fourth industrial revolution and includes applications for cyber-physical systems. (The third revolution characterized by computers and automation, the second by mass production, assembling lines and electricity, and the first one by mechanization, water power and steam power.)

Decisions are supported by algorithms. However, a user should be able to understand the application of rules the decisions are based on. It should also be possible to influence the results of decisions by users.

Rank	Principle	Frequency of use
1	Recoverability	96
2	Familarity	57
3	Consistency	57
4	Substitutivity	54
5	Task Migrateability	40
6	Synthesisability	34
7	Predictability	32
8	Perceptual Ergonomics	31

Figure 1: Weighted HCI rules according to frequency of use (taken from [11]).

The usability criteria of task migratability becomes more important. It is a usability design principle that describes how control for task execution is transferred between system and user. It describes the ability of an interactive application to pass control for the execution of a task so that it becomes either internalized by the user or the application or shared between them [6]. Hinze-Hoare [11] analyzed the literature according to what she called HCI principles. She looked at the most important authors and provided a ranking of the HCI principles that we would like to call usability criteria. The analysis procedure is described as follows: “The number of times that a particular HCI principle was proposed by a significant author multiplied by a weighting factor derived from the author citation frequency allowed a ranking of HCI principles to be determined.” Figure 5 provides the corresponding result.

Task migratability got rank five. However, most applications for assistive systems do still not support task migratability to a large extend. Decision making in software systems is a specific task. It has to be migratable as well. Additionally, decisions of the systems should be supported by explanations on demand that can be understood by users. The data science technologies play also the most important role in technologies for smart cities, which is motivated by sustainable development requirements of global environment and modern cities [17].

Human-Centered Software Engineering. The idea of human-centered software engineering was presented the first time in a large extend in the year 2005 [16]. Some related ideas were already discussed during an INTERACT workshop in Tokyo in 2001 [15] . The title of the workshop was “Software Engineering and Usability Engineering Cross-Pollination”. It was analyzed that classical software engineering did not look at UI design, task-base design, and usability evaluation aspects. In other words, software engineering did not consider human-computer interaction aspects for software development. First papers discussed the integration of development life cycle activities of software and usability engineering. The goal was a common development-process model. This

aspect is still discussed for agile development methods [1]. Nowadays, tasks are not performed by a single users but in teams in a collaborative way. The same is true for decision processes. Therefore, it is very important to understand the collaborative processes and to model them. This aspect is discussed in more detail in the following paragraph.

Modelling Collaboration with Heterogeneous Models. It was already mentioned that collaboration has to be supported by assistive software systems. Before the cooperative aspect will be demonstrated by an example, we will focus on the specification of activities of two roles that we call customer and salesman. The approach can be characterized as subject-oriented [8] and human-centered [16].

A customer asks in our simplified example for information about possible products that can be delivered. From the provided list, a product is selected and the delivery of the correspond-ing price is expected. The procedure of this two tasks can iteratively repeated. A salesman provides a list of products that are available. For a specific product, a price can be delivered. Both tasks can also be repeated several times successively. Specification 1 provides the corresponding task models in the notation of the language DSL CoTaL [2]. It is a domain-specific textual language and allows the specification of task trees in a rule-based way. Each refinement in the task tree is represented by one rule. The tasks of a customer and a salesman are both specified as trees with three levels. The first level starts with the root followed by an iteration on the next level. The iterative task is split into two subtasks. The end of the first task enables the start of the second task.

SPECIFICATION 1: Behavioral models for customer and salesman

```
role Customer {
    root buy = negotiate{*};
        task negotiate =
                ask_for_information >>
                select_product
}
```

```
role Salesman {
    root sell = provide_information{*};
        task provide_information =
                provide_list_of_products >>
                provide_price;
}
```

A customer first asks for information and later (temporal operator enabling - ») selects a product. This can be done iteratively (temporal operator iteration - *). A salesman provides a list of products and later a price for a specific product. The cooperation of both roles is specified by a different model. It is called team model in the context of CoTaL [2]. The notation looks very similar to the role models. However, other language constructs are available.Specification 2 provides a corresponding example.

Several communications can be performed (* after communicate). A communication is started by a customer asking for information. A salesman will afterwards provide a list of

SPECIFICATION 2: Cooperation model for Customer and Salesman example.

```
team coop {
    root trade = communicate{*};
        task communicate =
                Customer.ask_for_information >>
                Salesman.provide_list_of_products >>
                exchange_info_about_product;
            task exchange_info_about_product =
                    Customer.select_product >>
                    Salesman.provide_price;
}
```

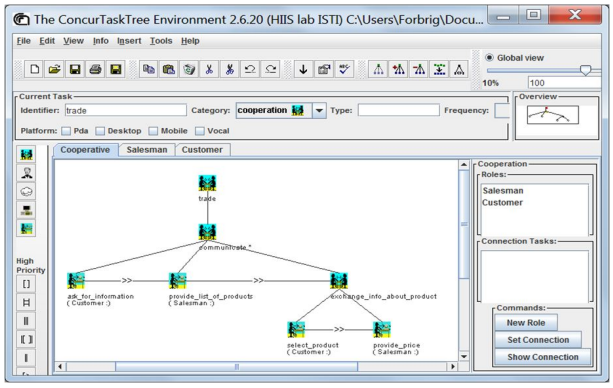

Figure 2: Cooperation model for Customer and Salesman example.

products. The task for exchanging information is followed. It has two sub-tasks. The first one is performed by a customer. A product is selected. Afterwards, a salesman will provide afterwards a price for this product. Temporal relations between tasks of different role models can be provided in this way in a team model. It allows the separation of concerns. Role models describe all related tasks while the team model specifies the collaboration aspect. The editor for CoTaL [2] was implemented with the tool Xtext [20]. It is intended for language engineering and provides the basis for code generation to other tools. For DSL-CoTaL code generation to CoTaSE [3], HAMSTERS [10] and CTTE [4] were implemented. The visualization of the team model in the tool CTTE is presented in Fig. 2.

The hierarchy of the task models is visualized as a tree in CTTE. Each model can be viewed via the corresponding tab and simulation is provided for evaluation. It might be the case that some developers are more familiar with statecharts than with task models. Therefore, it can be useful to provide both views or allow the developer to specify the view he/she is most familiar with. Specification 3 demonstrates the specification of the behavior of a salesman in a DSL for task models and a DSL for statecharts.

Previous examples focused on the task flow only. However, in business processes there are also objects involved. They

SPECIFICATION 3: Task model and statechart model as alternative behavior specifications of a salesman.

```
role Salesman {
    root sell = provide_information{*};
        task provide_information =
                provide_list_of_products >>
                provide_price;
}
```

```
activity_state request_information
  provide_list_of_products => request_price
end
activity_state request_price
  provide_price => request_information
end
```

can be specified in conjunctions with the tasks and used in preconditions or in object flows. Two different languages (object specification and task specification) were embedded in one general language. Relations between objects are omitted because of simplicity in example of specification 4.

SPECIFICATION 4: Cross reference from a task model to an object model

```
object product {
    attribute: String name, String price
}
role Salesman {
    root sell = provide_information{*};
        task provide_information =
                provide_list_of_products >>
                provide_price;
                pre provide_price -> product.price != ""
}
```

Heterogeneous modeling has been used in software development for several decades. However, it seems to become more attractive with the new tools for language engineering. The workshop at EICS 2018 with the title "Workshop on Heterogeneous Models and Modeling Approaches for Engineering of Interactive Systems" supports this impression. Heterogeneous modeling allows the separation of concerns and in this way the management of complexity. Domain-specific languages allow the embedding of different languages by combining their grammars. This was possible to demonstrate with the small provided examples. Different modeling languages were used for specifying:

- Tasks of certain roles (role model)
- Communication between different models (team model)
- Alternative specifications for the same purpose (task model versus state model)
- Combining different views (task model and object model)

Kramer et al. [13] and Lee [14] support the idea of heterogeneous models. It seems to be appropriate to support the specification of different aspects of a system by different views. Domain-specific textual languages seem to be a perfect support for combining different kinds of models.

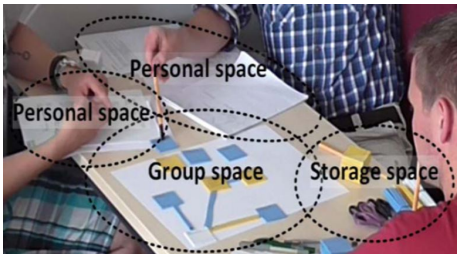

Figure 3: Cooperation model for Customer and Salesman example.

Decision Making At Design Zime

Unfortunately, the decision process during design and implementation of software is not supported very well by case tools yet.

UML Class Diagrams. UML class diagrams are one of the most used kind of specifications for designing software architectures. This section describes studies [5] that were conducted in supporting collaborative software design sessions. Several groups of software designers were observed while performing certain design tasks. The studies included an initial manual collaborative modeling sessions each with 3 participants (Fig. 3). Tangible material like paper was used to represent classes and associations. Based on this analysis, a software prototype for interactive table tops was developed that supports teams up to 3 designers in modeling class diagrams (Fig. 4). The software considers different spaces on its graphical interface. Based on the analysis, these spaces can be grouped into personal spaces for each designer and one group space for all designers. Different class diagram designs can be reflected to compare alternatives for certain design solutions. Ongoing investigation is made on improving design processes by supporting teams with different strategies. One strategy can be to guide sessions for structuring processes at all. However, tool support for multiple designers of collaborative teams differs from tool support for single designers.

Software intended for support of collaborative design sessions must satisfy needs of single persons and groups as well. Individual designers need their own personal spaces for editing classes, relations, and notes of models. All personal spaces are equipped with toolbars and software keyboards that sup-port making edits. The group space shows designed class diagrams and allows designers to adapt layouts. Classes, relations, and notes of diagrams can be blocked by designers that select them in the group space. This strategy helps to avoid conflicts when editing elements. However, the problem of merging models that come from more than one participating designer arises when targeting the functionality of how to deal with conflicting model elements. The same problem

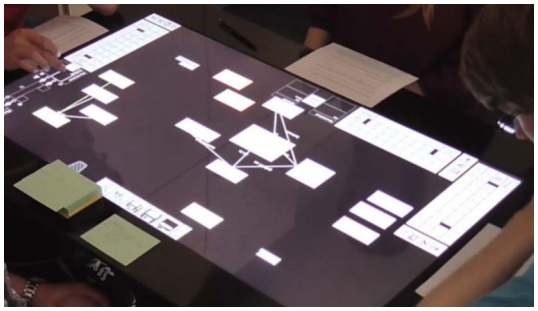

Figure 4: Cooperation model for Customer and Salesman example.

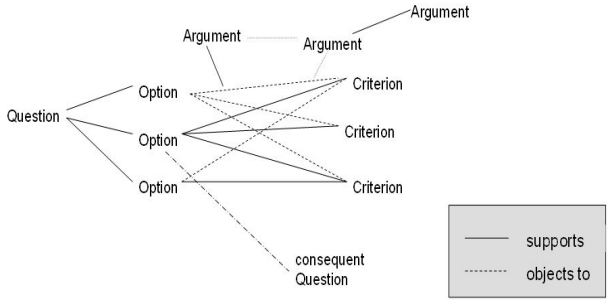

Figure 5: Cooperation model for Customer and Salesman example.

arises when designers try to merge different forks of alternative models designs. Solutions for this problem are parts of ongoing studies.

Business Processes in Context of Industry 4.0. Industry 4.0 is characterized by Wortmann et al [19] as: " the current trend of integrating automation systems with processes and stakeholders of the complete value-added chain as well as part of the high-tech strategy of the German Federal Ministry for Education and Research." Sometimes it is also characterized as smart factory that needs new forms of human-computer interaction, improvements of transferring digital instructions to the physical world, emergence of analytics and business intelligence capabilities, and computational complexity. Industry 4.0 is well characterized in [18]. There exist four design principles. They are called interoperability, information transparency, technical assistance, and decentralized decisions. Interoperability means the ability of machines, devices, sensors, and people to connect and communicate with each other via the Internet of Things (IoT) or the Internet of People (IoP). The ability of information systems to create a virtual copy of the physical world by enriching digital plant models with sensor data is called information transparency. This requires the aggregation of raw sensor data to higher-value context information. Technical assistance is divided into two aspects. First, the ability of assistance systems to support humans by aggregating and visualizing information comprehensively for making informed decisions and solving urgent problems on short notice. Second, the ability of cyber physical systems to physically support humans by conducting a range of tasks that are unpleasant, too exhausting, or unsafe for their human co-workers. The ability of cyber physical systems to make decisions on their own and to perform their tasks as autonomously as possible. Only in the case of exceptions, interferences, or conflicting goals, tasks are delegated to a higher level.

The idea of task migratability is not mentioned in this context. However, it would fit very well. There are also modeling approaches like Kannengiesser and Müller [12] that fit very well to our DSL CoTaL. They present an agent-based approach for smart factories that is subject-oriented. Our presented approach is subject-oriented as well. It has been already applied to smart environment applications [9].

Additionally, supportive systems for designing business processes like that provided by Fellman et al. [7] will be needed in the future. The idea of private modeling spaces and collaboration support can be applied to such systems as well.

3 DISCUSSION

Software engineering is currently very much related to decision support. It is intended to develop software that provides appropriate support for users while making their decisions. Therefore, PhD students have to be able to analyze application domains and to model cooperation activities and decision making. They have to know a portfolio of modeling techniques. They have e.g. to know class diagrams, state-based specifications, process specifications, task models and grammars. It would also be good if they knew the basic principles of logical programming, functional programmings and aspect-oriented programming. Disciplined heterogeneous modeling [14] has to be taught. This kind of modeling was discussed by using textual domain-specific language examples. Combining grammars of different languages provides the opportunity to specify heterogeneous models in one specification with references to each other. A Meta model is generated from the grammar specification that is the basis for the generated editor. PhD students should be able to select the appropriate specification techniques for a specific domain. Designing a domain-specific language is a perfect training for that. It ia not only good for educational purposes but can be applied to real life projects as well. The Xtext framework [20] facilitates the design and the generation of the corresponding editor very well. Relatively few effort is necessary for providing results. Changing keywords in the language (e.g. object to class) can be performed in a minute.

The specification of the code generation to different tools is more complicated. It needs knowledge of the external specification of the models of the tools and some ideas for the correct transformation of the instances of the domain-specific language to the models of the tools.

The unification of the concept of team model in CoTaL and the cooperation model in CTT in the domains-specific language DSL-CoTaL resulted in a quite readable specification. It restricts the expressiveness of the specifications but simplifies them. The discussed approach might be an inspiring example for further abstractions. Students appreciated simple modification options in the textual specification. It was much easier for them to move a sub-tree to another position than in the graphical editors that they did not know so well. Nevertheless, students used the visualization of the task hierarchy in the graphical editors to check their ideas. The textual representation opens a new perspective. Graphical and textual specifications should be used together to inspire each other. Additionally, students mentioned that they liked the rule-based structure of the language. Therefore, they did not have to specify identical sub-trees twice. Generic components were characterized as supportive as well.

It might be worth to look for further abstractions of languages for task models and business processes.

4 SUMMARY

Appropriate support for decision making is currently one of the biggest challenges of software engineering. This has to be reflected in education as well. Students have to be aware of the decsion processes during software development and the need of explaining decisions for end users.

Decision migratability was considered as important aspect and future challenge of smart systems. It was suggested to use the notation of QOC to represent the decision space and the argumentation for a decision.

The process of collaborative decision making has to be further analyzed. There are challenges in group composition because personal profiles might be in conflict to each other. Providing hints by tools in this sense seem to be useful as well. Making software developers sensible for decision making process during development and the fact that most of the time no best solution exists is also a challenge for the future.

REFERENCES

[1] Carmelo Ardito, Maria Teresa Baldassarre, Danilo Caivano, and Rosa Lanzilotti. 2017. Integrating a SCRUM-Based Process with Human Centred Design: An Experience from an Action Research Study. *2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)* (2017), 2–8.

[2] Gregor Buchholz and Peter Forbrig. 2017. Extended Features of Task Models for Specifying Cooperative Activities. *PACMHCI* 1, EICS (2017), 7:1–7:21. https://doi.org/10.1145/3095809

[3] CoTaSE 2018. SE Group. (2018). https://www.cotase.de/

[4] CTTE [n. d.]. HIIS Laboratory. ([n. d.]). http://hiis.isti.cnr.it:4500/research/CTTE/home.

[5] A. Dittmar, G. Buchholz, and M. Kuhn. 2017. Effects of Facilitation on Collaborative Modeling Sessions with a Multi-Touch UML Editor. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, Vol. 00. 97–106. https://doi.org/10.1109/ICSE-SEET.2017.14

[6] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. 2003. *Human-Computer Interaction.* Prentice-Hall.

[7] Michael Fellmann, Novica Zarvić, and Oliver Thomas. 2018. Business Processes Modeling Recommender Systems: User Expectations and Empirical Evidence. (04 2018), 64–79.

[8] Albert Fleischmann, Werner Schmidt, and Christian Stary. 2015. Subject-Oriented Business Process Management. In *Handbook on Business Process Management 2, Strategic Alignment, Governance, People and Culture, 2nd Ed.*, Jan vom Brocke and Michael Rosemann (Eds.). Springer, 601–621. https://doi.org/10.1007/978-3-642-45103-4_25

[9] Peter Forbrig, Anke Dittmar, and Mathias Kühn. 2018. A Textual Domain Specific Language for Task Models: Generating Code for CoTaL, CTTE, and HAMSTERS. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2018, Paris, France, June 19-22, 2018.* ACM, 5:1–5:6. https://doi.org/10.1145/3220134.3225217

[10] HAMSTERS 2018. ICS Group. (2018). https://www.irit.fr/recherches/ICS/softwares/hamsters.

[11] Vita Hinze-Hoare. 2007. The Review and Analysis of Human Computer Interaction (HCI) Principles. *CoRR* abs/0707.3638 (2007). arXiv:0707.3638 http://arxiv.org/abs/0707.3638

[12] Udo Kannengiesser and Harald Müller. 2013. Towards Agent-Based Smart Factories: A Subject-Oriented Modeling Approach. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 03 (WI-IAT '13)*. IEEE Computer Society, Washington, DC, USA, 83–86. https://doi.org/10.1109/WI-IAT.2013.155

[13] Max E. Kramer, Erik Burger, and Michael Langhammer. 2013. View-centric Engineering with Synchronized Heterogeneous Models. In *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling (VAO '13)*. ACM, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/2489861.2489864

[14] Edward A. Lee. 2010. Disciplined Heterogeneous Modeling. In *Model Driven Engineering Languages and Systems*, Dorina C. Petriu, Nicolas Rouquette, and Øystein Haugen (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 273–287.

[15] Ahmed Seffah and Peter Forbrig. 2001. Software and Usability Engineering Cross-Pollination. In *Proc. INTERACT*. IOS Press, 839.

[16] Ahmed Seffah, Jan Gulliksen, and Michel C. Desmarais. 2005. *An Introduction to Human-Centered Software Engineering.* Springer Netherlands, Dordrecht, 3–14. https://doi.org/10.1007/1-4020-4113-6_1

[17] Xiong Zhang Wang Jingyuan, Li Chao and Shan Zhiguang. 2014. Survey of Data-Centric Smart City. *Journal of Computer Research and Development* 51, 2, Article 239 (2014), 20 pages. http://crad.ict.ac.cn/EN/abstract/article_2077.shtml

[18] Wikipedia. 2018. Industry 4.0. (2018). https://en.wikipedia.org/wiki/Industry_4.0

[19] A. Wortmann, B. Combemale, and O. Barais. 2017. A Systematic Mapping Study on Modeling for Industry 4.0. In *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Vol. 00. 281–291. https://doi.org/10.1109/MODELS.2017.14

[20] Xtext 2018. Language Engineering For Everyone! (2018). https://www.eclipse.org/Xtext/.