Collaborative modelling and metamodelling with MetaEdit+

Steven Kelly
MetaCase
Jyväskylä, Finland
stevek@metacase.com

Juha-Pekka Tolvanen
MetaCase
Jyväskylä, Finland
jpt@metacase.com

***Abstract*—MetaEdit+ is a collaborative modelling tool and language workbench with a multi-user client-server object repository. Users, clients, server and repository can all be geographically distributed, and users access them with local apps, remote desktop or browser. Users can work with models and metamodels synchronously, with buffering via design transactions (minutes to hours), to prevent confusion and errors from releasing half-finished work. The tool uses automatic fine-granularity locking to avoid conflicts, allowing high concurrency. In this paper, we present an example case on which neophyte users can try this approach to collaboration in practice with minimal effort.**

***Index Terms*—collaborative modeling, language evolution, modeling tool, multi-user repository**

I. Introduction

The issue of collaboration is central to the scalability of model-driven development [1]–[4]. With domain-specific modelling (DSM) having proved the best form of model-driven development for many cases, the issue of collaboration among the developers of the language also becomes important. Even more vital becomes the issue of how language evolution affects existing models and the ongoing work of modellers. Human and organizational issues will have their effect on these questions, but the key determining factor is often the modelling tool.

Modelling tools have addressed collaboration in very different ways. Some have simply not got that far yet, and leave the issue for users to sort out on a file level or via 'one user at once' policies outside the tool. Others have tried to apply code-based practices such as character-based files under version control, with diff and merge to integrate changes by multiple people to the same file. Industrial tools have tended to offer a repository or database approach, avoiding conflicts and manual merging through locking — at various levels of granularity, and with locking automation varying from manual to fully automatic. Over the years there have also been trials of real-time collaboration with instant visibility of others' work: clearly useful at the early stages of brainstorming and whiteboarding, but often distracting in longer term work.

Few people have practical experience with the full range of these approaches, so a workshop to bring tool builders and modellers together and have them try the different approaches is most welcome.

A. Case

In this paper, we offer a case of collaboration between metamodellers and several modellers of various roles that we have commonly encountered in industrial projects. The earliest days of a project are generally in free-form tools like Word or whiteboards, which we will not cover here. Collaboration on the more formal models common in DSM begins at the next stage, with the most critical times for tooling coming once several models exist. We thus begin our case with an existing metamodel and set of models. The metamodel and models are extended and updated by the various participants in parallel, with the work of most touching all models — the hardest test for collaboration tooling. The domain-specific modelling language we use is based on familiar concepts of sensors, actions and states, targeting an Internet of Things consumer device. The level of the models is such that new users should be able to understand them and work with them quickly, without needing to learn a new domain.

B. Tool

MetaEdit+ [5] is an industry-proven collaborative modelling tool and language workbench with a multi-user client-server object repository. Users, clients, server and repository can all be geographically distributed, and users access them with local apps, remote desktop or browser. Users can work with models and metamodels synchronously, with buffering via design transactions (minutes to hours), to prevent confusion and errors from releasing half-finished work. The tool uses automatic fine-granularity locking to avoid conflict, allowing high concurrency. The automation works transparently and with low ceremony, but with details available when desired. This approach has been found to help users stay focused on their work, yet with full visibility of others' completed edits.

C. Solution

The practical demonstration of the MetaEdit+ approach to collaboration on this case will be carried out by several neophyte users in parallel, with only the briefest of introductions to the tool and little time for installation. We thus choose to offer the users remote access to MetaEdit+ client instances running in the cloud, connected to a MetaEdit+ server there. Users can work either with remote desktop software or in a browser. Our experience is that this approach is the most

solid for new users, particularly when they are dispersed geographically and organizationally. After a network dropout or even PC reboot, the user can simply reconnect and their session is still open, exactly as they left it. It also avoids the need for installation of modelling software on a user's own computer: unlikely to be a problem for these participants, but becoming increasingly important in commercial deployments.

II. Case

A. Description of the problem/application

1) What is the context and purpose of the case?: "The year is 2030. 15 years earlier, Juha-Pekka made a domain-specific modelling language for the Internet of Things, the pinnacle of cool tech back then. The language allowed modelers to create programs for a Thingsee One [6], a device with a number of environmental sensors and the ability to send various kinds of alerts. Using the language, he built some models for a boat he shared on the South-West coast of Finland back then. Now, he is planning for his well-earned retirement — on the tropical Canary Islands (he can dream!). He wants to expand the models and update them for the rather different climate there, and he has enlisted your help."

2) What is modelled?: The modelling support provided includes two languages, with Thingsee Purpose being our focus here, and Thingsee Profile a simple collection of Purpose models to run together on a device.

The Thingsee Purpose modelling language is based on the sensors and services of an Internet of Things device, the Thingsee One. The language offers the modeller various environmental sensors (Accelerator, Timer, Geofence, Location, Speed, Temperature, Humidity, Pressure, Luminance) and actions to interact with the world via services (cloud, mobile, SMS). There are also some sensors to monitor the status of the device itself, such as its battery level and charging status. The sensors and actions specified are connected together, and which are activated at any given point is narrowed down by timers and state-based logic. A transition in one Thingsee Purpose can also shift the application into a state in another Thingsee Purpose. Rules in the language help guide the modeller in creating applications, as well as offering checks to spot models that would require operating the device in unsafe situations (e.g. too hot, too great a G-force).

A code generator produces the JSON specification code that can be uploaded to the Thingsee device and executed by the fixed 'engine' code there.

An example of a simple Thingsee Purpose model to detect a car speeding is shown in Fig. 1. The system starts in state 'Below limit' (with the heavy outline), and if the speed is $>$120 km/h (shown under the speedometer on the right) for 15 seconds, an SMS text message and cloud notification "Kid is speeding" will be sent, and the system will move to state 'Above limit'. When the speed is $<$117 km/h for 10 seconds, the system returns to the 'Below limit' state.

A larger Thingsee Purpose model for a theft alarm and tracker is shown in Fig. 2, in the Diagram Editor of MetaEdit+.

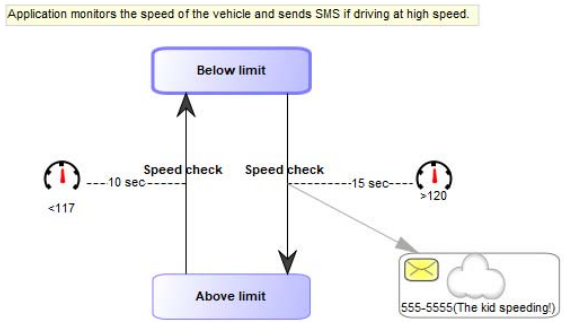

Fig. 1. Thingsee Purpose model to send warning when car is speeding.

3) Focus on the collaborative needs: teams, views, etc.: Modelling by a team can be handled in many different ways. At one extreme, the division of tasks can be primarily to work around a chosen tool's shortcomings: e.g. modularizing the models so that each model file is only worked on by one person. Where that modularization also follows a natural division of knowledge and skills, that may indeed be a good approach. At the other extreme, each person may work across many models, adding their own particular knowledge or skill to each model. This latter approach generally puts the most strain on tool collaboration facilities, and thus we choose it here as our acid test. If the tool can cope with this, it will be fine in situations where modularization and division of labour help keep users' from treading on each others' toes even without tool help.

a) Why is collaboration needed for this case?: Where work on models is divided according to skills, we have seen several common patterns over the years. The most common differentiator is **metamodellers** and non-metamodellers: although the metamodeller may also model, most modellers will not metamodel. Another common case is to have a group of modellers responsible for **initial models**, at the level an end-user could understand, and a second group of modellers who complete those models with more **technical details**. In many companies making products with an end-user interface, the first group would earlier have created their specifications with Word or PowerPoint, whereas the second group were programmers who implemented the specifications. A third case is a group who bring to bear some **specialist** knowledge or skill on individual values in models, without affecting the structure or behaviour: e.g. language specialists who check text for display to users or provide translations. A fourth group that we have not seen, but have often found wanting, is people who are able to **lay out** models in a readable fashion. For some reason, a non-trivial percentage of people who have good conceptual modelling skills are lacking in the visual and communication skills needed here, and their models could be made more useful for others by somebody neatening them up and making them visually clearer.

In this case, we will have an example of each of the bolded categories above, giving us five roles: Metamodeller, Model

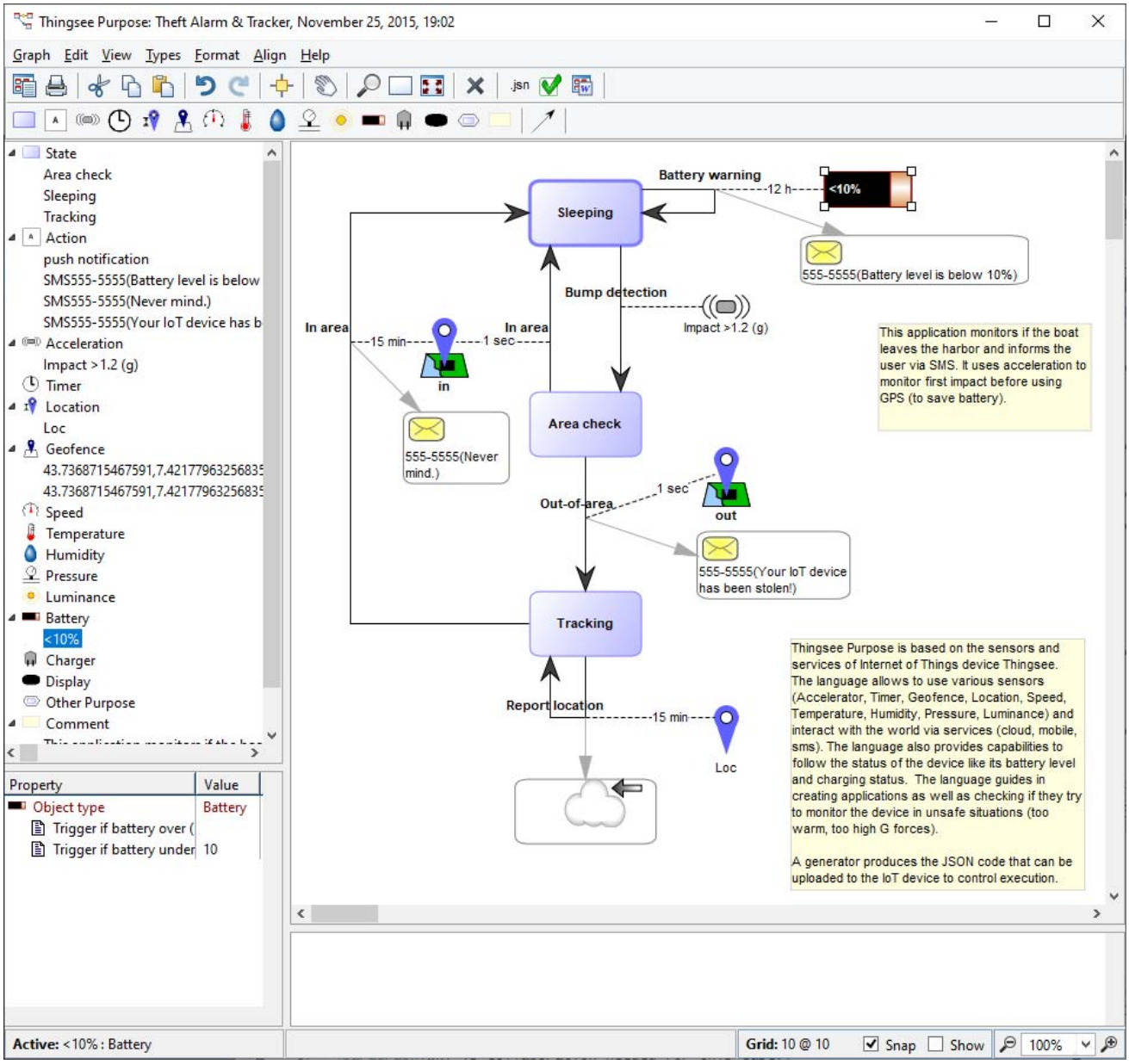

Fig. 2. Diagram Editor with Thingsee Purpose model for a theft alarm and tracker.

Creator, Technical Modeler, Climate Specialist, and Layout Expert. At the start of the case there will be the metamodel and an existing set of models. The Metamodeller will update the metamodel, affecting all models, and the Technical Modeller, Climate Specialist and Layout Expert will all be working simultaneously on all the models. The Model Creator's work will depend on the Metamodeller's work, and the others will also apply their skills to these new models. The Climate Specialist will use their intimate knowledge of weather patterns in South-West Finland and the Canary Islands — or Google! — to update the temperatures, speeds, accelerations caused by stronger seas etc.

b) What variations of the case affect collaboration?: Depending on time and numbers of participants, more than one participant could play the same role, some roles could be omitted, or one participant could play more than one role. Similarly, the task list for each role has elements that could be omitted if necessary.

A limitation of the workshop format is obviously time: a series of tasks that would take an expert user at most five minutes cannot realistically cover normal collaboration scale. With several people working, there would normally be one or two orders of magnitude more models than are feasible here. This spreads users' work out, reducing the number of

operations that could be perceived as conflicting. In the tiny set of models in this case, giving all users free rein to do whatever they want would inevitably lead to conflicts, if not on the tool level then at least on the semantic level. Conversely, very strict instructions to each user could avoid conflicts even in tools with poor concurrency support. It will be interesting to see how users and tool fare with broad instructions, and roles that force users to work on the same models.

III. Methodology/tool

A. General description of tool

MetaEdit+ [5] (metacase.com/products.html) is a language workbench and modelling tool offering strong multi-user support [7], [8] and version control integration with no need for manual diff and merge [9]. It supports multiple simultaneous modelling languages, multiple representations of the same model as matrices [10], tables and text as well as diagrams — which go beyond bitmaps or boxes to offer dynamic graphical languages [11] with real-time synchronous feedback in symbols [12]. It has a particular focus on domain-specific modelling with full code generation [13] and ease of language creation and evolution [14], [15].

1) Related tools' collaboration support: A recent systematic mapping study [16] investigated language workbenches used to create DSM tools and similar textual tools, categorising them into commercial and non-commercial. Of the commercial tool use reported, MetaEdit+ accounted for over half, with Enterprise Architect, Microsoft Visual Studio (and DSL Tools), Obeo Designer and MagicDraw covering most of the rest, as shown in Fig. 3. For reasons of space, we will restrict ourselves to this top 5, accounting for 96% of reported uses.

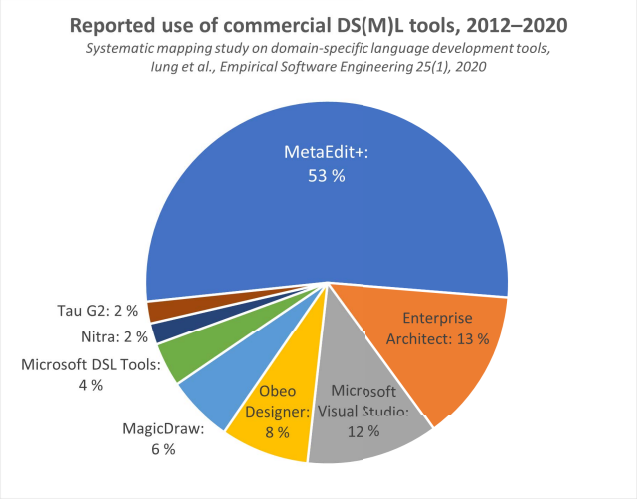

Fig. 3. Reported use of commercial DSM Tools.

The Corporate Edition of Enterprise Architect[1] adds support for storing models in relational databases, using a manual lock-modify-unlock approach to enforce that only a single user edits a given Package at once. Explicit "Get All Latest", "Check-out package", "Check-in package" and "Check-in Branch" commands are offered (the user must remember to use the last, if a change affects more than one Package). While a package is checked out, no other user can check-out that package.

Microsoft Visual Studio does not in itself offer collaboration on models, and nor apparently do the DSL Tools[2]: collaboration, multi-user aspects or merging are mentioned by neither the web site nor any of the papers in the study (all by the same co-authors).

Obeo Designer offers a separate Team edition[3], with Eclipse CDO on a relational database allowing multiple users to work on the same models. Editing elements automatically locks them, and explicitly saving the model propagates the updates to the repository and other users.

MagicDraw offers a Teamwork Server[4], with concurrent support via pessimistic locks on parts of models. The user must manually take the lock on each element or diagram, and manually release the lock after the edit.

Most of the non-commercial tools do not support simultaneous collaboration, only post hoc diff and merge. A partial exception is WebGME [17], which uses lightweight branching where branches share the models that have not been modified. The user is responsible for manual creation of branches, changes can be broadcast to all users and conflicting edits can be detected, retried, or rejected; there is no automatic conflict resolution, or locking to prevent conflicts. In the case of conflicting changes, the models diverge into two disjoint branches, and merging is left as a manual task for the users [18]. Another partial exception is AToMPM [19], which allows multiple users to edit, but offers no locking or conflict resolution: when multiple users change the same element, the first change seen by the server wins. Although this avoids manual locking or merging, it also loses users' work after they have done it.

B. Summary of typical use(s)

1) Real-time vs. offline collaboration: Real-time synchronous collaboration and offline asynchronous collaboration are often seen or at least presented as a black and white, binary decision. In practice, there are shades of grey, and of course no communication is entirely free of latency. Both approaches have their merits, both for the users and for the tool builders. In MetaEdit+, we have aimed at what is best for the users, in the majority of the time using the tool.

The majority of MetaEdit+ industrial customers use it as a multi-user domain-specific modelling tool. The models are thus concise and close to the minimal amount of information needed to specify that system or part of a system, within the common solution space supported by the language and its generators. Changes can thus be fast — little need to trawl

[1]http://www.sparxsystems.com/WhitePapers/Version_Control.pdf, retr. 3.9.2021

[2]https://docs.microsoft.com/en-us/visualstudio/modeling/modeling-sdk-for-visual-studio-domain-specific-languages, retr. 3.9.2021

[3]https://www.obeodesigner.com/en/collaborative-features retr. 3.9.2021

[4]https://docs.nomagic.com/display/MD190/Locking+a+model+for+editing+in+Teamwork+Server retr. 3.9.2021

through other models to check technical details that might impact the desired change — but also contain little routine work (cf. programming, where what is semantically one change may require a number of similar changes elsewhere in the code, only some of which are automatable by refactorings). Modelling in a DSM language thus keeps the modeller focused on the essential aspects of the task, and the time freed from routine work can be used to think on a higher level, from the point of view of the end user or resulting system.

To put it another way, DSM aims to keep the modeller's brain working at maximum capacity on the essential issues as much of the time as possible. As any programmer knows, minimizing distractions is key to productivity when doing serious brain work. Outside of initial brainstorming or whiteboarding, modellers are often best served by not being intimately aware of every action of every other modeller on the team. Yet there is little more frustrating than working for hours or days on the basis of one specification, only to find that the specification has changed in the meantime. Particularly time-consuming is the case where that specification was internally inconsistent, and should never have been released in that state. Modellers thus need relatively fast updates — hours or fractions of an hour, rather than days or weeks — but those updates should only be of work that the other person considers ready for release to the team. In MetaEdit+ this is handled through design transactions, with an explicit commit by the user to save the work and make it available to others. (To discard his changes, a user can alternatively abandon his transaction, in addition to normal multi-level undo operations.)

Where users' work clashes on the same model elements, however, we want to know straight away, to avoid later conflicts. This is managed in MetaEdit+ by locks at the server: when a user starts editing an element, he obtains a lock on it; if a second user attempts to edit the same element, before having seen the first user's changes, their attempt to obtain a lock will fail. The second user will see the lock through normal UI mechanisms such as greying of OK buttons or menu items, and if desired can choose Info… to see details on the reason and the identity of the first user.

In terms of conflict-prevention, MetaEdit+ thus operates as real-time synchronous collaboration. In terms of visibility of changes to others, only changes the user has accepted by committing are made available to others. In terms of visibility of changes by others, the user's work will not be interrupted mid-transaction by others' changes, but only when he begins a new transaction (i.e. MetaEdit+ operates at the highest level of ACID transaction isolation [20], [21]). Between commits, a user is thus assured of a consistent view of the repository, rather than feeling the world is shifting beneath their feet as they work.

2) Granularity of collaboration unit: Modelling tools with no support for collaboration lock implicitly at the level of a model file, often preventing users from even viewing that model (or models) until the first user closes it. Many tools' collaboration support works at the granularity of a single model, preventing two users from editing simultaneously within it. Where objects are reused between models, this granularity can expand to a set of related models. Bad experiences from these coarser levels of granularity are one reason why many are wary if they hear a tool uses locking for concurrency.

In MetaEdit+, locking is extended to the finest levels of granularity: individual properties in objects. This gives maximum concurrency and permeability to modelling, and minimizes the times a user will be unable to perform a desired edit. However, there are situations where allowing only the finest granularity will actually work against the user, and in those instances MetaEdit+ will take a broader lock: here are two examples.

Firstly, in many cases the properties in a single object are semantically interrelated: there may be a Start Date and an End Date, or as in the case of our Thingsee language, a Maximum Temperature and a Minimum Temperature. Allowing one modeller to edit one of these while another edits the other will often lead to illegal or inconsistent models, despite neither user having themselves done anything wrong. Even outside of these clearest cases, some level of interrelation is so common that we have found it best to lock all of the properties of an object together: the benefit to consistency and comprehensibility clearly outweighs the tiny number of cases in practice where users would actually edit different properties in the same object in overlapping transactions. This also means there is no need for the metamodeller to answer the often impossible question of what might be semantically interrelated, and modellers to wonder in which cases they might be protected: the tool will protect them for properties within an object, and for other cases they can use other mechanisms like checking reports or warning symbols.

Secondly, experiences from other modelling tools such as TDE [22] show that if graphical layout operations are allowed simultaneously to multiple users in the same diagram, this often results in a 'tug of war' between the users over objects. Graphical layout is thus locked as a whole for a diagram when opening it. By analogy, operations that affect the set of objects or relationships in a graph also lock the graph as a whole, while leaving the details of the individual objects or relationships themselves free for others to work with. In terms of the GOPPRR meta-metamodel of MetaEdit+: a user can have the lock on the Graph, without having the lock on any of its Objects or their Properties.

C. Any features specifically interesting for this case?

Locking in MetaEdit+ is automatic: the addition of explicit operations to lock and unlock would add extra work and cognitive load for the modeller, and the results of such manual decisions are unlikely to reach the levels of accuracy and concurrency afforded by good automation. However, there is one common case where users will want to avoid locking, to leave the locks free for others, and that is where the users will work their way through many models or elements without changing them. Most commonly this occurs in a review situation, or an ad hoc manual search. In those cases, the user can choose not to obtain a lock, by holding down

Shift while opening the element. (It is also possible to invert this for frequent read-only users, so Shift is needed to obtain a lock. Similarly, users can be granted read-only access to certain projects — sets of models.)

In the tasks in our case, this feature will be used by the Technical Modeller and Climate Specialist when they open graphical diagrams. By holding down Shift when opening each diagram (so roughly five times), they can edit the properties of objects in that diagram, while leaving the graphical layout unlocked for the Layout Expert. This is mostly a courtesy: in practice, the Layout Expert can also open each diagram early in his session, thus obtaining and holding the locks.

IV. Solution

A. How is the domain reflected in the tool?

The domain-specific modelling language for an Internet of Things device, Thingsee One, is implemented in MetaEdit+ as a graphical modelling language.

B. How does the tool support the required collaboration?

The aim in MetaEdit+ is to support many kinds of collaboration needs found in the development phases where a modelling language is used. (Pre-modelling phases such as brainstorming and free-form whiteboarding are not really in focus; nor are post-modelling phases such as code compilation, linking, automated testing, releasing etc.) In the case in this paper, we have deliberately chosen collaboration that involves multiple users editing the same models, traditionally the weak point of both file-based and many repository-based tools. We have also made sure the coverage extends across both metamodelling and modelling, across both conceptual and representational data, and across both graph-level and graph element levels of granularity.

Although the case is thus challenging, the architecture and mechanisms of the MetaEdit+ multi-user version — described in the previous sections — should be able to support it, at least in theory. Since the case is also based on the kinds of collaboration that are found in industry (extended to some extent with things we would like to see more of in industry), we have good evidence that these kinds of collaboration also work in practice. Testing their applicability to neophyte users in a tight timetable will be interesting.

Some features of the collaboration tasks demand a certain order, regardless of the tool. For example, if the Metamodeller defines a new Property slot in an Object type, the Technical Modeller can only start filling in those properties in objects after that definition has been made. Similarly, the Model Creator must create new models before the modellers in other roles can perform their tasks on those models.

C. Link to demonstration video (optional)

As yet, we do not have a video of the five roles in the case performing their work in parallel. For 5 minutes of video showing metamodeling and modelling collaboration in a different language in MetaEdit+, please see 1:57-5:04 and 7:10-9:03 in https://youtu.be/JQzt4cd8ppc.

For a brief introduction to the Thingsee modelling language and using it in the MetaEdit+ tool, interested readers can see https://vimeo.com/139681451. This shows a single Thingsee 'purpose' (an application) being built, to monitor when a car borrowed by one's teenage offspring is exceeding the speed limit.

The rough order and parallelism of the case tasks are shown in Table I. More detailed instructions will be provided to participants.

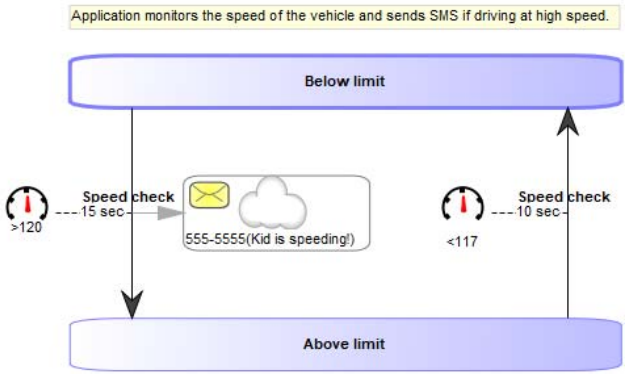

Fig. 4. Updated layout of Speeding model.

V. Hands-on session

A. What insights do you aim to gain from these hands-on experiences?

We are particularly interested in seeing the initial response of modellers more used to file-based tools: will the lower-ceremony approach of a repository be welcomed as simpler and more efficient, be thought difficult simply because it is different, or even be perceived as a threat — as indeed modelling itself was to some programmers. Offering the tool via the cloud, without the need for local installation, may be felt to be easier, or equally to be disempowering or confusing.

In addition to the final models and the workshop survey, we will also collect user feedback during the hands-on (e.g. via Zoom recording), and the automatic version history of the models will show how users worked. Where possible, users could also record their session as video locally, or capture their screen as their virtual web camera for Zoom.

B. Why are these insights relevant?

Seeing new users take their first steps with a tool is always a privilege. Particularly where a task has felt complex before, if a new tool's different approach can make it simpler, this can help make modelling more prevalent.

C. Are there any specific research questions you would like to answer through the participation of the community?

Over a quarter of a century ago, empirical research showed poor support for collaboration in modelling tools was an important factor preventing their wider use. This served as a motivator in aiming at strong multi-user support from the

TABLE I
TASKS FOR ROLES

Metamodeller	Layout Expert	Climate Specialist	Technical Modeller	Model Creator
Add new property slot - for Technical Modeller	Rearrange diagrams as in Fig. 4:	Update temperatures - warmer & narrower range	*wait for Metamodeller*	Create new models: - storm season
Add/Update rules Update symbols	- Sensors on left or above - Actions on right or below	Update velocities and accelerations - more open, rougher seas	Enter new property values	- tourist season

start in MetaEdit+. It will be interesting to see whether what we wrote back then [8] could still be found today:

"Even standard CASE tools have been slow to move from single to multi-user support. Empirical research has shown the current lack of multi-user support in CASE tools is a serious problem [23], [24]. In particular, Selamat et al. [25] found that lack of multi-user support was the single largest CASE-specific reason why CASE tools were not being adopted in Malaysia. In addition to these questionnaire-based surveys of organisations, an empirical laboratory examination by Vessey & Sravanapudi [26] found that support for co-operative working was poor in current CASE tools."

VI. CONCLUSION

A. Summary of the case and solution

In this paper, we offer a case of collaboration between metamodellers and several modellers of various roles that we have commonly encountered in industrial projects. The aim is to introduce users to a way of collaboration that is faster and easier than file-based diff and merge. The metamodel and models are extended and updated by users working simultaneously, with the work of most touching all models — the hardest test for collaboration tooling. The domain is an Internet of Things consumer device, with the domain-specific modelling language offering familiar concepts of sensors, actions and states, so that new users should be able to understand quickly and work with minimal instruction.

B. Future plans for the tool support

Over the last few years, modelling tool customers have increasingly virtualized their tooling infrastructure, using virtual machines, remote desktops and private clouds. A logical next step is to provide tool evaluation, piloting and even long term use via cloud services offered by the tool provider. Particularly with respect to evaluation, a workshop setting with neophyte users could provide valuable feedback on using such a cloud service.

REFERENCES

[1] A. Bagnato, E. Brosse, A. Sadovykh, P. Maló, S. Trujillo, X. Mendialdua, and X. De Carlos, "Flexible and scalable modelling in the mondo project: Industrial case studies," in *Workshop on Extreme Modeling, co-located with MODELS*, vol. 1239, 09 2014.

[2] J.-P. Tolvanen and S. Kelly, "Model-driven development challenges and solutions: experiences with domain-specific modelling in industry," in *International Conference on Model-Driven Engineering and Software Development MODELSWARD*. IEEE, 2016.

[3] M. Franzago, D. D. Ruscio, I. Malavolta, and H. Muccini, "Collaborative model-driven software engineering: A classification framework and a research map," *IEEE Transactions on Software Engineering*, vol. 44, no. 12, pp. 1146–1175, 2018.

[4] A. Bucchiarone, J. Cabot, R. F. Paige, and A. Pierantonio, "Grand challenges in model-driven engineering: an analysis of the state of the research," *Software and Systems Modeling*, vol. 19, no. 1, pp. 5–13, Jan 2020. [Online]. Available: https://doi.org/10.1007/s10270-019-00773-6

[5] S. Kelly, K. Lyytinen, and M. Rossi, "MetaEdit+: A fully configurable multi-user and multi-tool CASE and CAME environment," in *International Conference on Advanced Information Systems Engineering*. Springer, 1996, pp. 1–21.

[6] Haltian, "Thingsee One IoT device," 2015. [Online]. Available: https://haltian.com/reference/thingsee-one-iot-device/

[7] S. Kelly, "Application of repository technology and concepts to a metaCASE environment," in *Towards a comprehensive metaCASE and CAME environment: conceptual, architectural, functional and usability advances in MetaEdit+*. University of Jyvaskyla, 1997, ch. 5.

[8] ——, "CASE tool support for co-operative work in information system design," in *Information Systems in the WWW Environment, IFIP TC8/WG8.1 Working Conference on Information Systems in the WWW Environment, 15-17 July 1998, Beijing, China*, ser. IFIP Conference Proceedings, C. Rolland, Y. Chen, and M. Fang, Eds., vol. 115. Chapman & Hall, 1998, pp. 49–69.

[9] ——, "Collaborative modelling with version control," in *Software Technologies: Applications and Foundations*, M. Seidl and S. Zschaler, Eds. Cham: Springer International Publishing, 2018, pp. 20–29.

[10] ——, "A matrix editor for a metaCASE environment," *Information and Software Technologies*, vol. 36, no. 6, pp. 361–371, 1994. [Online]. Available: https://doi.org/10.1016/0950-5849(94)90036-1

[11] S. Kelly and R. Pohjonen, "Dynamic symbol templates and ports in MetaEdit+," in *Proceedings of the 2013 ACM Workshop on Domain-Specific Modeling*, ser. DSM '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 19–20. [Online]. Available: https://doi.org/10.1145/2541928.2541932

[12] S. Kelly and J.-P. Tolvanen, "Automated annotations in domain-specific models: Analysis of 23 cases," in *Proceedings of FPVM 2021: 1st International Workshop on Foundations and Practice of Visual Modeling*, A. Di Salle, A. Pierantonio, and J.-P. Tolvanen, Eds., 2021.

[13] ——, *Domain-specific modeling: enabling full code generation*. John Wiley & Sons, 2008.

[14] J.-P. Tolvanen and S. Kelly, "Effort used to create domain-specific modeling languages," in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2018, pp. 235–244.

[15] ——, "Applying domain-specific languages in evolving product lines," in *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B*, ser. SPLC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 40–41. [Online]. Available: https://doi.org/10.1145/3307630.3342389

[16] A. Iung, J. Carbonell, L. Marchezan, E. Rodrigues, M. Bernardino, F. P. Basso, and B. Medeiros, "Systematic mapping study on domain-specific language development tools," *Empirical Software Engineering*, vol. 25, no. 5, pp. 4205–4249, Sep 2020. [Online]. Available: https://doi.org/10.1007/s10664-020-09872-1

[17] M. Maróti, T. Kecskes, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendoszky, and A. Ledeczi, "Next generation (meta)modeling: Web- and cloud-based collaborative tool infrastructure," *CEUR Workshop Proceedings*, vol. 1237, pp. 41–60, 01 2014.

[18] T. Ma and J. Sallai, "MiW: A domain specific modeling environment for complex molecular systems," *Procedia Computer Science*, vol. 108, pp. 1232–1241, 12 2017.

[19] J. Corley, E. Syriani, and H. Ergin, "Evaluating the cloud architecture of atompm," 01 2016, pp. 339–346.

[20] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Computing Surveys*, vol. 15, pp. 287–317, 1983.

[21] ISO/IEC 9075-2:2016, "Information technology - database languages - SQL - part 2: Foundation (SQL/foundation)," International Organization for Standardization, Geneva, CH, Standard, 2016.

[22] A. Taivalsaari and S. Vaaraniemi, "TDE: supporting geographically distributed software design with shared, collaborative workspaces," in *Advanced Information Systems Engineering, 9th International Conference CAiSE'97, Barcelona, Catalonia, Spain, June 16-20, 1997, Proceedings*, ser. Lecture Notes in Computer Science, A. Olivé and J. A. Pastor, Eds., vol. 1250. Springer, 1997, pp. 389–408. [Online]. Available: https://doi.org/10.1007/3-540-63107-0_28

[23] S. Stobart, A. van Reeken, G. Low, J. Trienekens, J. Jenkins, J. Thompson, and D. Jeffery, "An empirical evaluation of the use of CASE tools," in *Proceedings of 6th International Workshop on Computer-Aided Software Engineering*, 1993, pp. 81–87.

[24] E. Rupnik-Miklič and J. Zupančič, "Experiences and expectations with CASE technology—an example from Slovenia," *Information Management*, vol. 28, no. 6, p. 377–391, Jun. 1995. [Online]. Available: https://doi.org/10.1016/0378-7206(94)00050-S

[25] M. Selamat, C. Choong, A. Othman, and M. Rahim, "Non-use phenomenon of CASE tools: Malaysian experience," *Information and Software Technology*, vol. 36, no. 9, pp. 531–537, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0950584994900981

[26] I. Vessey and A. P. Sravanapudi, "CASE tools as collaborative support technologies," *Commununications of the ACM*, vol. 38, no. 1, p. 83–95, Jan. 1995. [Online]. Available: https://doi.org/10.1145/204865.204882