Collaborative Modeling and Metamodeling in the Cloud with Versioning

Steven Kelly
MetaCase
Jyväskylä, Finland
stevek@metacase.com
ORCID: 0000-0003-0931-157X

Juha-Pekka Tolvanen
MetaCase
Jyväskylä, Finland
jpt@metacase.com
ORCID: 0000-0002-6409-5972

***Abstract*—This tutorial investigates and demonstrates proven solutions for industrial scale collaborative modeling, maintaining a version history in standard tools as the models and modeling language evolve on the fly. It will show how to achieve smooth collaboration between multiple simultaneous modelers. The aim is low effort and low ceremony for modelers, while maintaining the use and benefits of version control systems familiar from coding. Alongside the models, the modeling language can also be evolved collaboratively in a similar way, with minimal work for metamodelers and minimal impact on modelers. The evolution of the models and metamodels will be recorded in the version history, with participants able to see their own changes and document them in version comments. Participants will gain practical experience and understanding by taking part in the modeling and metamodeling using the provided no-install desktop or browser-based tools — or their own tools, prepared ahead of time with the materials. Explanations, discussions and tasks will help all participants think through the hows and whys of the collaboration mechanisms.**

Keywords—collaboration, modeling, metamodeling, multi-user, versioning, cloud, domain-specific modeling, co-evolution

I. Tutorial Information

A. Short bios

Dr. Steven Kelly is the CTO of MetaCase and co-founder of the DSM Forum. He has over thirty years of experience of consulting and building tools for Domain-Specific Modeling. Steven is the architect and lead developer of MetaEdit+, MetaCase's domain-specific modeling tool. He is co-author of a book on Domain-Specific Modeling and has published over 70 articles in various software development journals and conferences. Steven has a Ph.D. in computer science from the University of Jyväskylä and a Master's degree from Cambridge.

Dr. Juha-Pekka Tolvanen is the CEO of MetaCase and co-founder of the DSM Forum. He has been involved in model-driven development and tools, notably metamodeling and code generators, since 1991. He has acted as a consultant world-wide for modeling language development, authored a book on Domain-Specific Modeling, and written over 70 articles for various software development magazines and conferences. Juha-Pekka holds a Ph.D. in computer science and he is an adjunct professor (docent on software development methods) at the University of Jyväskylä.

B. Proposed length

1.5 hours

C. Level of the tutorial

Beginner to Advanced

D. Target audience

This tutorial is intended for anyone interested in how to integrate the work of multiple modelers working on the same set of models. The approach shown is independent of the modeling language, programming language and version control system.

There are no prerequisites beyond average modeling skills.

II. Tutorial Description

This tutorial investigates and demonstrates proven solutions for industrial scale collaborative modeling, maintaining a version history in standard tools as the models and modeling language evolve on the fly. It will show how to achieve smooth collaboration between multiple simultaneous modelers. The aim is low effort and low ceremony for modelers, while maintaining the use and benefits of version control systems familiar from coding. Alongside the models, the modeling language can also be evolved collaboratively in a similar way, with minimal work for metamodelers and minimal impact on modelers. The evolution of the models and metamodels will be recorded in the version history, with participants able to see their own changes and document them in version comments.

Participants will gain practical experience and understanding by taking part in the modeling and metamodeling using the provided no-install desktop or browser-based tools — or their own tools, prepared ahead of time with the materials. Explanations, discussions and tasks will help all participants think through the how and why of the collaboration mechanisms.

A. Tutorial methods

Most of the tutorial will be practical, with additional slides and discussion based on participant interest.

In the practical parts of the tutorial, participants will collaborate as metamodelers and modelers in various roles that we have commonly encountered in industrial projects. We begin with an existing metamodel and small set of models. The

metamodel and models are extended and updated by the participants in parallel, with the work of most touching all models — the hardest test for collaboration tooling. The domain-specific modeling 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 and their language is such that participants will be able to understand them and work with them quickly, without needing to learn a new domain. Similarly, the participants will not need to install modeling tools, but will be able to work with the graphical models and the metamodels in no-install desktop or browser-based editors in the cloud.

III. Modeling Case and Outline

A. *Modeling language problem domain and context*

"The year is 2030. 15 years earlier, Juha-Pekka made a domain-specific modeling language for the Internet of Things, the pinnacle of cool tech back then. The language allowed modelers to create programs for a Thingsee One [1], 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."

B. *Modeling language used*

The modeling 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 modeling language is based on the sensors and services of an Internet of Things device, the Thingsee One. The language offers the modeler 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 modeler 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

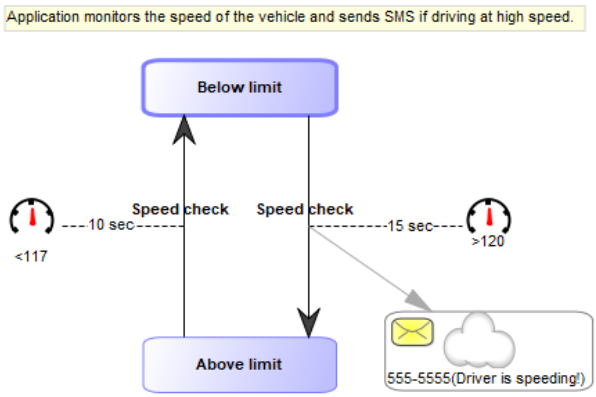


Fig. 1 Thingsee Purpose model to send warning when speeding

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 a MetaEdit+ [2] Diagram Editor in Fig. 2.

C. *Collaborative needs in the case: team roles, views, etc.*

Modeling 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 a tool can cope with this, it will have no problem in situations where modularization and division of labor help keep users from treading on each other's toes even without tool help.

Where work on models is divided according to skills, we have seen several common patterns over the years. The most

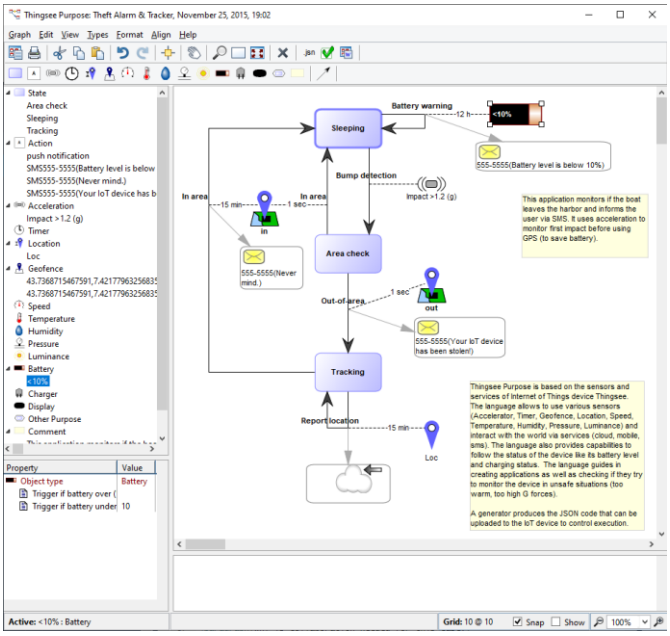


Fig. 2 Thingsee Purpose model for a theft alarm and tracker.

common differentiator is **metamodelers** and modelers: although the metamodeler may also model, most modelers will not metamodel. Another common case is to have a group of modelers responsible for **initial models**, at the level an end-user could understand, and a second group of modelers 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 behavior: 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 modeling 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: Metamodeler, Model 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 Metamodeler will update the metamodel, affecting all models, and the Technical Modeler, Climate Specialist and Layout Expert will all be working simultaneously on all the models. The Model Creator's work will depend on the Metamodeler'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.

Depending on time and numbers of participants, more than one participant could play each role, or the participants could be split into groups and the material and roles duplicated for each group. The roles can be allocated according to participant experience level and interest. The task list for each role has elements that could be omitted if necessary, and extra tasks are available for users of any role to add. The details of the tasks and allocation will be adjusted to suit the number of participants and ensure a useful learning experience: enough material to work on, but also a small enough set that conflicts between simultaneous users would become apparent if the tool did not enable collaboration sufficiently.

IV. TOOL

MetaEdit+ [2] (metacase.com/products.html) is a language workbench and modelling tool offering strong multi-user support [3]–[4] and version control integration with no need for manual diff and merge [5]. It supports multiple simultaneous modelling languages, multiple representations of the same model as matrices, tables and text as well as diagrams — which go beyond bitmaps or boxes to offer dynamic graphical languages with real-time synchronous feedback in symbols [6]. It has a particular focus on domain-specific modelling with full code generation [7] and ease of language creation and evolution [8]–[9]. As well as being installed as a desktop application it can also be used from the cloud, accessed via the browser or as a desktop-integrated remote application with no local installation needed. The cloud installation will be provided for use in this tutorial.

Participants experienced with their own tool that is suitable for these tasks may also use that tool, if they prepare it in advance. The materials (language definition, models and tasks) will be made available in advance for such tools. Experience with collaboration, evolution and versioning with various tools will of course be most welcome, leading to fruitful discussions.

V. NOVELTY OF THE TUTORIAL

Portions of this tutorial were used successfully as part of the MetaEdit+ hands-on case study in the HoWCoM workshop at Models 2021. The focus there was on a short demonstration of multiple geographically distanced collaborators. Here we can expand to focus more on the how and why of the collaboration mechanisms, and add the integration with version control systems.

VI. REQUIRED INFRASTRUCTURE

A large projector screen would be useful to enable participants to see more than one user's display at once. Two screens could also work.

Participants will need Wi-Fi with reasonable speed and reliability for a remote desktop connection to the cloud server (peaking at 1 Mbps per participant and several Mbps for the sum over participants).

REFERENCES

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

[2] 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.

[3] 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.

[4] S. Kelly, "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.

[5] S. Kelly, "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.

[6] 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.

[7] S. Kelly and J.-P. Tolvanen, Domain-specific modeling: enabling full code generation. John Wiley & Sons, 2008.

[8] 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.

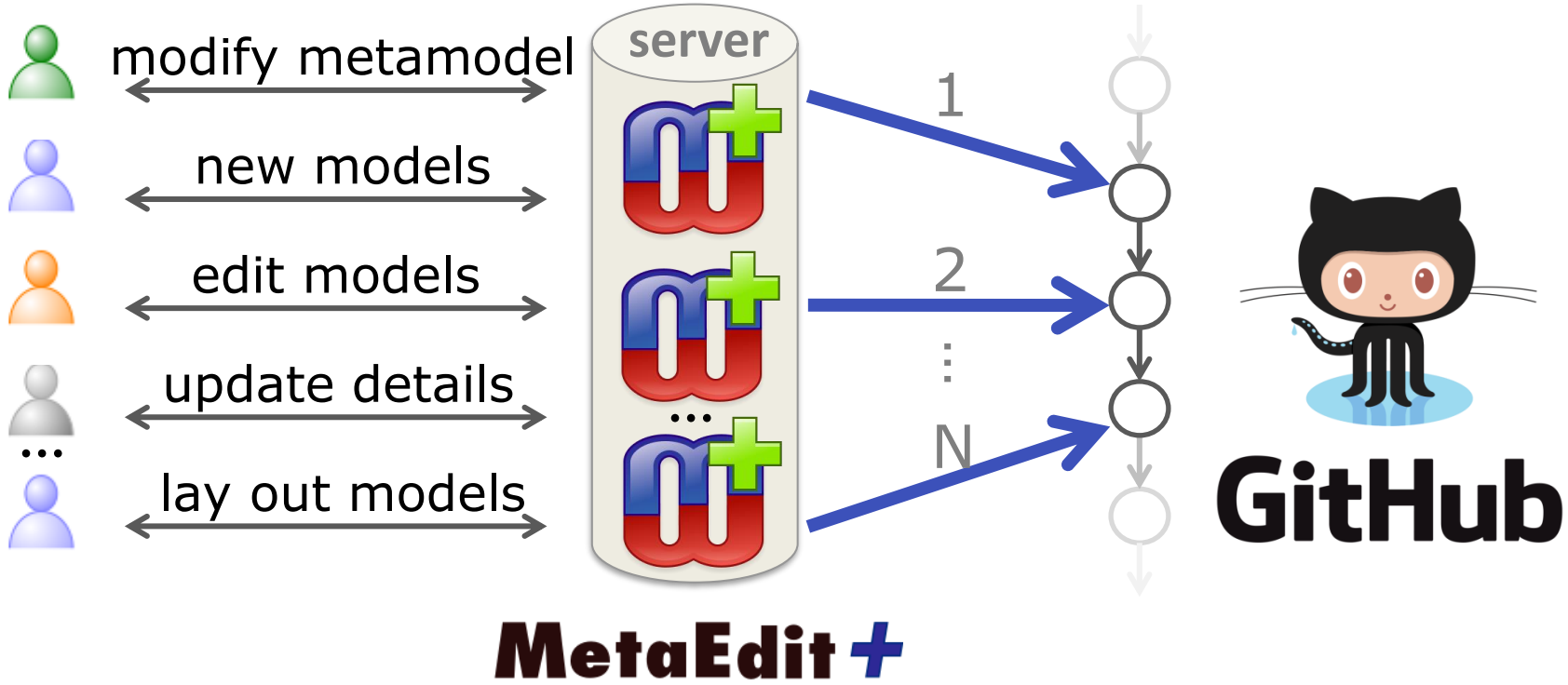
[9] J.-P. Tolvanen and S. Kelly, "Applying domain-specific languages in evolving product lines," in Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B, ser. SPLC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 40–41.

MetaCase

Collaborative Modeling and Metamodeling in the Cloud with Versioning

Steven Kelly, Juha-Pekka Tolvanen
stevek | jpt @metacase.com

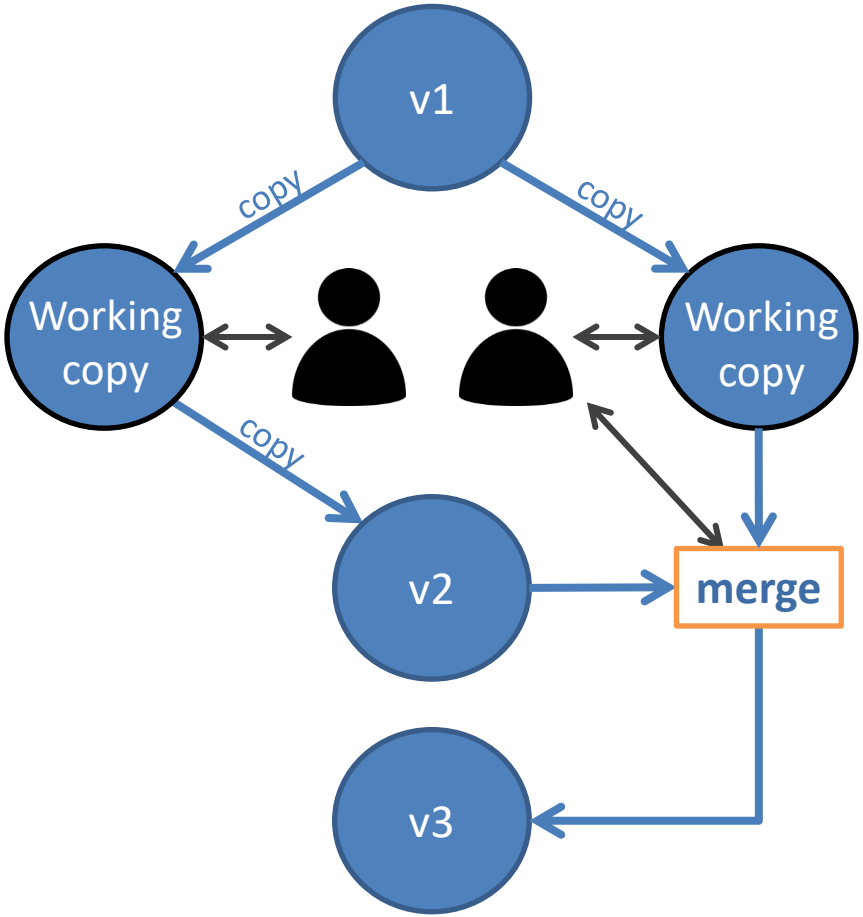
Collaborative (meta-)Modeling and Versioning of Models and Languages

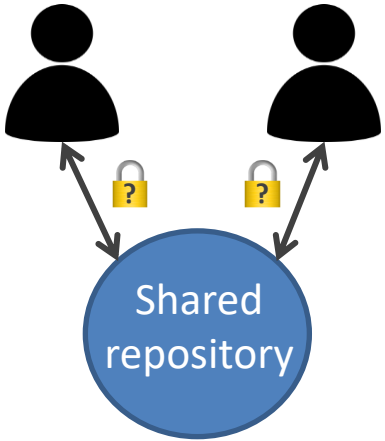


Clone *vs.* Share

merge after the fact

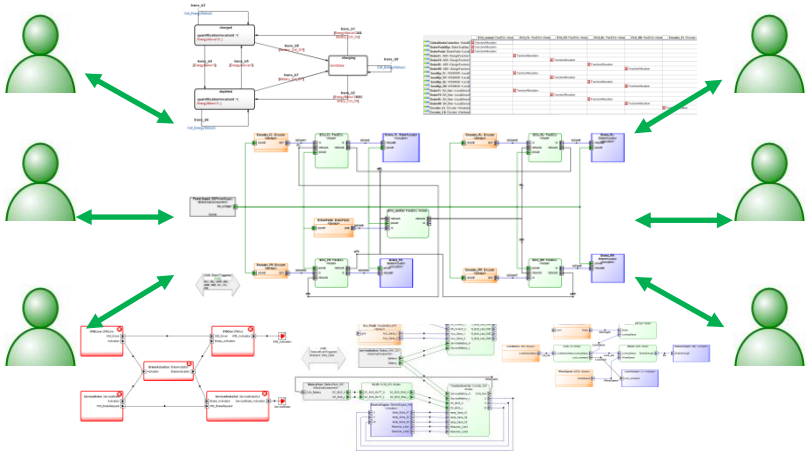
continuous integration



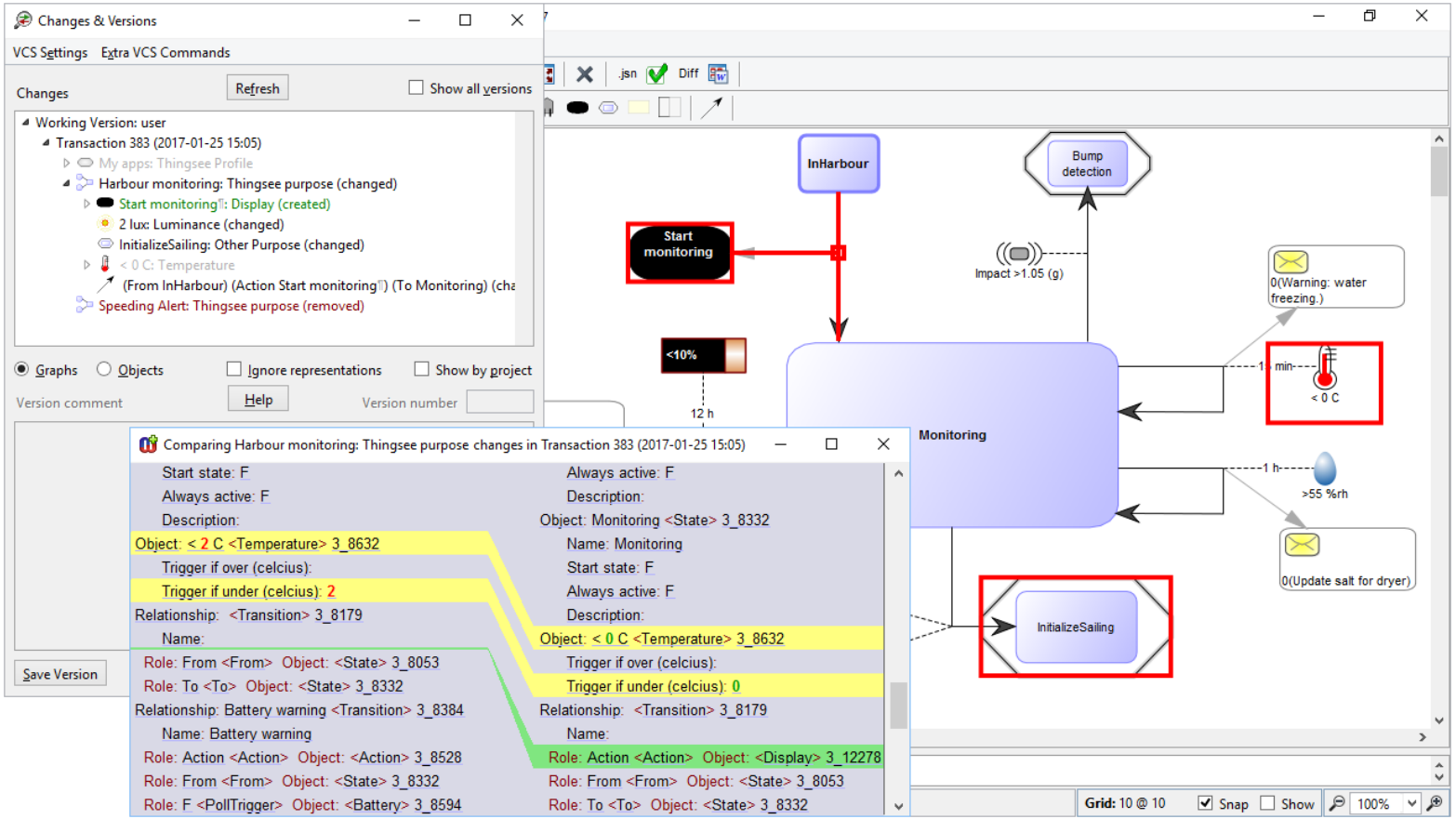


MetaEdit+

- Mature, commercial, supported Language Workbench
- Collaborative modeling support
  - Several developers
  - Multiple models
  - Multiple languages
  - Tracks history & changes
  - Integrated with VCSs
- **Collaborative language engineering**
  - **Updates existing models**
  - **Integrates with VCSs**



Compare changes between versions



Collaboration at the object level

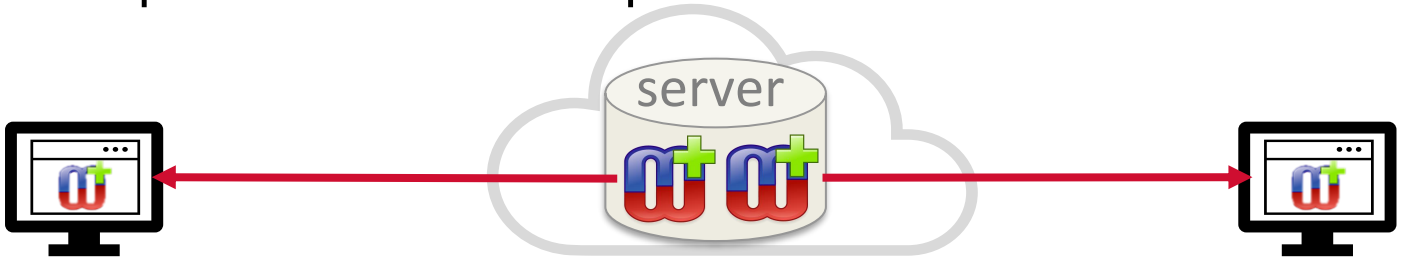
- Repository-based: users work in parallel on same data
  - No need to merge
  - Models and metamodels can be edited simultaneously
- Locks ensure no conflicts
  - Fine granularity, so can work closely without lockout
- Low ceremony: locking and unlocking is automatic
  - Users can focus on work, not tool
- Design transactions: minutes to hours
  - Commit a coherent set of changes together
    - Releasing half-finished/inconsistent data would confuse others
  - See a consistent version of repository during transaction
    - Avoid being distracted by others' changes appearing live

RSL: Remote Shared License

- IT departments want to avoid installing software
- Want to allow users to work from anywhere

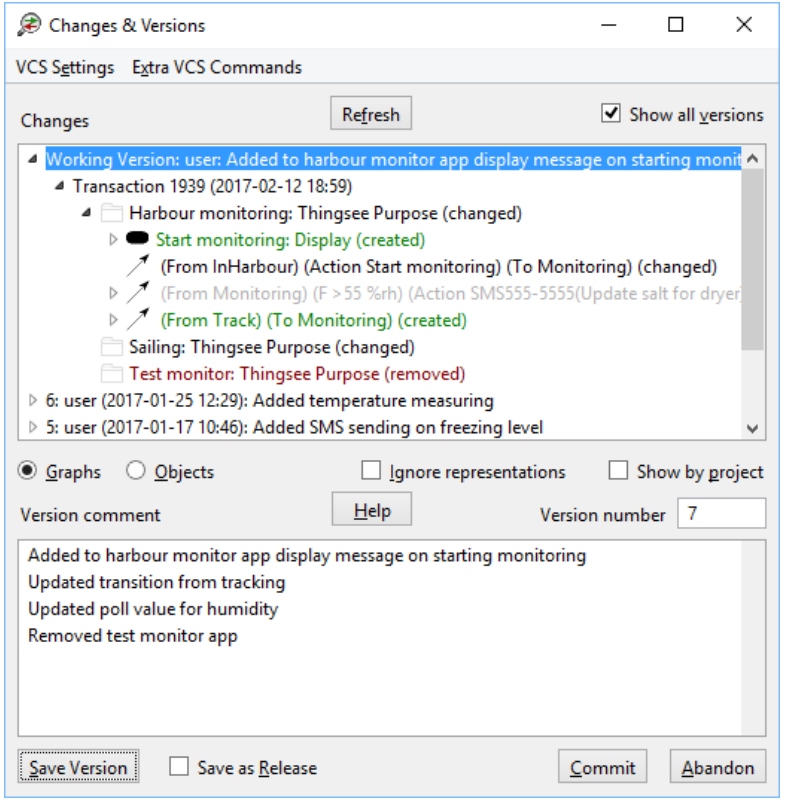
- Graphical modelling has high performance reqs at client
- Large repositories are slow to move over a network
  - Particularly with high latency over VPNs + load as needed

- Put MetaEdit+ server and client on one Windows Server
- Multiple remote desktop connections from users



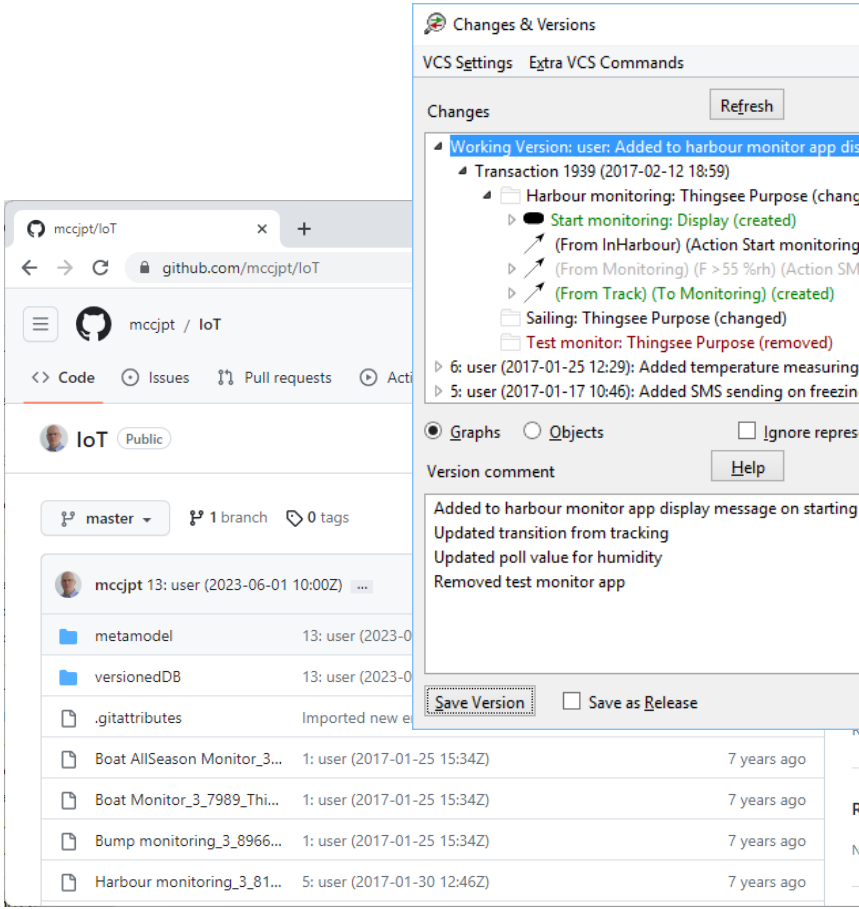
Changes and Versions tools

- Automatic trace of model changes
- See changes graphically directly in your models
- View changes as a tree using your language's structure and symbols, not XML
- Compare changes as a textual diff with live links to models
- Filters (your changes/all changes, data/representation)



Versioning with external VCSs

- Simple setup and use
- Automated background execution of versioning commands
- Maintain same model in several places, syncing with GitHub etc.
- Version and diff your language and generators along with models
- Git, SVN predefined
  - extend with your own



Task overview by role (in parallel)

Metamodeler	Layout Expert	Climate Specialist	Technical Modeler	Model Creator
Add new property slot	Rearrange diagrams: - Sensors left - Actions right	Update temperatures - warmer, stabler Update velocities - rougher seas	*Wait for metamodeler*	Create new models: - storm season - tourist season
Add/update rules Update symbols			Enter values for new property	

Thank you!

Questions?
Experiences?
Arguments?