MetaCase

New UX for Participatory Modeling

…a vision paper…

Steven Kelly
stevek@metacase.com

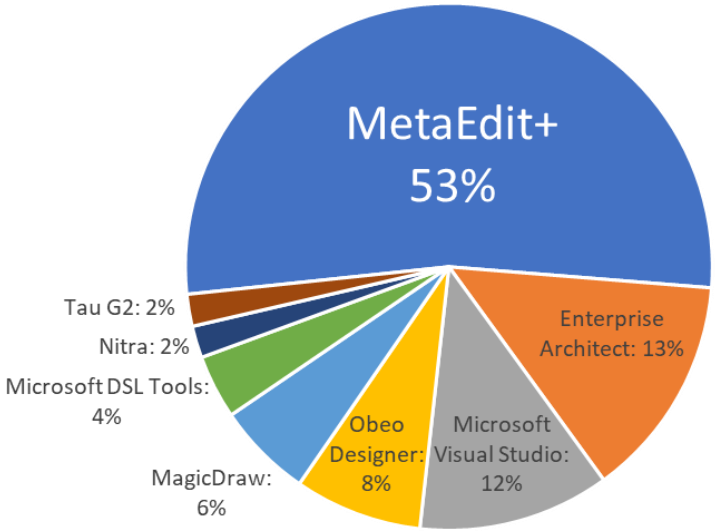
CoPaMo, MODELS 2024
Sun 22 Sept 2024 11:44

MetaEdit+

- Mature, commercial, supported Language Workbench
  - 10s of years, 100s of DSLs, 1000s of commercial users (industry and academic)
- Collaborative modeling
- Diagram Editor, Matrix Editor, Table Editor, various Explorers
- Metamodelling tools, graphical Symbol Editor, Generator Editor + Debugger
- Most widely-used commercial tool in research and academia

Reported use of commercial DS(M)L tools, 2012–2020

Systematic mapping study on domain-specific language development tools, Iung et al., Empirical Software Engineering 25(1), 2020



Tooling for Participation in Domain-Specific Modeling

Tooling for participation:

- Technical facets of collaborative modeling
  - **Multi-paradigm** and **multi-view** aspects of collaboration
- Organizational+human facets of participatory modeling
  - **Methods** for designing participatory modeling: **tools**

Domain-Specific Modeling for participation:

- Language of requirements: non-techie participants <3
- Formal, generates full system: techie participants <3

Tools for Domain-Specific Modeling

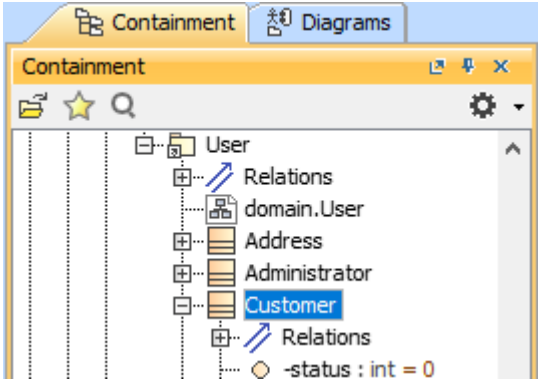
- Language Workbench creates modeling tools efficiently
  - 2000x faster than coding with graphics+model frameworks
- Many things can be done to accommodate participants
  - Filtering, hiding details, tweaking visual representation
- But if you have to drop down to coding, that 2000x hurts!

- **Textual**: Xtext, Spoofax, Rascal *not graphical:* 7%
- **Graphical**: MetaEdit+, MS DSL Tools, Sirius, GEMS 68%
- **Projectional**: MPS, Intentional, Whole Platform 17%

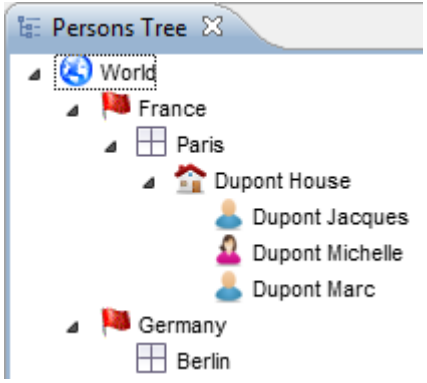
UX Areas in Modeling Tools

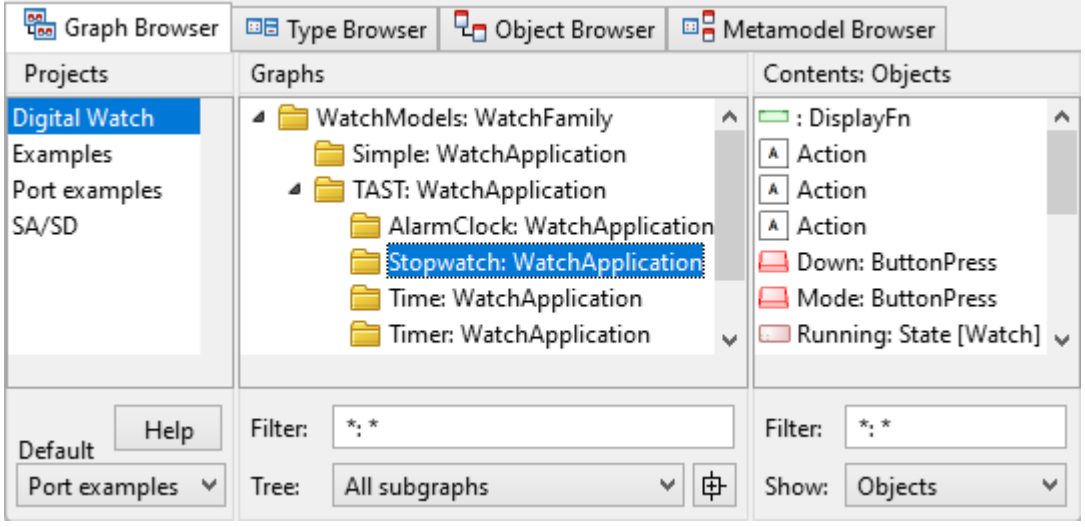
- **Explorers** *for navigating to models and their elements*
  - Project explorer: workspace, solution, projects
  - Models explorer: by type, containment hierarchy
  - Model explorer: objects, sub-objects, sub-models
- **Editors** *for creating, editing and viewing models*
  - Diagram
  - Table
  - Matrix
- **Property views** *for editing model element details*
  - Property dialog (traditional UI widgets)
  - Property sheet (simple grid with mostly textual display)

Explorers



- Standard widgets
- Simple fixed queries
- Maybe filter, sort etc.
- Low hanging fruit?



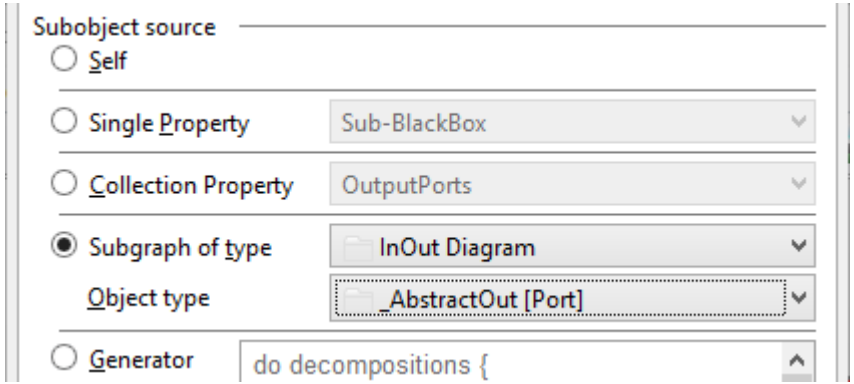


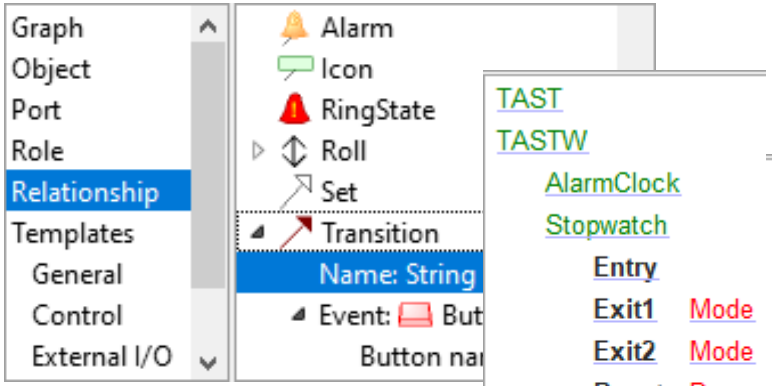
Build your own Explorer!

Build your own Explorer!

- Query starting point
  - Current project / graph / object
- Accessor or navigation path
  - => List, Tree, Table, Tree+Table
- Filter, sort
- Display format
  - Icon, Name, Type

Query Builder for participants' needs





- `do >Transition { :Name; ' '; :Event; newline }`
- `do >Transition; where :Guard='' {…}`
- `do graphs { :GraphName; newline`
  `do >Transition {' '; :Name; … newline }`
  `}`

Liveness of New UX

- Just get the results as text, a list, table, tree etc.
- Keep result open, save/export it
- Output is Live: inspect elements, dive into them
- Save & repeat query, share
- Automatic update of result as models change
  - A user could also choose to stop this
- Keep same elements +boilerplate, but names update
  - Better to rerun whole query? Name affects order, filtering

- Meta: Actual UI doesn’t need to update after opening
  - E.g. if definition of new UX query changes

Bidirectionality of New UX

- Would editing the results affect source models?
  - Changing order? Removing? Adding?
- Similar to familiar questions:
  - reverse engineering
  - editing generated code
  - auto-layout vs. remembering positions
    - Computer doesn’t need positions (or names!); human does.
- The non-modeling participant is actually modeling then

Conclusions of a meta-explorer

- Allow building explorers for participants' needs
  - Non-modeling participants often ask 'show me all X's here'
- Seems not to have been described or built before
- Closest are *ad hoc* browsers from 1990s CASE tools
  - Excelerator 'spider diagrams': just one step, all accessors
  - TDE Navigator: each step chosen each time, confusing
- Fast and simple enough for all participants
  - Tool developer; metamodeler / modeler / participant
  - Liveness seems worthwhile; bidirectionality hard, even bad
- Next step: built a prototype, test with users

Thank you!

Questions? Experiences? Arguments?

Co-evolution Tutorial: Mon 16:00 T6
Co-evolution keynote: Tue 9:30 ME'24
Industry Day talk: Tue 12:07