Discussion on DSM topics during workshop 27 Oct 2015
Please post questions, comments, discussion ideas on the lines below...

- How can we model system behavior in a more abstract way? Are automata suitable?
    - Perhaps there are some problems where the model of computation provided by automata is not sufficient (?) Any ideas on what those could be?

- How must we formalize tool chains to fully embrace MDE?
    - What do you mean by “formalize”?
        - Make tools model aware, not only process models

- Patterns for template generators still need adult supervision. When/how can that supervision come from automatically parsing exemplary code? or exemplary models?

- Composition and language reuse is still a very important area, because we do not want to lose the results we got from other efforts
    - +1
    - Doing a slice as melange suggested would be nicest. Rarely does the whole kit and kabootadoodle get reused.

- I think DSM still has the promise to bridge from “certifiable” to “verifiable” code, based on enforced annotations, style, and structure. What domains are the lowest of the fruit-hangers for this idea?

- Brian’s talk used the idea of Block languages in the context of DSM. Such languages are very popular in the K-12 education area and have not been tapped very well for general usage in the DSL context. What benefits do Block languages (which are also syntax directed, and appear to be textual with some visual boundaries) offer? What are some challenges of their use? Can they offer more guidance in specifying a system than traditional metamodeling tools (specifically, those that are graphical)?
    - In this case we are considering visual programming (as an example). Has anybody experience with Greenfoot (although it is restricted to Java)
    - Decision on Finland starting 2015 is to teach “coding” from the beginning of elementary school (age 7 focus on basic operations, steps etc) aims to move towards visual programming languages (age 10 forward) like block languages a’la Lego Mindstorm.

EDUCATIONAL CONCERNS

- How should we teach MDSE and Language Engineering for graduate students? Which domain examples should we select to exercise? Which examples are appropriate for model transformations?

    - **Example 1**: In one course mentioned, students (EE, CS, other engineering students) all know about OO and software dev; lead into how UML can describe structure; then how hard UML may be hard for some experts to use; leads the class to DSM ideas
    - **Example 2**: Another grad course has all students know about software; question is what is the objective of the course - 1) basic part of DSM, and 2) project-based design to create a DSL for a specific domain and create a generator. Finding the domain is key - students should have some knowledge of the domain. Model transformation should also be a goal of the course and how the domain is selected.
        - Tool chaining may be too difficult to students; what to do as a teacher; use xText and Xtend
        - Do we need a “dumbed down” metamodeling environment for education activities, instead of Eclipse-based tools?
        - When do students get the “aha” moment when they see that DSM has advantages.
        - Including maintenance can help to show the benefits.
        - Group keeps coming back to the domain as being very important. Domain must be complex, perhaps, but not overwhelming.
        - Would be good for a group to define a really well polished language, transformations to showcase the benefits.

    - What is the earliest age that we should teach these ideas? Assume a student at age X knows how to program in a general purpose language. At what point (X+Y) should we teach DSM ideas?
        - Is there an activity that students currently do at a young age that is comparable to modeling? That is, as DSML’s often try to be natural to the user and domain, I am sure there are examples of activities (maybe some sort of drawings) that students currently do that might meet a (loose) definition of DSM.
    - What cognitive levels does a student need to achieve to be taught DSM ideas? Is the concept of abstraction important as a pre-req?
        - That is very Piaget-ian ;)
            - Yes! Also, see Jeff Kramer’s paper from around 2007 in CACM.
    - What are the main learning/teaching objectives für such a course?
    - Does this relate to this idea of Computational Thinking?
    - This was a helpful example when I was taking a MDE course: http://engineering.vanderbilt.edu/news/2013/vanderbilt-wins-9-3m-darpa-contract-to-evolve-tools-for-military-vehicle-design/

- Hard for a tool to be “everything to everyone” to span both education and real industrial practice.
    - Learnability: A good tool should start out simply, and allow the user to grow with it

EXAMPLE DOMAINS

- There is a lot of talk in the SE world about the idea of sustainability. What can DSM approaches bring to support the sustainability efforts?
- Automotive and avionics domains are often cited as some of the best examples where identified benefit has been seen with DSM approaches. What are some other domains, possibly not even explored much yet, where DSM might offer a “killer app”?
    - From educational perspective, hard to have students buy into these examples.

DOMAIN-SPECIFIC vs GENERICITY

- Are there too many different DSLs?
    - Are there too many domains?
    - Ongoing question of how to integrate DSLs used to describe separate parts of system behavior.

- Should we continue to put effort into domain-specific solutions or attempt more generic approaches?
    - An old adage from the past decade suggested that the more narrow and tight the domain, the better potential for benefit. Is this still true (or was it always a myth)?
    - I think there is still a lot of value in working in a restricted space (ie, domain space rather than general) such as promoted rapid development and better validation, etc
    - I don’t think it was ever a myth (in the sense of untruth) but it was a myth (in the sense of explaining how it could work) ;) Use meta-tools to describe the family of all DSM/DSL languages, but use composition (where appropriate) to built whole from parts is visually easy---semantically complex and in some cases not possible
    - I think languages that can (incrementally) be extended can help by being a middle ground between GPLs and DSLs.

FUTURISTIC OUT OF THE BOX IDEAS

- Could there be any value to implementing DSMLs in VR or AR?
    - Tango - better way to program with Rift or Tango; how to leverage that is not available otherwise
    - That would be a neat idea! Maybe it depends on the domain? Would some domains be better suited than others, and if so, what are some examples of characteristics of those domains?
    - Demonstration-based approaches
    - What does a metamodel mean in this context?

TOOLING CHALLENGES

- There has been a lot of low-lying fruit in the DSM space by looking at traditional programming languages and their toolsets, and finding common features that are missing in the DSM context (e.g., debuggers, testing, version control). What remains to be done that is needed in metamodeling tools or language workbenches?
    - What about collaboration, such as what we are doing in this document? Can anyone provide literature that exists or ongoing work where collaborative model editing is available? How useful even is that idea?
    - The MetaEdit guys always say they support concurrent editing…
        - This works in MetaEdit+ for both modeling level and at metamodeling level: http://www.methodsandtools.com/archive/collaborativedsl.php
    - Also, Eclipse WhateverTheNameOfThatEMFExtensionIs supports it.
        - CDO does real-time collaboration
        - EMFStore is a transactional database
        - GenMyModel
        - WebGME supports this (version control and real time collaboration)
        - MDEForge (http://www.mdeforge.org) is an online EMF-friendly modeling platform

- Why is our tooling generally still bad?
    - A panel at MODELS compared current toolsets to the game of Pong from the 1970s, rather than a modern high-end game with graphics. It seems that we have a lot of work to go in this area
    - I am quite surprised by such statements when looking at the MPS stuff. It is in no means worse that IDEs for programming languages. Are those IDEs also Pong-era?? Not saying it can't be improved, but Pong-era is not a good way of characterizing it.
    - Positive: xText and MPS can build just as good as GPL editors; maybe people are just using wrong tools (?)
    - Eclipse discussion:
        - Benefit is plugin architecture, large user base (in general, for modeling tools)

- How important is web support for a language tool?
    - What about cloud support for web-based tools, or is that assumed in the above?
    - That's a good point. It was not assumed.
    - Connectivity and legal issues are a barrier to the cloud.
    - Examples (not exactly what was asked but might be useful for discussion):
        - http://webgme.org/
        - http://www.metamorphsoftware.com/metamorph-tools
        - ATOMPM
    - Reduces install time

- “Editor, Editor, Editor” is a common refrain, at what point can the editor be configured in a nice way s.t. more Eclipse-like fixes are used to enforce DSM-ish behaviors
    - “did you mean to use (varname)” can become an autofix if necessary
    - “did you want to have this code verifiable? probably change it in the following way...”
    - How is this different from Quick Fixes that are already available?
    - Many newbies are not sure where to start
    - Domain-intelligence auto-complete
    - Language workbench contest in 2013 - only two tools supported such a feature
    - Quick fixes should balance offering suggestions while not becoming overwhelming/annoying or getting in the way (like Clippy)
    - Possible education benefits
    - Should be possible to turn the suggestions off, or in specific types of situations