Visual languages and their usage in IDEs

Yurii Litvinov
y.litvinov@spbu.ru

29.12.2022

Visual Languages: big hopes

- First visual languages — 1970s, mainly for architecture
  - Much like blueprints in “classical” engineering
- “CASE boom” in 1990s — visual languages as next generation of high-level languages
- UML, RUP — 1995
- But then came Agile methodologies

Visual Languages: current state

In broad strokes

- Many IT companies don’t use visual languages at all
  - “We have N gigabytes of code but not a single UML model”
- Most open source projects lack visual models (and, in fact, actual architectural documentation)
- Some companies still base their development processes on visual modeling, mainly in mission-critical projects
  - E.g. DRAKON in Russian aerospace systems
- Last update of UML standard was in 2017
  - Which is not necessarily bad by itself
- Domain-specific modeling is gaining attention

Domain-specific modeling

- The idea is to specialize a language for a specific domain
  - Code generation becomes possible
  - Language can be much more concise and accessible for domain experts
- Domain-specific modeling enables end-user programming or “low-code/no-code solutions”
  - Affordable cloud solutions and IoT
  - E.g. Unreal Engine’s Blueprint, Webflow Logic, Microsoft Robotics Developer Studio, Robolab, NXT-G, TRIK Studio

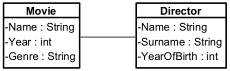
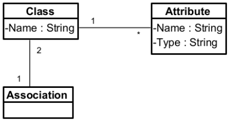
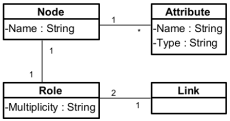
DSM platforms

- To specialize a language for a specific domain is to build a specialized tool every time
  - Not feasible except for very rare special cases
- Solution — tools to create visual tools: DSM platforms
  - E.g. MetaEdit+, Eclipse Sirius, QReal
- They use formal definition of visual language to generate tooling
- But how to define visual language?

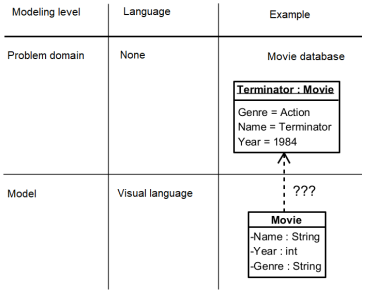
Formal definition of visual languages

- Metamodel — a model of all correct models
  - Much like grammar for textual languages
  - Grammars were used for visual languages too, but rarely
- Defines entities, their attributes (with types) and possible relations
- Metamodels can be textual and visual
- UML uses visual metamodel defined in UML standard
  - Standalone version is known as MOF

Metalevels

Modeling level	Language	Example
Problem domain	None	Movie database
Model	Visual language	 Movie -Name : String -Year : int -Genre : String Director -Name : String -Surname : String -YearOfBirth : int
Metamodel	Metalanguage	 Class -Name : String 1 * Attribute -Name : String -Type : String 2 1 Association
Meta-metamodel	Metalanguage	 Node -Name : String 1 * Attribute -Name : String -Type : String 1 1 Role -Multiplicity : String 2 1 Link

It’s not as easy



- Object in UML is not an instance of a class
- Multi-level and deep metamodeling were invented to fix such problems
  - Every model can be seen as a metamodel for a model below
  - E.g. class diagram as a metamodel of object diagram
  - Tools: MetaDepth, Melanee, REAL.NET

Visual modeling in IDEs

- Use existing library and build ad-hoc solutions on top of it:
  - IntelliJ IDEA — yFiles
  - Community plugins, e.g. Visual Studio Code and Diagrams.net
- Create own DSM platform and build tooling with it
  - Eclipse — EMF, GMP; de-facto standard for visual languages research, can do everything, but complex
  - Visual Studio — MS DSL Tools (Modeling SDK); little adoption as a standalone platform, but VS models built on top of it (kind of)

Visual modeling in IDEs, problems

- Round-trips with textual representation
  - Requires something like PSI
  - Models are considered as a view on an underlying code model, just like textual code
- Usability!
  - Actually, it is a general problem for diagram editors
  - Textual code editing is much less painful

Visual languages in SPbU

- RTST and earlier works (1980s) — SDL and Algol 68, for telecommunications
- REAL and REAL-IT (1990s) — UML 1.0, Visual Basic for information systems generation
- QReal (2005) — UML 2.0 (at least supposed to), metamodeling
- QReal:Robots/TRIK Studio (2011) — actual technology on QReal, widely spread in Russia as educational robotics tool
- REAL.NET (2016) — .NET and web version, multilevel metamodeling

Keypoints

- A «winter of visual modeling» seems to be close to an end, due to need for end-user programming
- Visual languages support in IDEs is lacking (but present), due to low interest from programmers
- None of the existing IDEs support actual UML standard
  - Most popular tools are only mimic UML, and do it badly
- Low-code solutions integrated into light-weight IDEs seem to be interesting vector of future work
- Visual languages support requires effort in the fields of language theory and usability
- There is a need for reusable assets for support of domain-specific visual languages