Advanced Metaprogramming

vadim@grammarware.net

Advanced Programming / Software Languages Team
Vadim Zaytsev, SWAT, CWI

CC BY-SA 2012
Software is complex

- The principles of software are easy
  - just a bunch of computer instructions
  - IO, arithmetic, control, done!
  - adv: patterns, concurrency, agile/formal — still easy

- The practice of software is incomprehensible
  - too much code
  - too much diversity
  - CPU is too fast
  - too much memory
Solutions/instruments?

- Higher level programming
- Program comprehension
- IT portfolio management
- ...
- Metaprogramming
Higher level programming

- Generative programming
- Domain-specific languages
- Model-driven engineering
- Metaprogramming
GP

- Obtain a collection of reusable components
- Always maximise the automation
- Aim at product lines, not at single programs
- Compose a system specification
- Generate the final system(s) from all of the above
• Choose a problem domain
• Compose a dictionary of domain concepts
• Make a language with those concepts
• Define mappings to some executable language
• Program/model in your own language
MDE

- Investigate application domain
- Encapsulate domain knowledge in a model
- Produce software artefacts from the models
- Model-based...
  - ...code generation
  - ...testing
  - ...architecture

Program comprehension

- Static analysis
- Dynamic analysis
- Integrated development environments
- Metaprogramming
Static program analysis

- Analyse the structure of the source code
- Control flow analysis
- Data flow analysis
- Model checking
- Abstract interpretation
- Verification of assertions in the code
Dynamic analysis

- Analyse the behaviour of the program in the runtime
- Model checking
- Testing + code coverage criteria
- Simulation
IDEs

- Source code editor
  - with syntax highlighting
  - with code completion
  - with advanced navigation

- Debugger
  - with runtime memory inspection
  - with stepwise algorithm execution

- Compiler
  - with dependency hell resolution system
IT portfolio management

- Mining software repositories
- Size & quality measurement
- Benchmarking
- Metaprogramming
Mining software repos

- Software repositories contain data on software evolution
  - versions, releases, bugfixes, documentation, discussion

- Data analysis is used to uncover
  - interesting information
  - actionable information

- E.g.
  - defect prediction
  - analysis of client satisfaction

Software metrics

- Size
  - number of files? of lines of code? of languages?

- Complexity
  - number of files per artefact? lines per file?

- Performance
  - speed? bottlenecks? algorithmic complexity?
A practical model for measuring maintainability
Heitlager, Kuipers, Visser in QUATIC 2007, IEEE Press

a. Aggregate measurements into “Quality Profiles”
b. Map measurements and quality profiles to ratings for system properties
c. Map ratings for system properties to ratings for ISO/IEC 9126 quality characteristics
d. Map to overall rating of technical quality
Metaprogramming is EASY

- **Extract**
  - Fast context-free general top-down parsing
  - Pattern matching & generic traversal

- **Analyze**
  - Relational queries and comprehensions
  - Backtracking, fixed point computation, ...

- **SYnthesize**
  - String templates
  - Concrete syntax
  - Interactive visualization generator
Metaprogramming is EASY

• **Extract**
  - Fast context-free general top-down parsing
  - Pattern matching & generic traversal

• **Analyze**
  - Relational queries and comprehensions
  - Backtracking, fixed point computation, ...

• **SYnthesize**
  - String templates
  - Concrete syntax
  - Interactive visualization generator
Stay tuned!
Bus lines

© koblenz.de; fair use claimed
Bus lines


start syntax System = Line+;
syntax Line = Num ":" {Id ","}+ "." ;
layout WS = [\t\n\r]* !>> [\t\n\r];
lexical Id = [A-Za-z][A-Za-z-züäöß\- ]+[A-Za-z-z] !>> [A-Za-z-z];
lexical Num = [0-9]+ !>> [0-9];

rel[Id,Id] extractGraph(loc source) = {<from,to> | Line b := parse(#start[System],source), (Line)`<Num _>: <{Id","}* _>, <Id from>, <Id to>, <{Id","}* _>.` := b};

bool umsteigen(rel[Id,Id] sys, Id hs) = size(sys[hs]) > 1;

void synthesizeDotGraph(loc source, loc target) {
  rel[Id from,Id to] conn = extractGraph(source);
  writeFile(target,
    "digraph Metro { node [shape=box]
    '<for (<from, to> <- conn) {>
      "<from>" -> "<to>"
    '</for (st <- conn<from>, umsteigen(conn, st)){>
      "<st>" [shape=ellipse]
    '</for (st <- conn<from>, umsteigen(conn, st))}>
    '})\n  ;
}
Alan Turing

- Alan Turing (1912–1954)
- Programming pioneer
- Defined “algorithm”
- Defined “computability”
- Defined “intelligent”
- Cracked Enigma with Bombe
- ...
- RTFM
Grammars...
To summarise

- Software is complex
- Metaprogramming helps
- Rascal
  - no magic, no boilerplate
  - works across languages
- Rapid prototyping
- Rapid tool development
- ⇒ http://rascal-mpl.org