Software Improvement Group

SLE 2008

# A Case Study in Grammar Engineering

Tiago Alves & Joost Visser

September 30, 2008

Arent Janszoon Ernststraat 595-H
NL-1082 LD Amsterdam
info@sig.nl
www.sig.nl

# The Software Improvement Group


## Characterization

- Based in Amsterdam, The Netherlands
- Spin-off from the Centre of Mathematics and Information Technology (CWI)
- Fact-based IT consulting

## Services

- Software Risk Assessment
  - Exhaustive research of software quality and risks
  - Answers specific research questions
  - One time execution
- Software Monitor
  - Automated quality measurements executed frequently (daily-weekly)
  - Information presented in a web portal
- DocGen
  - Automated generation of technical documentation

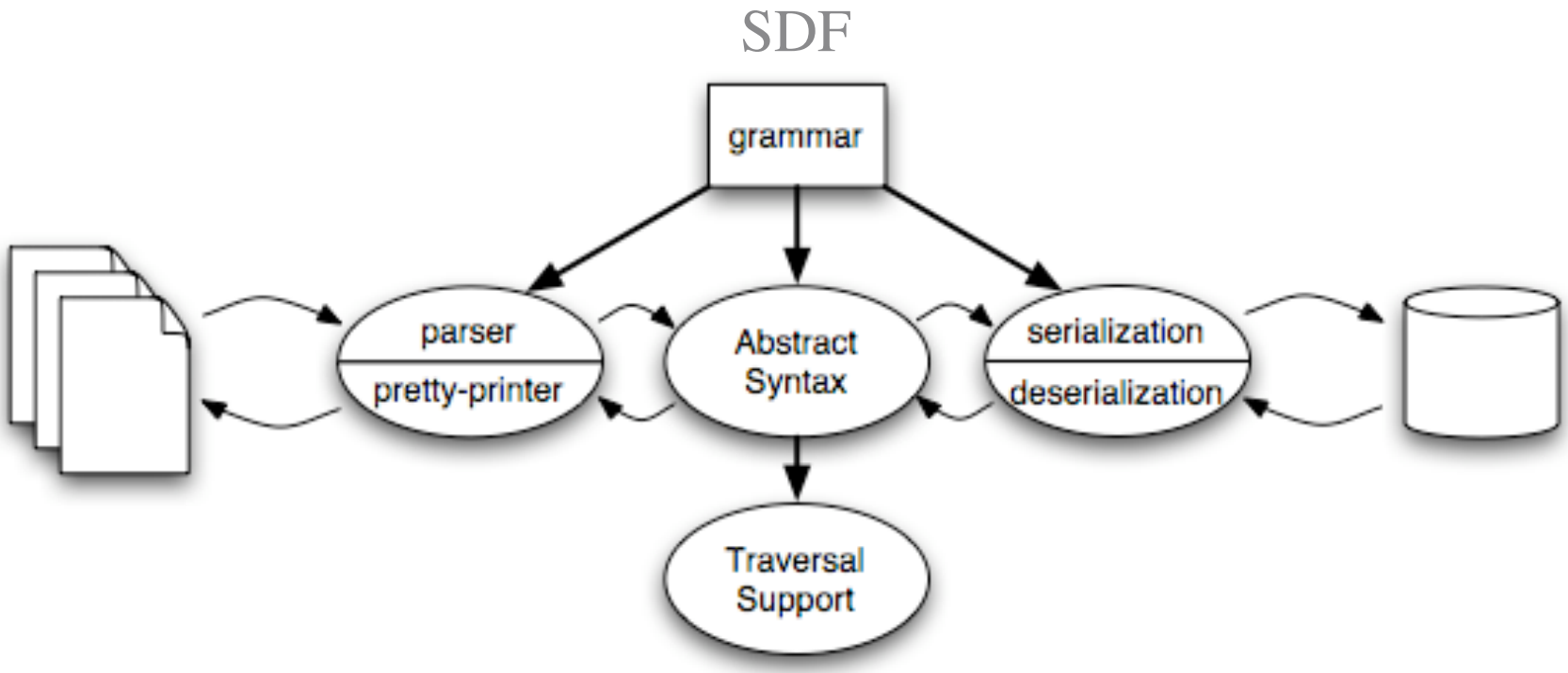
# Overview

Software Improvement Group

- **Problem**
  - Grammars are hard to write
- **Motivation:**
  - Effective and efficient way to develop and maintain a grammar
- **How:**
  - Apply solid software engineering techniques to grammars
- **Methodology:**
  - Versioning
  - Metrics
  - Visualization
  - Testing
- **Case study:**
  - Recovering of a VDM-SL grammar from the ISO document

# Tool-based grammar engineering

Software Improvement Group



# Approach for grammar engineering


- **Grammar evolution (CVS)**
  - Recovery
  - Error correction
  - Extension or restriction
  - Refactoring
- **Grammar metrics**
  - Size, complexity, and structure metrics
  - Halstead metrics
  - Disambiguation metrics (SDF specific)
- **Grammar testing**
  - Unit, integration, and regression testing
- **Coverage metrics**
  - (Production) rule coverage
  - Non-terminal coverage (equivalent to BNF rule coverage)

# Approach for grammar engineering
# Grammar evolution deliverables


1. **Initial grammar (revision 1)**
   - Recovery (context-free syntax + lexical syntax manual transcription)
   - Error correction (visual detection of top and bottom non-terminals)
   - + Addition of SDF specific constructs
   - + Modularized grammar
2. **Disambiguated grammar (revision 32)**
   - Extension or restriction (associativity attributes, priorities, reject productions, look-ahead restrictions)
   - + Injections removal
3. **Refactored grammar (revision 48)**
   - Addition of annotations for AST building
   - Injections removal

**All deliverables can be used for parsing!!!**

# Approach for grammar engineering
# Grammar metrics

Software Improvement Group

| Size and complexity metrics | |
|---|---|
| TERM | Number of terminals |
| VAR | Number of non-terminals |
| MCC | McCabe's cyclomatic complexity |
| AVS-P | Average RHS per production |
| AVS-N | Average RHS per non-terminal |
| **Structure metrics** | |
| TIMP | Tree impurity (%) |
| CLEV | Normalized count of levels (%) |
| NSLEV | Number of non-singleton levels |
| DEP | Size of largest level |
| HEI | Maximum height |

| Halstead metrics | |
|---|---|
| n1 | Number of distinct operators |
| n2 | Number of distinct operands |
| N1 | Total number of operators |
| N2 | Total number of operands |
| n | Program vocabulary |
| N | Program length |
| V | Program volume |
| D | Program difficulty |
| E | Program effort |
| L | Program level |
| T | Program time |

| Disambiguation metrics (SDF specific) | | | |
|---|---|---|---|
| FRST | Number of follow restrictions | ASSOC | Number of associativity attributes |
| REJP | Number of reject productions | UPP | Number of unique productions in priorities |

# Approach for grammar engineering
# Grammar testing

Software Improvement Group

- **Unit testing (define start non-terminal)**
  - Positive and negative
    - Parsing (weaker)
    - AST (stronger)
- **Integration testing (full test suite)**
  - Used to find errors in the initial grammar
  - Used for regression testing

# Approach for grammar engineering
# Coverage metrics

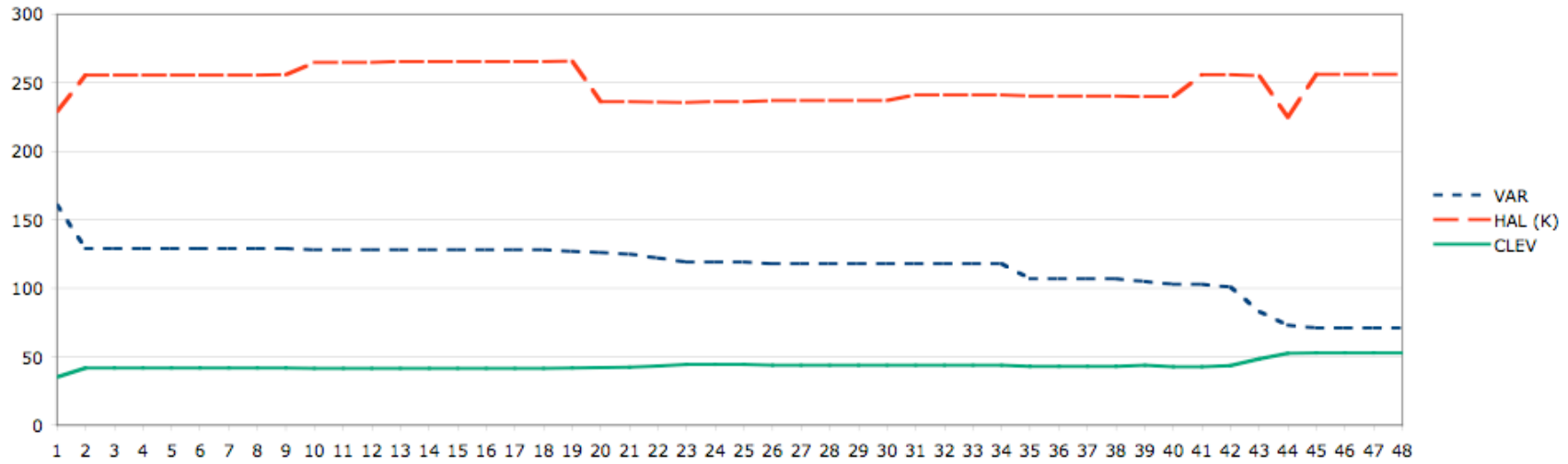
Software Improvement Group

- **Rule Coverage (RC)**
- **Non-terminal Coverage (NC)**

| EBNF | SDF |
|---|---|
| type = “int”<br>| “(“, type, “)”<br>; | “int” -> Type<br>“(“ Type “)” -> Type |

# Grammar evolution

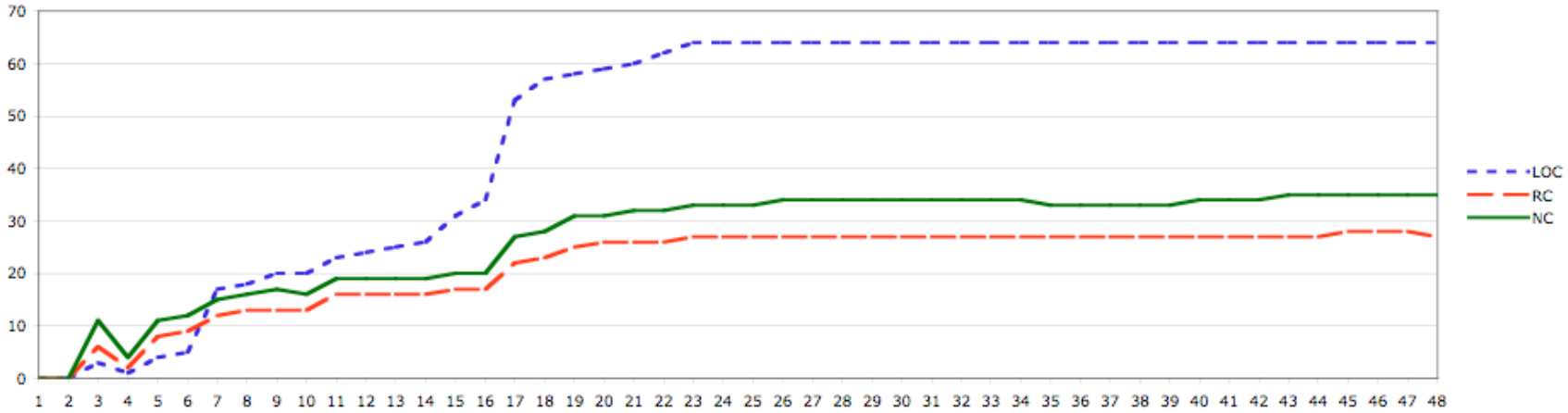
Software Improvement Group




| Version | TERM | VAR | MCC | AVS-N | AVS-P | HAL-E | TIMP | CLEV | NSLEV | DEP | HEI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial | 138 | 161 | 234 | 4.4 | 2.3 | 55.4 | 1% | 34.9 | 4 | 69 | 16 |
| Disambiguated | 138 | 118 | 232 | 6.4 | 2.8 | 61.1 | 1.5% | 43.9 | 4 | 39 | 16 |
| Refactored | 138 | 71 | 232 | 10.4 | 3.3 | 68.2 | 3% | 52.6 | 3 | 27 | 14 |

# Grammar evolution II
# Disambiguation information

Software Improvement Group




| Test set | LOC | RC | NC |
|---|---|---|---|
| Specification of the MAA standard (Graeme Parkin) | 269 | 19% | 30% |
| Abstract Data Types (Matthew Suderman and Rick Sutcliffe) | 1287 | 37% | 53% |
| A crosswords assistant (Yves Ledru) | 144 | 28% | 43% |
| Modeling of Realms (Peter Gorm Larsen) | 380 | 26% | 38% |
| Exercises formal methods course University of Minho (Tiago Alves) | 500 | 35% | 48% |
| **Total** | **2580** | **50%** | **70%** |

# Grammars comparison

Software Improvement Group

| Grammar | TERM | VAR | MCC | AVS-N | AVS-P | HAL | TIMP | CLEV | NSLEV | DEP | HEI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Fortran 77 | 21 | 16 | 32 | 8.8 | 3.4 | **26** | 11.7 | 95.0 | 1 | 2 | 7 |
| *ISO C* | 86 | 65 | 149 | 5.9 | 5.9 | **51** | 64.1 | 33.8 | 3 | 38 | 13 |
| *Java v1.1* | 100 | 149 | 213 | 4.1 | 4.1 | **95** | 32.7 | 59.7 | 4 | 33 | 23 |
| AT&T SDL | 83 | 91 | 170 | 5.0 | 2.6 | **138** | 1.7 | 84.8 | 2 | 13 | 15 |
| *ISO C++* | 116 | 141 | 368 | 6.1 | 6.1 | **173** | 85.8 | 14.9 | 1 | 121 | 4 |
| *ECMA Standard C#* | 138 | 145 | 466 | 4.7 | 4.7 | **228** | 29.7 | 64.9 | 5 | 44 | 28 |
| ISO VDM-SL | 138 | 71 | 232 | 10.4 | 3.3 | **256** | 3.0 | 52.6 | 3 | 27 | 14 |
| VS Cobol II | 333 | 493 | 739 | 3.2 | 1.9 | **306** | 0.24 | 94.4 | 3 | 20 | 27 |
| VS Cobol II (*alt*) | 364 | 185 | 1158 | 10.4 | 8.3 | **678** | 1.18 | 82.6 | 5 | 21 | 15 |
| PL/SQL | 440 | 499 | 888 | 4.5 | 2.1 | **715** | 0.3 | 87.4 | 2 | 38 | 29 |

# Contributions


- **Documented methodology for iterative grammar development**
  - Version control
  - Grammar metrics
  - Unit testing
  - Coverage
- **Demonstrated how to make grammar development a controlled process**
- **Presented case study of a real-world grammar: VDM-SL**
- **Extension of Power & Malloy:**
  - Extensive use of unit tests + coverage analysis
  - Metric reference data for several languages

# Future work

Software Improvement Group

- Testing
  - Context-dependent branch coverage - Lämmel (FASE'01)
  - Automatic test generation
- VDM-SL grammar
  - Add VDM++ extensions
  - Java front-end

Software Improvement Group

# Thank you!