Parser Development for Overture Tools

http://www.overturetool.org

Marcel Verhoef, 25 May 2008, Fourth Overture Workshop
Abo Akademi University – Turku - Finland

Iteration One (2004)

- Pieter van der Spek (MSc thesis project, TU Delft)
  - Build parser and simple pretty printer
  - Experiments on improved error support in parser generator (published as ACM Sigplan Notices)
  - Delivered as Eclipse plug-in

  - *Limited (no) XML support*
  - *Direct manipulation of concrete syntax tree*

Iteration Two (2005)

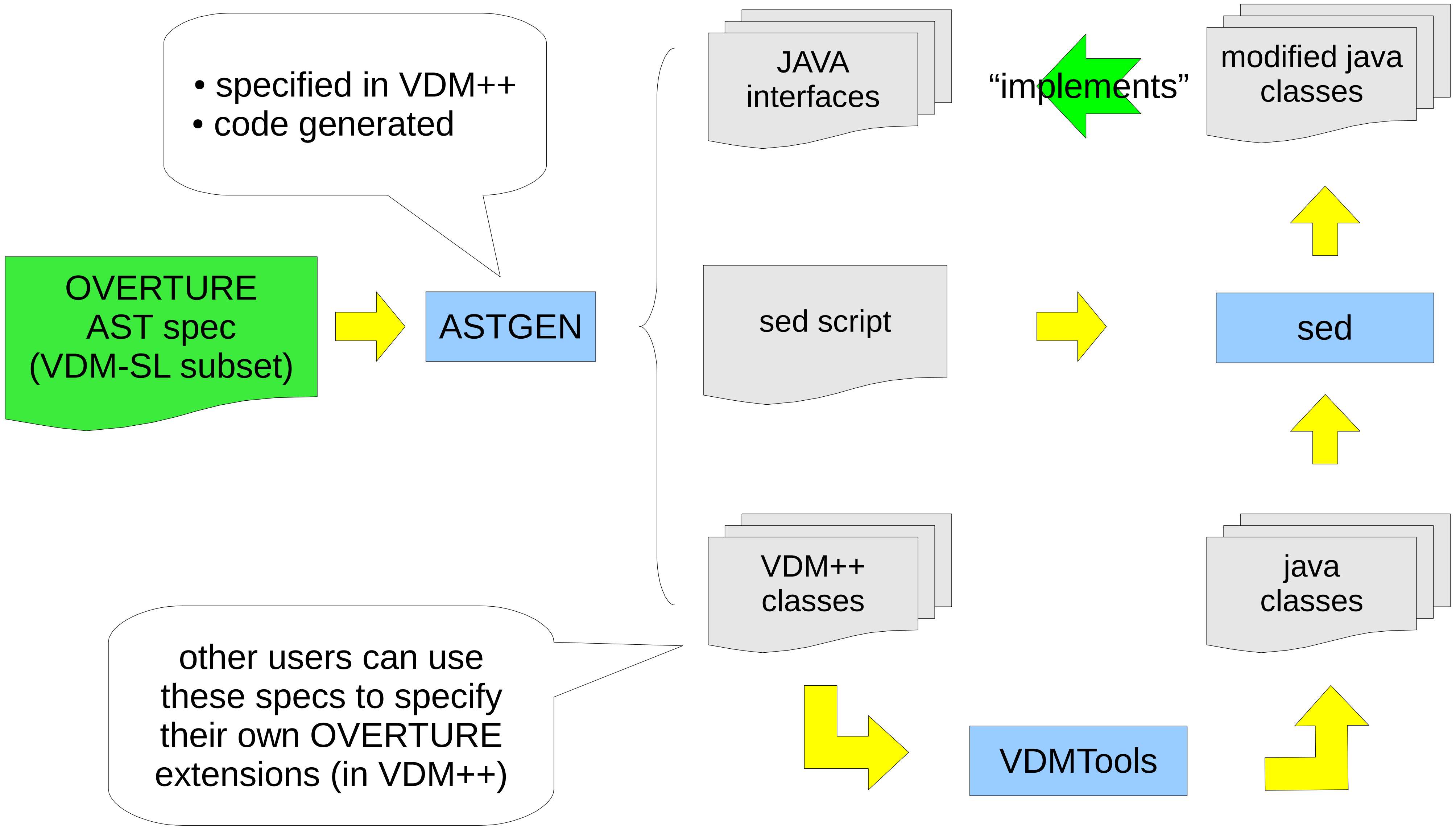
- Jacob Porsborg Nielsen & Jens Kielsgaard Hansen (MSc thesis project, TU Denmark)
  - Re-implemented parser using ANTLR
  - abstract syntax with appropriate Java interfaces
  - XML support for reading / writing AST instances
  - Experimented with Eclipse architecture, many useful suggestions and prototype plug-ins

  - *Hand-coded AST implemention – very error prone*
  - *Many errors in parser implemention*

Meanwhile in 2006

- Address problem of AST maintenance
- Executive decision: we need a ***robust*** solution...
- ... to support language experiments
- Back To Basics: how can we re-use VDMTools parser and know-how?
- How good are the open source ***jflex*** and ***byaccj*** tools?
- Automation is key
- “eat our own dogfood”

Automatic AST generation



AST specification and code examples (1)

```
%directory "d:\projects\ShowTraceParser";
%package org.overturetool.tracefile.ast;

TraceFile ::
  Trace : seq of TraceEvent;

TraceEvent =
  ThreadSwapIn | ....
  DeployObj;

--
-- THREADS
--

ThreadSwapIn ::
  -- id of the thread
  id : nat
  -- id of the object
  objref : [nat]
  -- name of the class
  clnm : [seq of char]
  -- id of the CPU
  cpunm : nat
  -- swap-in overhead (time units)
  overhead : nat
  -- observation time
  time  : nat;
```

(THE JAVA INTERFACE)

```
package org.overturetool.tracefile.ast.itf;

import jp.co.csk.vdm.toolbox.VDM.*;

public abstract interface IThreadSwapIn extends ITraceEvent
{
        abstract Integer getId() throws CGException;
        abstract Integer getObjref() throws CGException;
        abstract Boolean hasObjref() throws CGException;
        abstract String getClnm() throws CGException;
        abstract Boolean hasClnm() throws CGException;
        abstract Integer getCpunm() throws CGException;
        abstract Integer getOverhead() throws CGException;
        abstract Integer getTime() throws CGException;
}
```

AST specification and code examples (2)

```
%directory "d:\projects\ShowTraceParser";
%package org.overturetool.tracefile.ast;

TraceFile ::
  Trace : seq of TraceEvent;

TraceEvent =
  ThreadSwapIn | ....
  DeployObj;

--
-- THREADS
--

ThreadSwapIn ::
  -- id of the thread
  id : nat
  -- id of the object
  objref : [nat]
  -- name of the class
  clnm : [seq of char]
  -- id of the CPU
  cpunm : nat
  -- swap-in overhead (time units)
  overhead : nat
  -- observation time
  time  : nat;
```

(VDM++ CLASSES - INTERFACE DEFINITION)

```
class IOmlThreadSwapIn
 is subclass of IOmlTraceEvent

operations
  public getId: () ==> nat
  getId() == is subclass responsibility;

operations
  public getObjref: () ==> nat
  getObjref() == is subclass responsibility;

  public hasObjref: () ==> bool
  hasObjref () == is subclass responsibility;

operations
  public getClnm: () ==> seq of char
  getClnm() == is subclass responsibility;

  public hasClnm: () ==> bool
  hasClnm () == is subclass responsibility;
...
end IOmlThreadSwapIn
```

AST specification and code examples (3)

```
%directory "d:\projects\ShowTraceParser";
%package org.overturetool.tracefile.ast;

TraceFile ::
  Trace : seq of TraceEvent;

TraceEvent =
  ThreadSwapIn | ....
  DeployObj;

--
-- THREADS
--

ThreadSwapIn ::
  -- id of the thread
  id : nat
  -- id of the object
  objref : [nat]
  -- name of the class
  clnm : [seq of char]
  -- id of the CPU
  cpunm : nat
  -- swap-in overhead (time units)
  overhead : nat
  -- observation time
  time  : nat;
```

(VDM++ CLASSES - IMPLEMENTATION)

```
class OmlThreadSwapIn
  is subclass of IomlThreadSwapIn

operations
  public identity: () ==> seq of char
  identity () == return "ThreadSwapIn";

  public accept: IOmlVisitor ==> ()
  accept (pVisitor) == pVisitor.visitThreadSwapIn(self)

instance variables
  private ivId : [nat] := nil

operations
  public getId: () ==> nat
  getId() == return ivId;

  public setId: nat ==> ()
  setId(parg) == ivId := parg
...
end OmlThreadSwapIn
```

The Proof of the Pudding ...

- Applied this approach to implement Overture / VICE Tracefile viewer
- Implemented parser using JFLEX and BYACCJ
- Some “extra” bonuses
  - implemented standard “visitor pattern” support
  - implemented AST attribution “NodeProperty”
  - default visitors for writing VDM++ and VDM-SL values
- Many changes occurred during development
- Turn-around time new parser: just a few hours
- JFLEX / BYACC seem quite robust and FAST
- IMHO: this is the way to go!

... is in the eating.

- Unfortunately BYACCJ needed to be patched
  - VDM++ is a very (very) large language
  - parser table initialization exceeds Java 64kb code limit
  - we fixed this problem (initialization split over multiple operations)
  - byaccj maintainers did not find the change “useful” (?!)
- ASTGEN was developed “as we go along”
  - certainly not fit for public release, ad-hoc tool
  - lacking support (no manuals, no documentation)
- Generated Java code from VDMTools needs patching
  - dependence on extra set of (standard Unix) tools (cygwin)

Iteration 3 (2006, 2007)

- parser released through Overture web-site (zip file)
  - as a pre-compiled Java binary library / executable
  - partial source release of key implementation files
    - overture.ast → description of the AST
    - parser.y → bjaccy parser source file
    - scanner.l → jflex scanner source file
    - Java interfaces for the parser and AST implementation
    - VDM++ sources of the AST
- developers can
  - specify analysis tools in VDM++ using the VDM++ AST classes
  - implement their own tools on top of Java parser library

The Good, The Bad and The Ugly

- **GOOD**
  - very robust and stable parser (2000 test cases), quite acceptable performance
  - seamless integration into Java and VDM++ environments
  - language changes are easy to specify
  - visitor support on AST
- **BAD**
  - no position information available in the AST
  - no Eclipse plug-in, no XML transformation available
- **UGLY**
  - build process is not for the faint of heart, steep learning curve
  - depends on many external tools, involves intricate manual steps
  - cannot yet be built directly from (open-)source repository

Iteration 4 (2008)

- developers have actually proposed language extensions
  - Thomas Christensen (2006, MSc project, Aarhus, Denmark)
    - *generic class type (as in Ada)*
    - *typeless (truly polymorphic) explicit functions*
  - Marcel Verhoef (2007, PhD project, Nijmegen, The Netherlands)
    - *added “system”, “cycles” and “sporadic” constructs*
    - *extended existing “duration” construct (VICE)*
  - Adriana Sucena (2008, MSc project, Minho, Portugal)
    - *added “traces” definition block for test case generation*
- “edit-compile-debug” cycle remains responsibility of Overture core
- implementation of position information is now available

Iteration 4 (2008)

- new parser is released in May 2008 on Overture web-site
- main new feature is availability of accurate position information
- required significant update to ASTGEN and parser (BYACC source)
- each AST node provides *getLine()* and *getColumn()* operations

- availability of Overture core members to update parser is low
- to break this critical resource dependency:
  - byaccj binary (+ source patch) is made available on-line
  - ASTGEN binary + short “rough guide” is made available on-line

Parsing “3 + true”

```
new OmlDocument("expr3.vpp",
  new OmlBinaryExpression(
    new OmlSymbolicLiteralExpression(new OmlNumericLiteral(3,1,1),1,1),
    new OmlBinaryOperator(24,1,3),
    new OmlSymbolicLiteralExpression(new OmlBooleanLiteral(true,1,5),1,5),
    1,
    3
  ),
  [
    new OmlLexem(1,1,435,"3",0),
    new OmlLexem(1,3,43,"+",0),
    new OmlLexem(1,5,430,"true",1)
  ]
)
```

Plans for 2008 / 2009

- planned support activities:
  - write paper and workshop focused on tool development
  - rethink, rewrite and release ASTGEN support tool
  - deal with reported bugs (if any) and language change requests

- planned development activities:
  - add “poor mans” pretty printer for LaTeX
  - move Overture repository on SourceForge from CVS to SVN
  - enable build from repository with binary versions of support tools
  - develop Eclipse parser plug-in

LETS PREPARE FOR OVERTURE @ FM 2009 @ EINDHOVEN