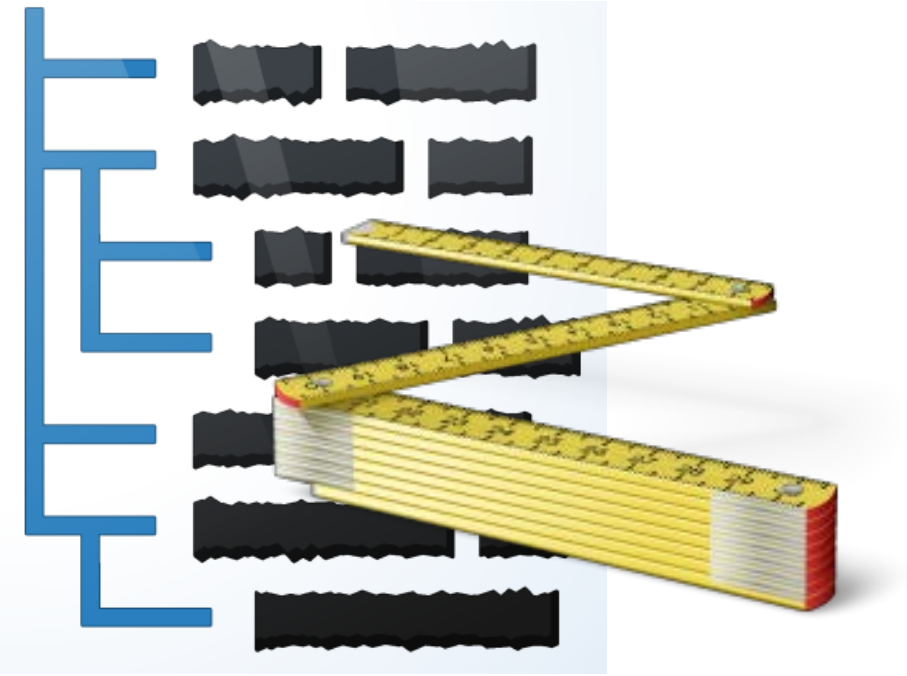
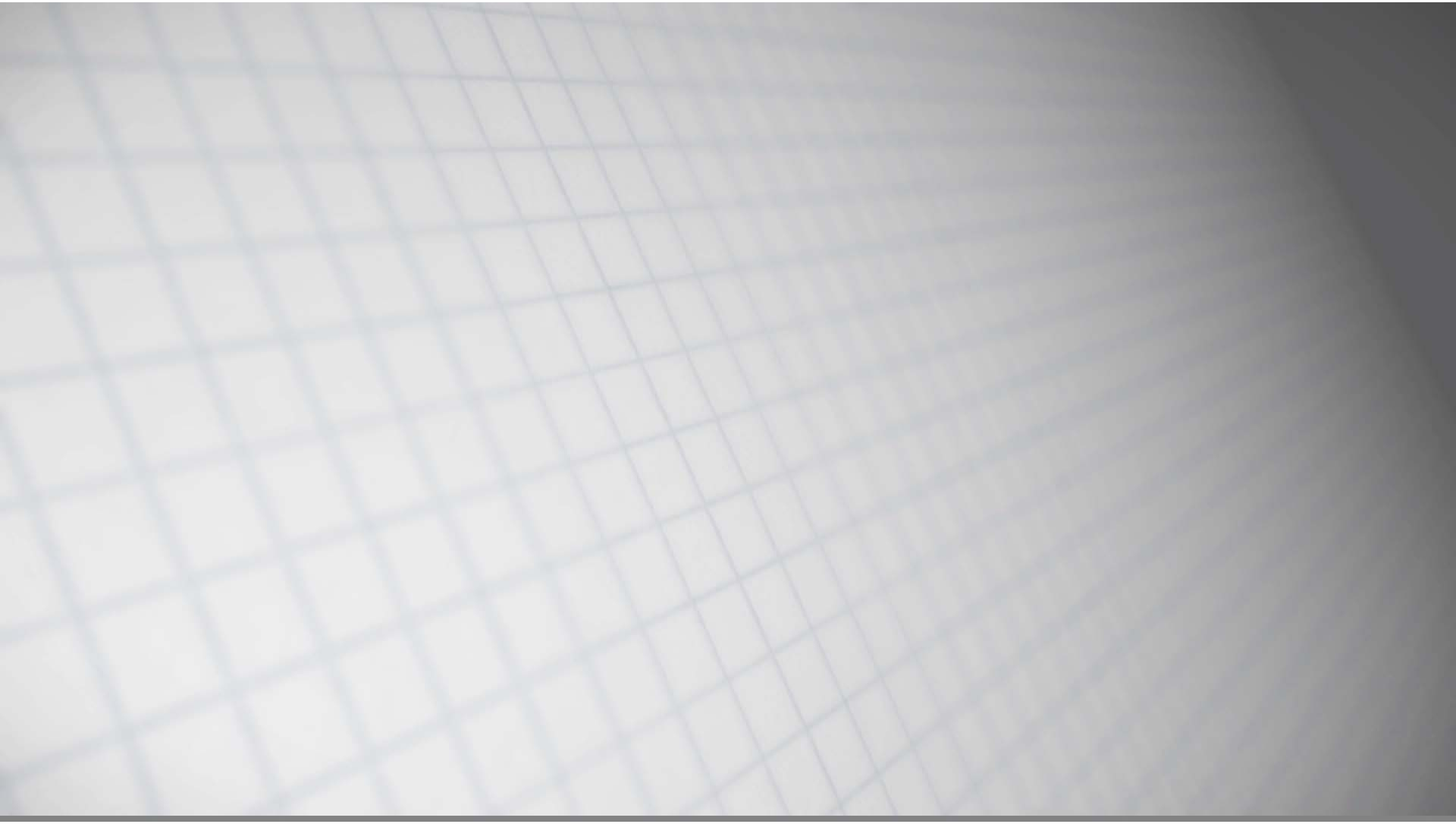
Analyzing ABL Source Code with *Proparse*



Mike Fechner, Consultingwerk Ltd.
mike.fechner@consultingwerk.de



Consultingwerk Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and **related technology**
- Located in Cologne, Germany, subsidiary in UK
- Customers in Europe, North America, Australia and South Africa
- Vendor of tools and consulting services
- 27 years of Progress experience (V5 … OE11)
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration

Sample code download

- https://github.com/mikefechner/proparse-samples
- Most sample code has no dependencies
- Some samples rely on commercial code from Consultingwerk

Agenda

- **Why Source Code Analysis**
- Proparse
- Utilities based on Proparse
- Proparse.NET
- Using Proparse from ABL
- Building and enhancing Proparse

Why Source Code Analysis?

- Quality Assurance, Linting of Code

- Refactoring, Foundation for converting code form one form to another

Linting

- Linting is the process of flagging suspicions code within a programming language
- Linting requires understanding of source code

- Extension to syntax checks
- Code may compile, but still be wrong
  - DEFINE VARIABLE without NO-UNDO
  - FIND with no NO-ERROR

Refactoring, Code conversion

- Refactoring is the process of restructuring computer code without changing its external behavior
- Refactoring requires understanding of code
- Identifying and locating relevant pieces of code
- Ignoring less relevant bits
- Provide ability to change or extract code

ABL Built in Source Code Analysis

- Compiler Output
  - Cross Reference (XREF, XML-XREF)
  - LISTING (Buffer and Transaction Scope)
  - PREPROCESS/DEBUG-LISTING
- Profiler Output
  - Tracing of executed lines of code + performance

Agenda

- Why Source Code Analysis
- **Proparse**
- Utilities based on Proparse
- Proparse.NET
- Using Proparse from ABL
- Building and enhancing Proparse

Proparse

- Proparse is a utility to return an ***abstract syntax tree*** for ABL code (AST)
- Static code analysis
- Interpreting the grammar of the ABL
- Knowledge about keywords and their valid combinations
- Should understand any piece of ABL code that compiles
- Providing a structured view on source code

Why an *abstract* syntax tree?

- ABL syntax “flexible”:
  - Formatting
  - Large number of keywords
  - Abbreviated keywords
  - Keyword order in statements
  - Uppercasing, lower casing, many keywords may be used as identifiers
  - Single / Double Quotes
  - Comments

Why use an abstract syntax tree

- Because analyzing ABL source code as a text file is hard!

That‘s why:

simple-1.p | simple-2.p

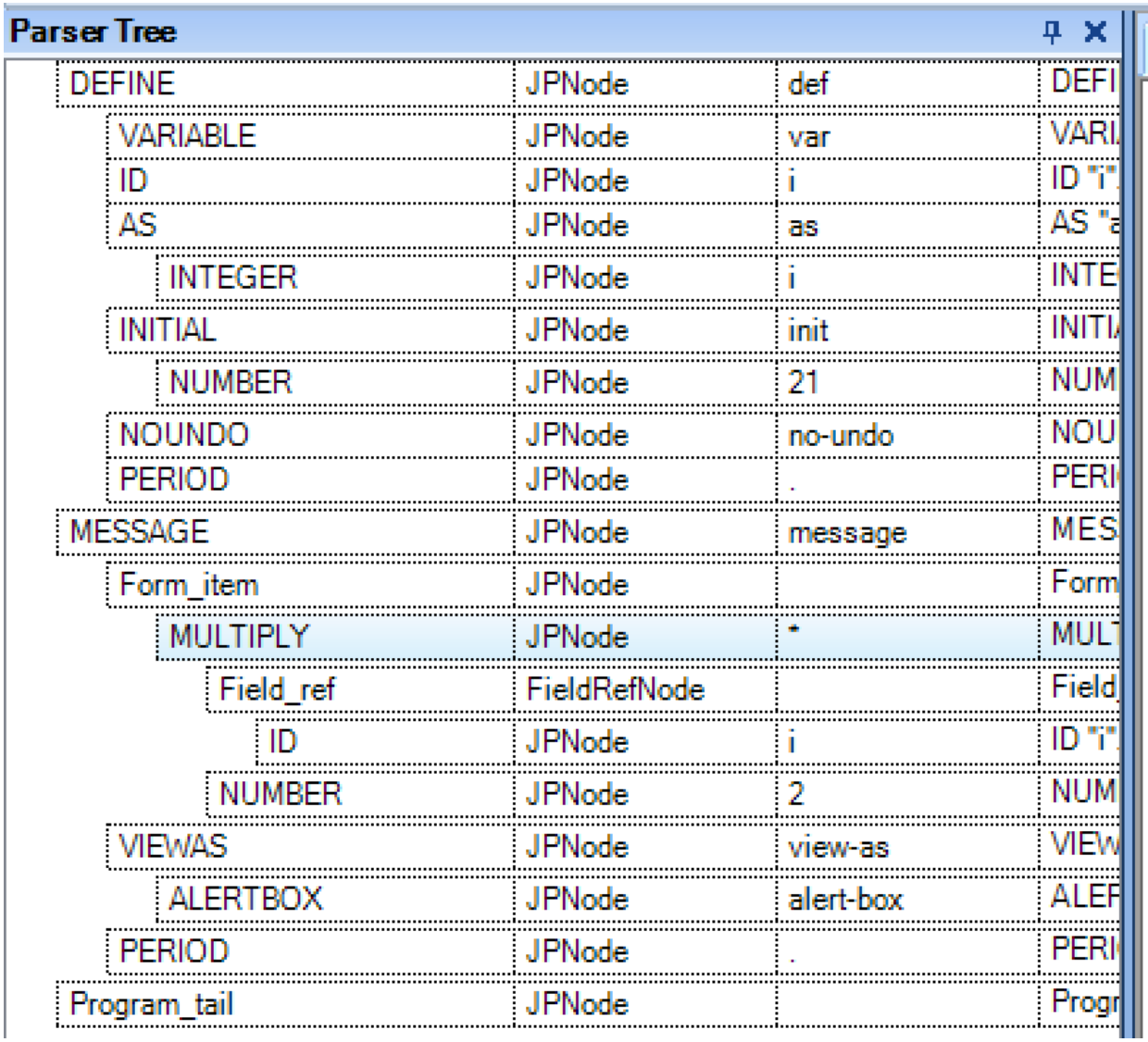
```
/*------------------------------------------------------------------------*/

/* ***************************  Definitions  ************************** */

DEFINE VARIABLE i AS INTEGER NO-UNDO INITIAL 21 .

/* ***************************  Main Block  *************************** */

MESSAGE i * 2
    VIEW-AS ALERT-BOX.
```

simple-1.p | simple-2.p

```
def var i as i init 21 no-undo. message i * 2 view-as alert-box.
```

simple-1.p | simple-2.p | simple-3.p

```
def var i as i init 21 no-undo.
message i * 2 // * 3
view-as /*alert-box*/
        alert-box.
```



simple-3.p

```
def var i as i init 21 no-undo.
message i * 2 // * 3
view-as /*alert-box*/
        alert-box.
```

Proparse

- Original Author: John Green / Joanju
- http://www.joanju.com/proparse/
- http://www.oehive.org/proparse
- Eclipse public license
- Extracts the *Abstract Syntax Tree* from a compilation unit (procedure or class)
- Is NOT a compiler, nor a Syntax Checker
  - similar requirements as the compiler to understand source code
- Based on **ANTLR**, quite an ancient version - 2.7

ANTLR

- "**An**other **T**ool for **L**anguage **R**ecognition"
- Toolkit for building language parses
- Java based
- Generated parsers are Java code
- A lot more tooling available in more recent versions of ANTLR

- If you're not maintaining Proparse, you don't need to use any of that tooling

Proparse JavaDoc

- http://www.joanju.com/analyst/javadoc/index.html
- Look for
  - org.prorefactor.core.JPNode
  - org.prorefactor.treeparser.ParseUnit

Proparse

- Multiple public repositories
  - OE Hive SVN
  - github.com/oehive/proparse
  - **github.com/consultingwerk/proparse**
  - github.com/riverside-software/proparse

- After a dormant phase a few years back, it's actively maintained again
- Support for full OpenEdge 11.7 syntax available

Agenda

- Why Source Code Analysis
- Proparse
- **Utilities based on Proparse**
- Proparse.NET
- Using Proparse from ABL
- Building and enhancing Proparse

Utilities based on Proparse

- Prolint
- SonarSource plug-ins for OpenEdge
- SmartComponent Library

Prolint

- http://www.oehive.org/prolint
- Tool for automated source code review of Progress 4GL code
- Reads one or more source files and examines them for bad programming practice
- Mostly procedural syntax support
- Active times around V9 and V10 …

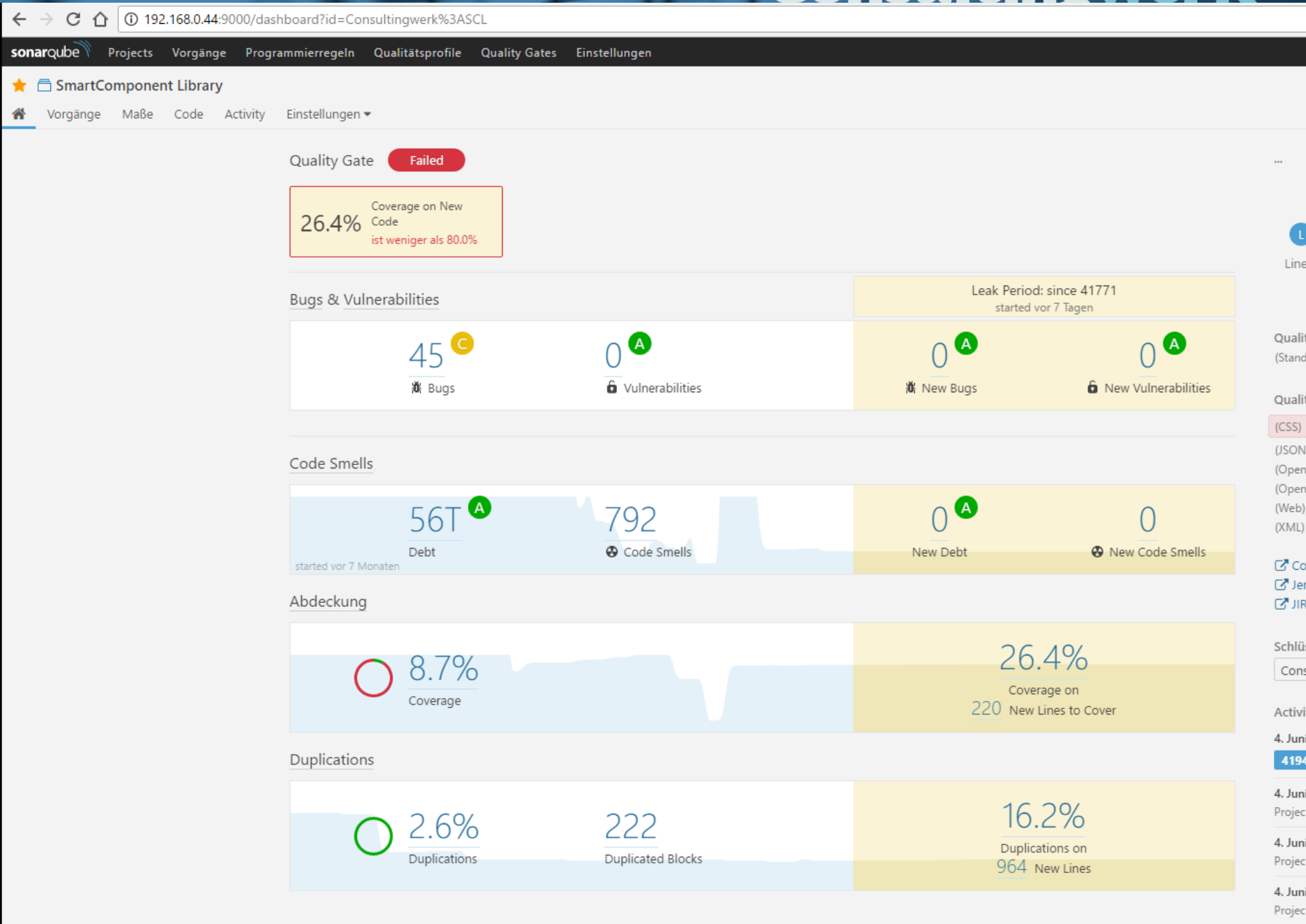
SonarQube by SonarSource

- Commonly used open source Lint Tool
- Support for various programming languages via plug-ins, Java, JavaScript, C#, HTML, XML, …
- OpenEdge plug-in developed by Riverside Software (Gilles Querret)
  - engine open source
  - rules commercial
- Available since 2016, permanently new features added

SonarQube by SonarSource

- Locates problems or potential bugs
- Violation of coding-standards
- Code duplication detection
- Unit-Test coverage

- Web-Dashboard
- CLI Utility (HTML or XML Reports)
- Eclipse Integration


192.168.0.44:9000/dashboard?id=Consultingwerk%3ASCL
sonarqube
Projects
Vorgänge
Programmierregeln
Qualitätsprofile
Quality Gates
Einstellungen
SmartComponent Library
Vorgänge
Maße
Code
Activity
Einstellungen
Quality Gate
Failed
26.4%
Coverage on New Code
ist weniger als 80.0%
Bugs & Vulnerabilities
Leak Period: since 41771
started vor 7 Tagen
45
C
Bugs
0
A
Vulnerabilities
0
A
New Bugs
0
A
New Vulnerabilities
Code Smells
56T
A
Debt
started vor 7 Monaten
792
Code Smells
0
A
New Debt
0
New Code Smells
Abdeckung
8.7%
Coverage
26.4%
Coverage on
220
New Lines to Cover
Duplications
2.6%
Duplications
222
Duplicated Blocks
16.2%
Duplications on
964
New Lines

SonarLint for Eclipse Demo

- Integration into Progress Developer Studio

```
/*------------------------------------------------------------------------
 Purpose:
 Notes:
------------------------------------------------------------------------*/
@VisualDesigner.
METHOD PRIVATE VOID button1_Click( INPUT sender AS System.Object, INPUT e AS

    DEFINE VARIABLE cTest AS CHARACTER NO-UNDO INIT "abc" .

    cTest = "def" .

    REPEAT:

    END.

END METHOD.

METHOD PRIVATE VOID InitializeComponent(  ):
```

SmartComponent Library based Tools

- Commercial ABL developer framework by Consultingwerk
- Business Entity Designer round trip development is based on Proparse
- Legacy code modernization utilities uses Proparse for analyzing legacy code

Demo

- Business Entity Designer round trip Development

Agenda

- Why Source Code Analysis
- Proparse
- Utilities based on Proparse
- **Proparse.NET**
- Using Proparse from ABL
- Building and enhancing Proparse

Proparse.NET

- Proparse is written in Java
- ABL has no built in bridge to Java
- ABL has a bridge to .NET
- .NET saves the day – actually, the Mono Project

IKVM.NET

- Part of the Mono Project – Open Source implementation of the .NET framework
- Java VM implemented in .NET
- Java Byte Code embedded in .NET Assembly (.dll file)
- Allows execution of Java code from .NET applications
- Since ABL can use (most) .NET Assemblies, ABL can use Proparse via IKVM.NET

Integrating Proparse.NET into OpenEdge

- Get familiar with the *GUI for .NET Programming* guide!!!
- -assemblies startup parameter
- assemblies.xml file
- Proparse.NET Assemblies available at https://github.com/consultingwerk/proparse

- Think of –assemblies like a PROPATH definition for .NET classes

assemblies.xml

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<references>
   <assembly name="IKVM.OpenJDK.Core,
   Version=7.2.4630.5, Culture=neutral,
   PublicKeyToken=13235d27fcbfff58"/>
   <assembly name="IKVM.Runtime, Version=7.2.4630.5,
   Culture=neutral,
   PublicKeyToken=13235d27fcbfff58"/>
   <assembly name="proparse.net, Version=4.0.1.1166,
   Culture=neutral,
   PublicKeyToken=cda1b098b1034b24"/>
</references>
```

Codepage used by Proparse.NET

- Option in **prowin32.exe.config** / **prowin.exe.config**
- http://www.oehive.org/proparse#comment-2118
- Add **ikvm:file.encoding** property to the .config file
- File is dependent on the OpenEdge version – ***don't break it! That file is important!***
- Refer to the .NET framework documentation for details

```
<appSettings>
<add key="ikvm:file.encoding" value="ibm850" />
</appSettings>
</configuration>
```

Agenda

- Why Source Code Analysis
- Proparse
- Utilities based on Proparse
- Proparse.NET
- **Using Proparse from ABL**
- Building and enhancing Proparse

Using Proparse from the ABL

1. Setting up environment
2. Invoking the parser
3. Iterating the AST
4. Understanding your code

Setting up the environment for Proparse

- Similar requirements as an ABL compile time session
- PROPATH
- Database connections and schema
- SESSION settings like OPSYS, PROVERSION and WINDOW-SYSTEM that might be used in &IF

Initializing the Proparse environment

```
USING com.joanju.proparse.NodeTypes FROM ASSEMBLY .
USING org.prorefactor.core.JPNode   FROM ASSEMBLY .
USING org.prorefactor.nodetypes.*   FROM ASSEMBLY .
USING org.prorefactor.treeparser.*  FROM ASSEMBLY .
USING Progress.Lang.*               FROM PROPATH .

DEFINE VARIABLE proparseEnv    AS com.joanju.proparse.Environment
DEFINE VARIABLE proparseSchema AS org.prorefactor.core.schema.Schema
DEFINE VARIABLE prsession      AS org.prorefactor.refactor.RefactorSession
```

```
RUN ExportSessionSettings .
RUN ExportDatabaseSchema .
RUN InitializeParserSession .
```

Session Settings

```
/**
 * Purpose: Exports the ABL Session Settings
 * Notes:
 */
PROCEDURE ExportSessionSettings:

    proparseEnv = com.joanju.proparse.Environment:instance().

    /* Export ABL Session settings to Proparse */
    proparseEnv:configSet ("batch-mode":U, STRING(SESSION:BATCH-MODE, "true/false":U)).
    proparseEnv:configSet ("opsys":U, OPSYS).
    proparseEnv:configSet ("propath":U, PROPATH).
    proparseEnv:configSet ("proversion":U, PROVERSION).
    proparseEnv:configSet ("window-system":U, SESSION:WINDOW-SYSTEM).

END PROCEDURE .
```

```
/**
 * Purpose: Exports the Database Schema to Proparse
 * Notes:   Will only export the database schema to proparse when there are new
 *          databases connected or new aliases defined
 */
PROCEDURE ExportDatabaseSchema:

    DEFINE VARIABLE iAlias                  AS INTEGER        NO-UNDO .
    DEFINE VARIABLE schemaDumpFile          AS CHARACTER      NO-UNDO .

    proparseSchema = org.prorefactor.core.schema.Schema:getInstance().

    proparseSchema:clear().

    IF NUM-DBS > 0 THEN DO:
        schemaDumpFile = Consultingwerk.Util.FileHelper:GetTempFileName() .

        RUN Consultingwerk/Studio/Proparse/schemadump1.p (schemaDumpFile).

        proparseSchema:loadSchema(schemaDumpFile).

        DO iAlias = 1 TO NUM-ALIASES:
            proparseSchema:aliasCreate (ALIAS (iAlias), LDBNAME (ALIAS (iAlias))) .
        END.
    END.

    FINALLY:
        OS-DELETE VALUE (schemaDumpFile) .
    END FINALLY.

END PROCEDURE .
```

Initialize the Proparse Session

```
/**
 * Purpose: Initializes the Propase Session
 * Notes:
 */
PROCEDURE InitializeParserSession:

    org.prorefactor.refactor.RefactorSession:invalidateCurrentSettings () .

    prsession = org.prorefactor.refactor.RefactorSession:getInstance().
    prsession:setContextDirName (SESSION:TEMP-DIRECTORY) .

END PROCEDURE.
```

Invoke the Parser ☺

```
DEFINE VARIABLE javafile        AS java.io.File
DEFINE VARIABLE pu              AS org.prorefactor.treeparser.ParseUnit



/* Parse ABL source code */
javafile = NEW java.io.File ("simple-3.p").

IF (NOT javafile:exists()) THEN
    UNDO, THROW NEW AppError (SUBSTITUTE ("Could not find file: &1.",
                                          "simple-3.p"), 0) .

pu = NEW ParseUnit(javafile).
/* Invoke the actual parser */
pu:treeParser01().

DELETE OBJECT javafile .
```

Walk the Parse Unit

- Proparse represents ABL source as a tree
- Single root
- Every node may have children and siblings – depending on allowed syntax
- Class: org.prorefactor.core.JPNode
- http://www.joanju.com/analyst/javadoc/index.html?org/prorefactor/core/JPNode.html

Walk the Parse Unit

- Starting from **pu:getTopNode() // Program_Root**
- Process that JPNode instance
- Start from **:firstChild()**, iterate while :**nextSibling()** is valid

```
PROCEDURE ProcessAst:

    DEFINE INPUT PARAMETER poNode    AS JPNode  NO-UNDO .

    DEFINE VARIABLE oChild    AS JPNode    NO-UNDO .

    ASSIGN oChild = poNode:firstChild () .

    DO WHILE VALID-OBJECT (oChild):
        RUN ProcessAst (oChild) .

        oChild = oChild:nextSibling () .
    END.

END PROCEDURE.
```

JPNode properties (Java style)

- **getType()** – the actual type of the node, representing a keyword, block structure or identifier NodeTypes:getTypeName(oNode:getType())
- **getText()** – the node's piece of ABL source code
- **getColumn()**, **getLine()**, **getFileName()**
- **firstChild()**, **nextSibling()** – similar to the ABL widget trees

Demo

- Parsing the simple-3.p
- Review recursive loop

JPNode child types

- Some JPNode's provide very specific additional information, which is implemented through child types of JPNode
- BlockNode, FieldRefNode, RecordNameNode, …
- Requires CAST from JPNode reference
- Provides direct properties and references to additional types

Demo

- Parsing the customer-tt.p
- Review recursive loop including RecordNameNode handling

Demo

- Parsing temp-table-sample.p
- Extract TEMP-TABLE fields from ABL source into XML file
- Review ProparseHelper methods
- Review TempTableParser methods

Demo

```
ASSIGN oNode = pu:getTopNode():firstChild() .

/* Assuming, the temp-table is not falsely defined inside a block */
DO WHILE VALID-OBJECT (oNode):
    IF NodeTypes:getTypeName(oNode:getType()) = "DEFINE" AND
       ProparseHelper:HasChildNodeOfNodeType(oNode, "TEMPTABLE") THEN DO:

        ASSIGN cTempTableName = ProparseHelper:FindChildNodeOfNodeType(oNode, "ID"):getText ().

        oParser = NEW TempTableParser() .

        oParser:ProcessTable(oNode, cTempTableName) .
        oParser:GetTable (OUTPUT TABLE eField) .

        TEMP-TABLE eField:WRITE-XML ("file",
                                     SUBSTITUTE ("&1.xml", cTempTableName),
                                     YES, ?, ?) .
    END.

    oNode = oNode:nextSibling () .
END.
```

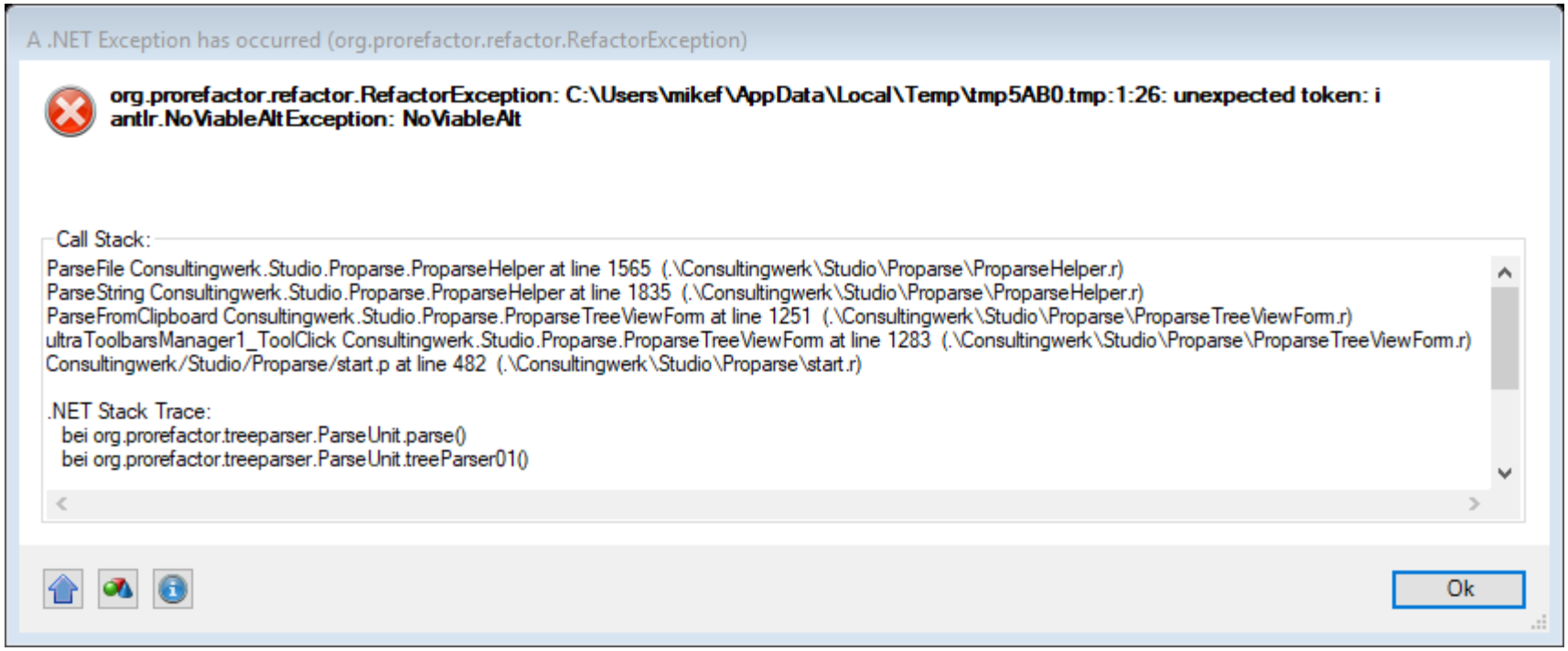
Agenda

- Why Source Code Analysis
- Proparse
- Utilities based on Proparse
- Proparse.NET
- Using Proparse from ABL
- **Building and enhancing Proparse**

Maintaining Proparse

- ABL Syntax is evolving, new keywords added in almost every release.
  - 11.7 added SERIALIZABLE options for class members
  - Proparse requires knowledge of keywords and syntax
- Some “odd” syntax constructs may cause parsing issues (e.g. parenthesis in unexpected locations)

DEF VAR VAR i AS INTEGER NO-UNDO .



- Error message refers to file, line number and column of the token causing the issue.

Keeping Proparse up to date

- ANTLR grammars need to be extended with new syntax constructs
- KEYWORDS need to be added to vocabulary ("*importVocab*")
- Built-in functions need to be added to "*builtinfunc*" rule
- Almost everything needs to be added to "*NodeTypes*" (keyword, reserved, function, system handle, etc…)
- Extended parser ("*treeparser01*") – add some scope detection and references support
- Parser for preprocessor code evaluation ("*proeval*")

Tooling

- Ecliplse, JDT based
- ANTLR IDE doesn't support ANTLR version 2.x ☹
- Old Eclipse plugin works with Eclipse Mars (4.5.2) - http://antlreclipse.sourceforge.net

Build new version

- Add new keywords to vocabulary – *BaseTokenTypes.txt*
- Update grammar file - *proparse.g*
- ”Compile” grammar file
- Compilation “translate” the grammar file to two Java classes that implements the parser: *ProParser.java* and *ProParserTokenTypes.java*
- Add new entries on *NodeTypes.java*
- Run Unit Test scripts, add new one as needed or simply add new syntax samples to be validated

Building Proparse.NET

- Build proparse.jar from Java binaries

```
<jar jarfile="proparse.jar" >

    <manifest>
        <attribute name="Author" value="Joanju Software"/>
        <attribute name="Class-Path" value=". ${lib.list}"/>
        <section name="Proparse">
            <attribute name="Author" value="Joanju Software" />
            <attribute name="Home" value="joanju.com, oehive.org"/>
            <attribute name="Build" value="${build.number}"/>
            <attribute name="Date" value="${TODAY}"/>
        </section>
        <section name="Copyright">
            <attribute name="Copy" value="(C) Joanju Software 2002-2011"/>
            <attribute name="License" value="Eclipse Public License version 1.0"/>
        </section>
    </manifest>

    <zipfileset dir="bin" excludes="${core_xclds}" />

</jar>
```

Building Proparse .NET

- Build proparse.net.dll from proparse.jar

```
<target name="make_dotnet">
    <exec executable="C:\Work\Proparse\Github\proparse\ikvmbin\ikvmc.exe" dir="." failonerror="true">
        <arg line="-out:proparse.net.dll -version:4.0.1.${build.number} -keyfile:proparse.snk proparse.jar lib\*.jar"/>
    </exec>
    <delete dir="prorefactor/projects/unittest/pubs" />
    <zip destfile="proparse.assemblies.zip">
        <zipfileset file="proparse.net.dll" prefix="proparse.assemblies" />
        <zipfileset file="ikvmbin/IKVM.OpenJDK.Core.dll" prefix="proparse.assemblies" />
        <zipfileset file="ikvmbin/IKVM.Runtime.dll" prefix="proparse.assemblies" />
    </zip>
</target>
```

- ANT script contained in Github repo
- Requires IKVM.NET tools
- Compare .dll size before and after

Future tasks

- Reduce redundancy in code caused by legacy
  - Proparse
  - Prorefactor
- Upgrade to more recent ANTLR
- Keep up with new ABL syntax

Questions