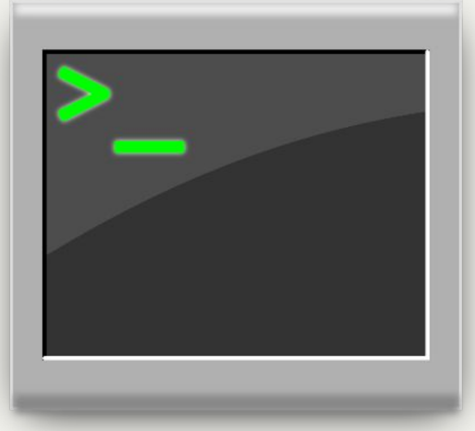
An Industrial Case Study in Compiler Testing

11th International Conference on Software Language Engineering

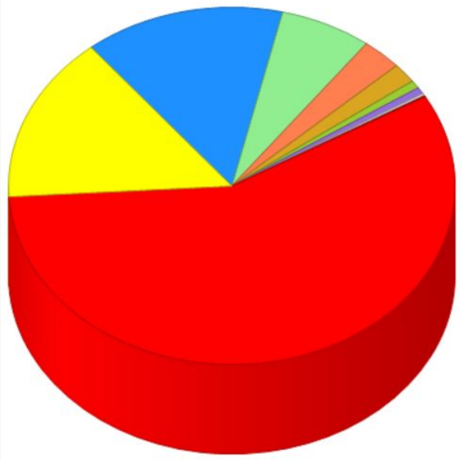
Dr. Vadim Zaytsev aka @grammarware

raincode LABS compiler experts

Introduction

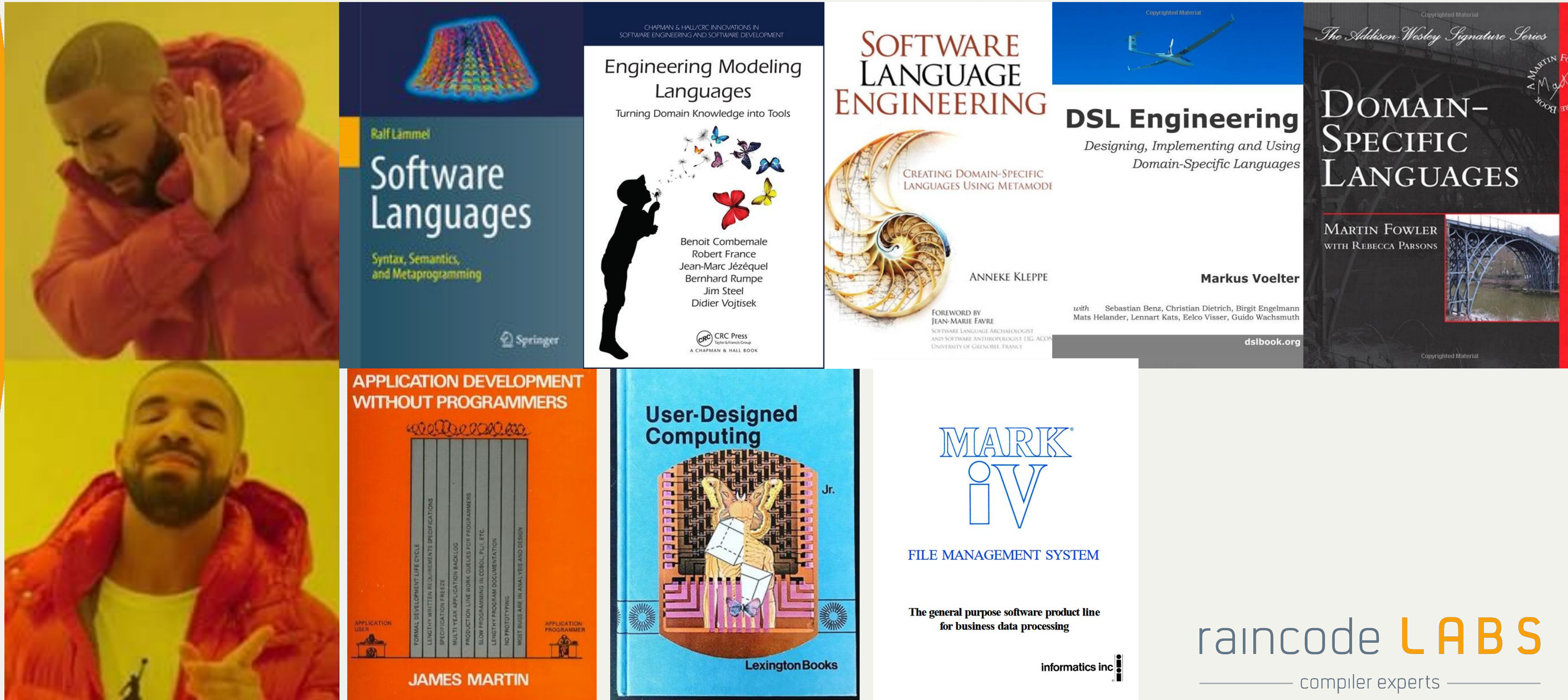


- I am @grammarware
- MSc (ru, nl), PhD (nl/de)
- Ex-researcher (UKL, CWI)
- Ex-lecturer (UvA)
- Now @ Raincode [Labs]
- Chief Science Officer
- Writes compilers for a living
- Last project: TIALAA (4GL)
  - There Is A Life After AppBuilder
  - https://www.raincode.com/technical-landscape/tialaa/



raincode LABS
compiler experts

4GLs are [Badly Designed] DSLs

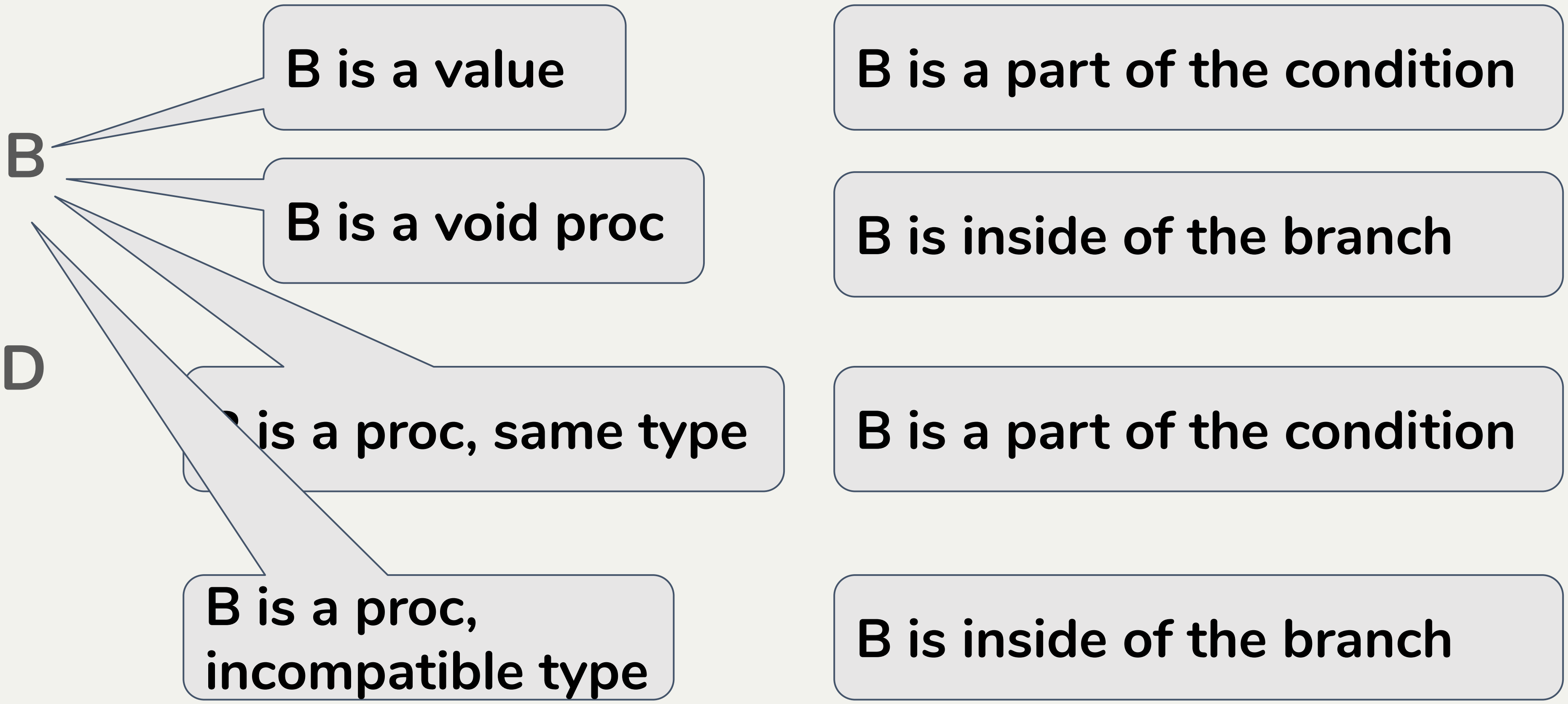

Ralf Lämmel
Software Languages
Syntax, Semantics, and Metaprogramming
Springer
CHAPMAN & HALL/CRC INNOVATIONS IN SOFTWARE ENGINEERING AND SOFTWARE DEVELOPMENT
Engineering Modeling Languages
Turning Domain Knowledge into Tools
Benoit Combemale
Robert France
Jean-Marc Jézéquel
Bernhard Rumpe
Jim Steel
Didier Vojtisek
CRC Press
A CHAPMAN & HALL BOOK
SOFTWARE LANGUAGE ENGINEERING
CREATING DOMAIN-SPECIFIC LANGUAGES USING METAMODE
ANNEKE KLEPPE
FOREWORD BY JEAN-MARIE FAVRE
SOFTWARE LANGUAGE ARCHAEOLOGIST AND SOFTWARE ANTHROPOLOGIST, LIG, ACON
UNIVERSITY OF GRENOBLE, FRANCE
Copyrighted Material
DSL Engineering
Designing, Implementing and Using Domain-Specific Languages
Markus Voelter
with Sebastian Benz, Christian Dietrich, Birgit Engelmann
Mats Helander, Lennart Kats, Eelco Visser, Guido Wachsmuth
dslbook.org
Copyrighted Material
The Addison Wesley Signature Series
DOMAIN-SPECIFIC LANGUAGES
MARTIN FOWLER
WITH REBECCA PARSONS
Copyrighted Material
APPLICATION DEVELOPMENT WITHOUT PROGRAMMERS
FORMAL DEVELOPMENT LIFE CYCLE
LENGTHY WRITTEN REQUIREMENTS SPECIFICATIONS
SPECIFICATION FREEZE
MULTI YEAR APPLICATION BACKLOG
PRODUCTION LINE WORK QUEUES FOR PROGRAMMERS
SLOW PROGRAMMING IN COBOL, PL/I, ETC
LENGTHY PROGRAM DOCUMENTATION
NO PROTOTYPING
MOST BUGS ARE IN ANALYSIS AND DESIGN
APPLICATION USER
APPLICATION PROGRAMMER
JAMES MARTIN
User-Designed Computing
Jr.
Lexington Books
MARK IV
FILE MANAGEMENT SYSTEM
The general purpose software product line for business data processing
informatics inc

AppBuilder concepts are dialectic

- A “rule” is program
  - not declarative
- A “set” is a lookup table
  - not a set at all
- A “view” is a model
  - in MVC terms
- “SetEncoding” is a getter
  - while “SetVisible” is a setter

AppBuilder rule syntax is ambiguous

```
CASEOF X
    CASE A B
        FOO
    CASE C D
        BAR
ENDCASE
```



B is a value

B is a part of the condition

B is a void proc

B is inside of the branch

B is a proc, same type

B is a part of the condition

B is a proc,
incompatible type

B is inside of the branch

AppBuilder typing is not name-unique

MAP A IN A TO A

MAP A OF B TO X // might mean A OF C OF B

MAP A OF B OF C (N) TO X // the index may refer to A, B or C

AppBuilder semantics is *special*

- TRUNC(X, -3)
- DATE(‘00-00-0000’)
- DATE(‘00-00-0000’)+1
- DATE(‘00-00-0000’)-1
- USE RULE X // closes the window if client-client call
- MAP A TO B

Need for testing methodology

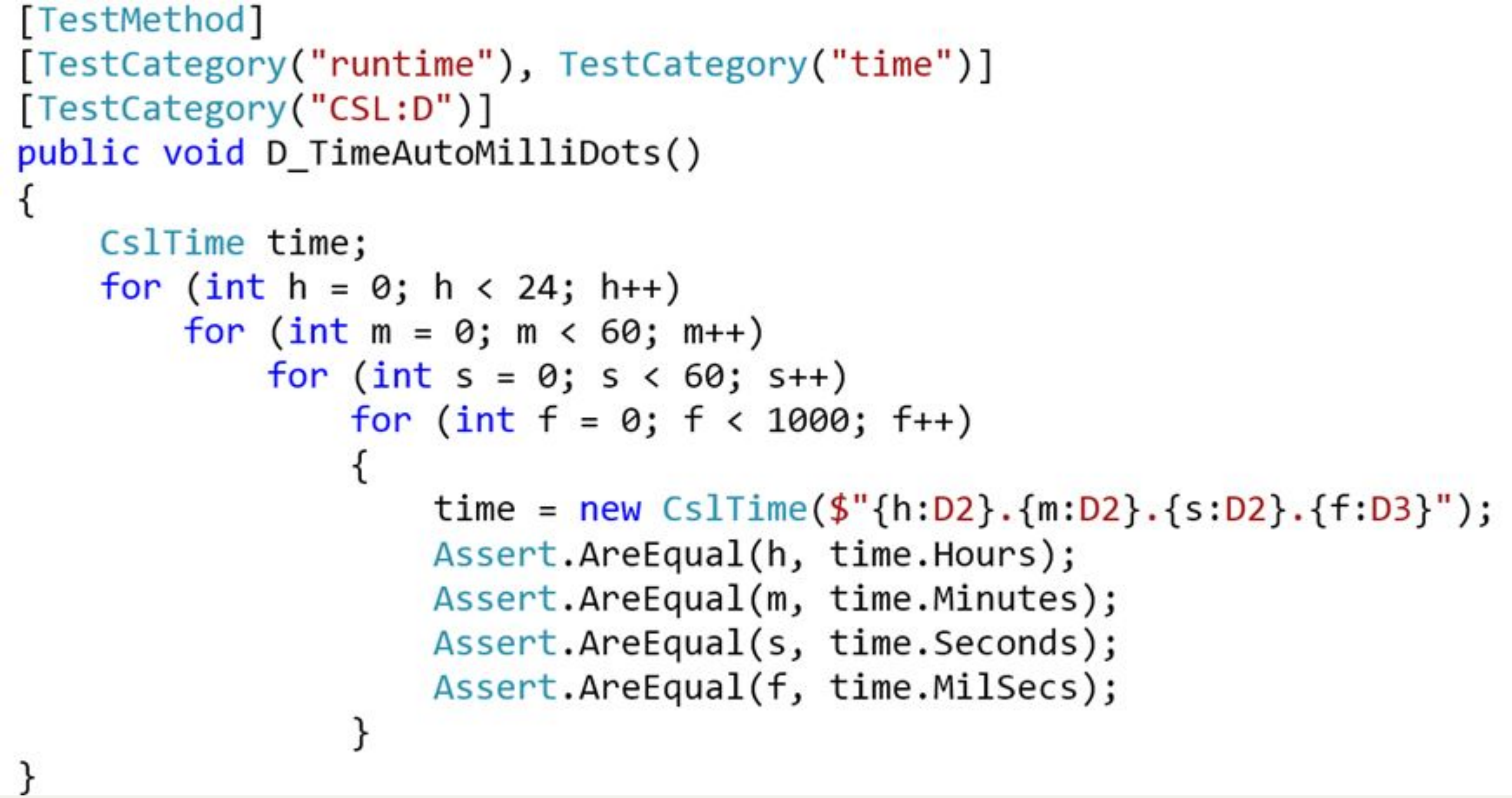
DESPERATE

need for

compiler testing

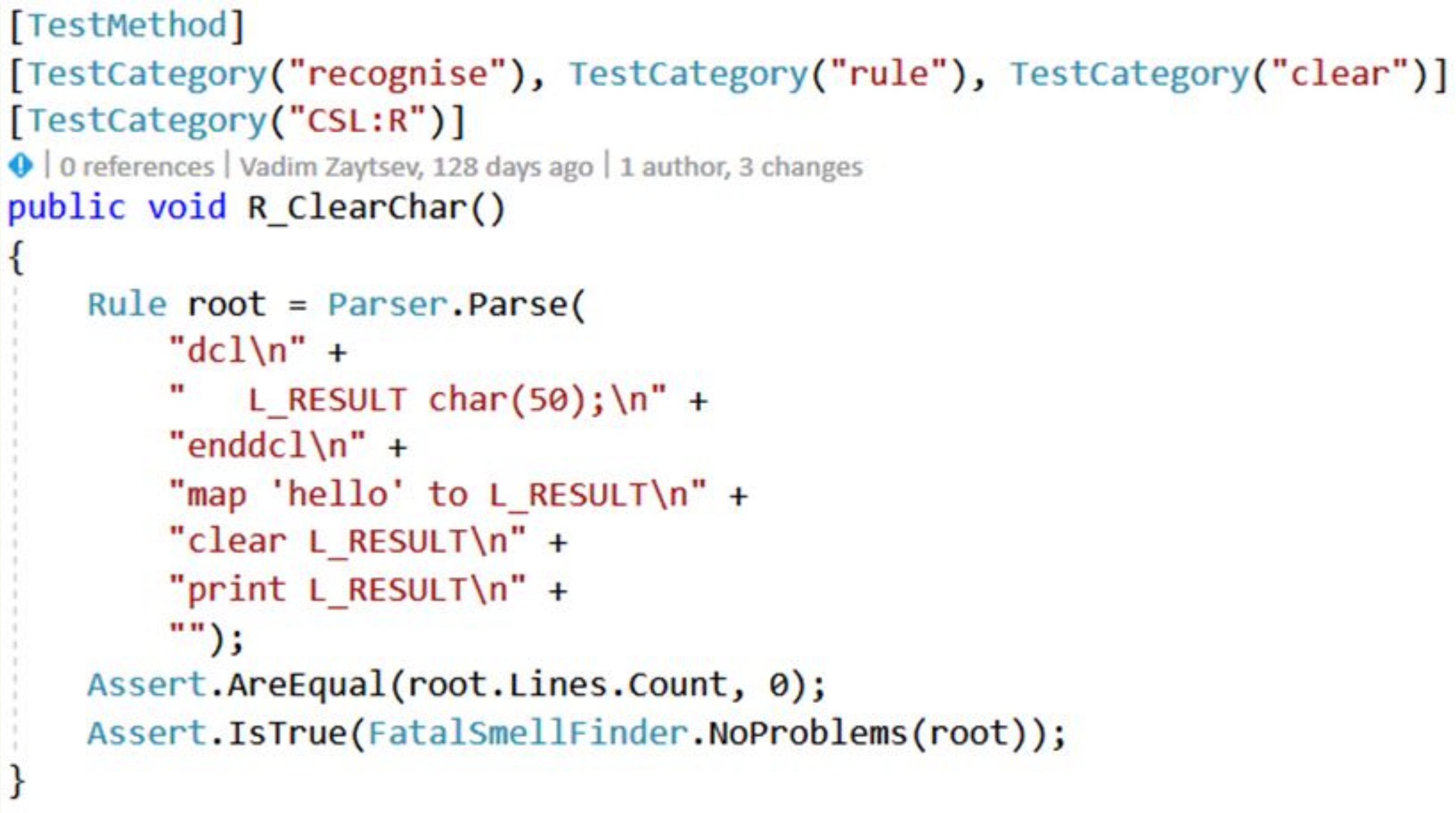
methodology

D-tests: direct access to runtime



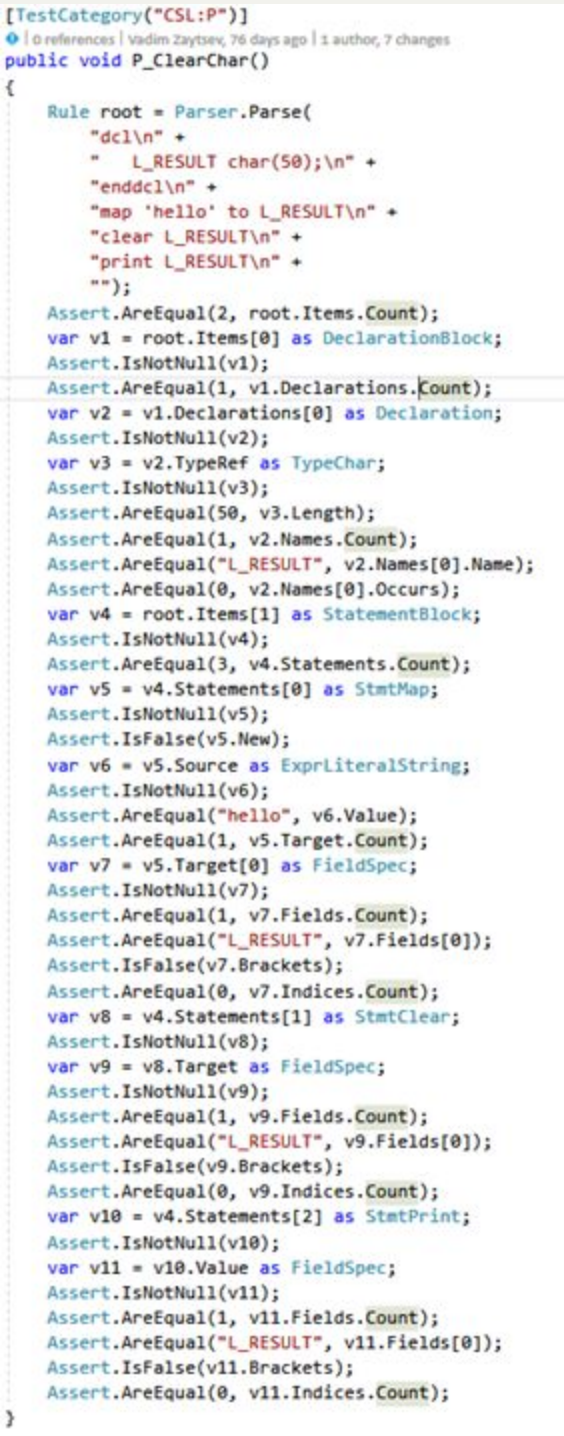
```
[TestMethod]
[TestCategory("runtime"), TestCategory("time")]
[TestCategory("CSL:D")]
public void D_TimeAutoMilliDots()
{
    CslTime time;
    for (int h = 0; h < 24; h++)
        for (int m = 0; m < 60; m++)
            for (int s = 0; s < 60; s++)
                for (int f = 0; f < 1000; f++)
                {
                    time = new CslTime($"{h:D2}.{m:D2}.{s:D2}.{f:D3}");
                    Assert.AreEqual(h, time.Hours);
                    Assert.AreEqual(m, time.Minutes);
                    Assert.AreEqual(s, time.Seconds);
                    Assert.AreEqual(f, time.MilSecs);
                }
}
```

R-tests: yes/no recognition

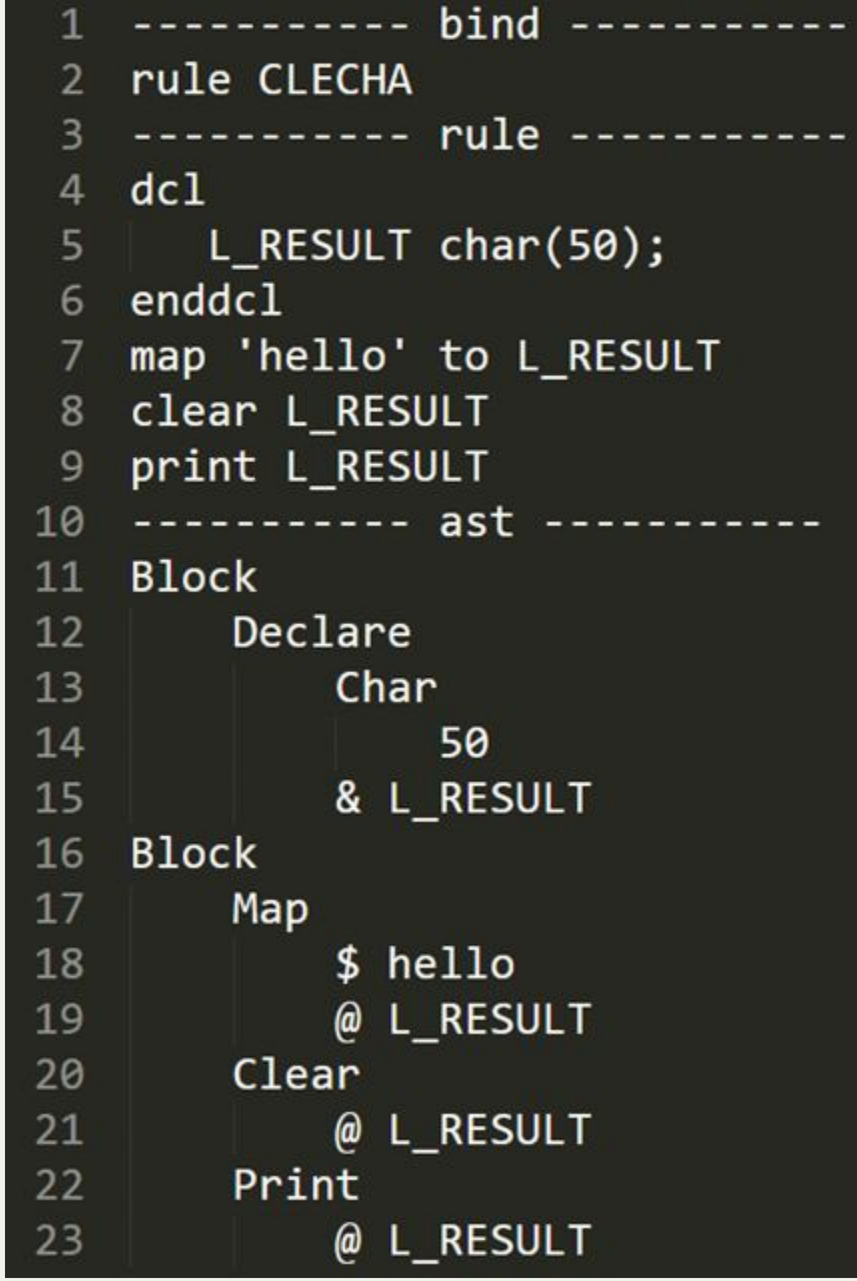


```
[TestMethod]
[TestCategory("recognise"), TestCategory("rule"), TestCategory("clear")]
[TestCategory("CSL:R")]
| 0 references | Vadim Zaytsev, 128 days ago | 1 author, 3 changes
public void R_ClearChar()
{
    Rule root = Parser.Parse(
        "dcl\n" +
        "    L_RESULT char(50);\n" +
        "enddcl\n" +
        "map 'hello' to L_RESULT\n" +
        "clear L_RESULT\n" +
        "print L_RESULT\n" +
        "");
    Assert.AreEqual(root.Lines.Count, 0);
    Assert.IsTrue(FatalSmellFinder.NoProblems(root));
}
```

P-tests: parsing

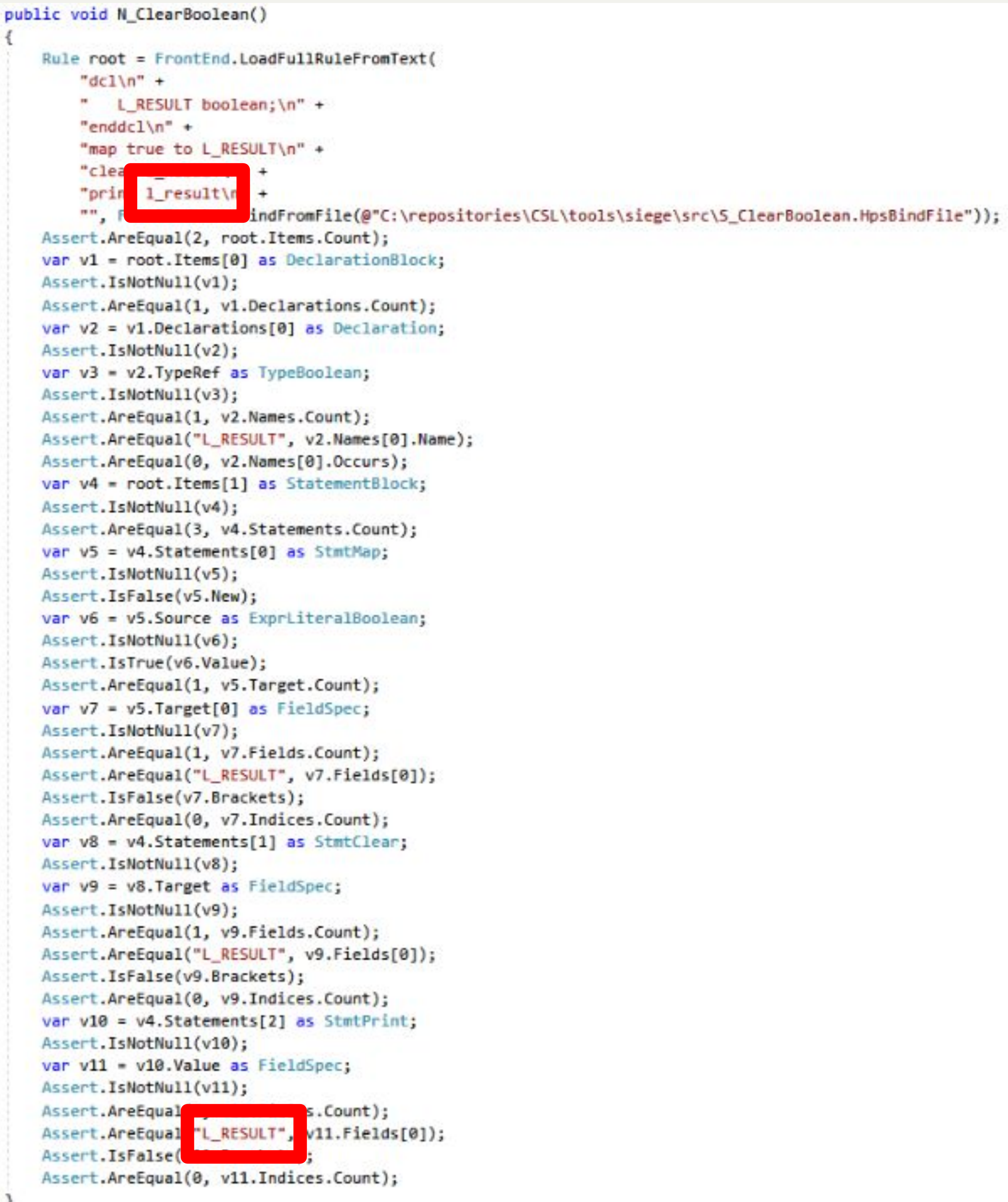


```
[TestCategory("CSL:P")]
| 0 references | Vadim Zaytsev, 76 days ago | 1 author, 7 changes
public void P_ClearChar()
{
    Rule root = Parser.Parse(
        "dcl\n" +
        "   L_RESULT char(50);\n" +
        "enddcl\n" +
        "map 'hello' to L_RESULT\n" +
        "clear L_RESULT\n" +
        "print L_RESULT\n" +
        "");
    Assert.AreEqual(2, root.Items.Count);
    var v1 = root.Items[0] as DeclarationBlock;
    Assert.IsNotNull(v1);
    Assert.AreEqual(1, v1.Declarations.Count);
    var v2 = v1.Declarations[0] as Declaration;
    Assert.IsNotNull(v2);
    var v3 = v2.TypeRef as TypeChar;
    Assert.IsNotNull(v3);
    Assert.AreEqual(50, v3.Length);
    Assert.AreEqual(1, v2.Names.Count);
    Assert.AreEqual("L_RESULT", v2.Names[0].Name);
    Assert.AreEqual(0, v2.Names[0].Occurs);
    var v4 = root.Items[1] as StatementBlock;
    Assert.IsNotNull(v4);
    Assert.AreEqual(3, v4.Statements.Count);
    var v5 = v4.Statements[0] as StmtMap;
    Assert.IsNotNull(v5);
    Assert.IsFalse(v5.New);
    var v6 = v5.Source as ExprLiteralString;
    Assert.IsNotNull(v6);
    Assert.AreEqual("hello", v6.Value);
    Assert.AreEqual(1, v5.Target.Count);
    var v7 = v5.Target[0] as FieldSpec;
    Assert.IsNotNull(v7);
    Assert.AreEqual(1, v7.Fields.Count);
    Assert.AreEqual("L_RESULT", v7.Fields[0]);
    Assert.IsFalse(v7.Brackets);
    Assert.AreEqual(0, v7.Indices.Count);
    var v8 = v4.Statements[1] as StmtClear;
    Assert.IsNotNull(v8);
    var v9 = v8.Target as FieldSpec;
    Assert.IsNotNull(v9);
    Assert.AreEqual(1, v9.Fields.Count);
    Assert.AreEqual("L_RESULT", v9.Fields[0]);
    Assert.IsFalse(v9.Brackets);
    Assert.AreEqual(0, v9.Indices.Count);
    var v10 = v4.Statements[2] as StmtPrint;
    Assert.IsNotNull(v10);
    var v11 = v10.Value as FieldSpec;
    Assert.IsNotNull(v11);
    Assert.AreEqual(1, v11.Fields.Count);
    Assert.AreEqual("L_RESULT", v11.Fields[0]);
    Assert.IsFalse(v11.Brackets);
    Assert.AreEqual(0, v11.Indices.Count);
}
```



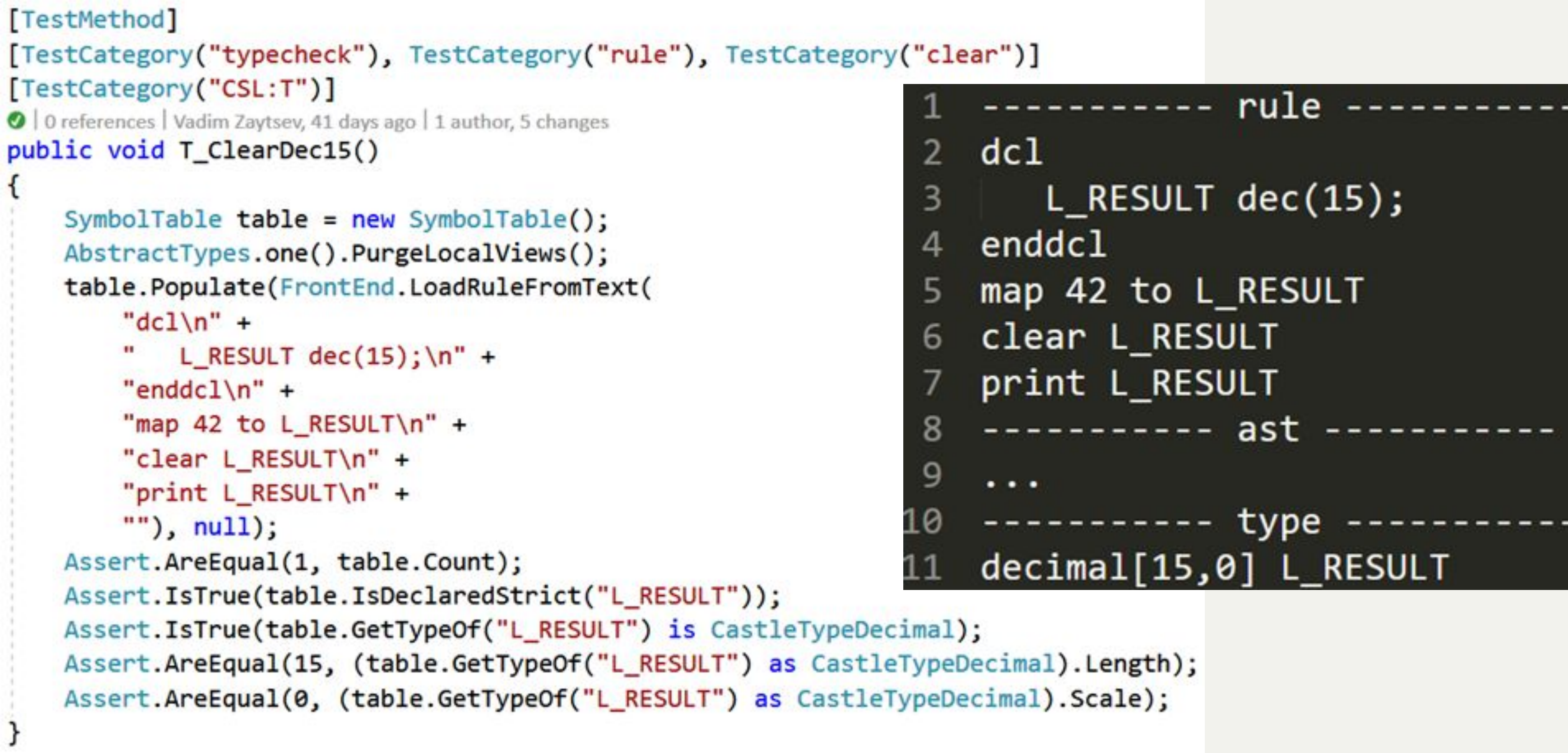
```
----------- bind ----------
rule CLECHA
----------- rule ----------
dcl
    L_RESULT char(50);
enddcl
map 'hello' to L_RESULT
clear L_RESULT
print L_RESULT
----------- ast ----------
Block
    Declare
        Char
            50
        & L_RESULT
Block
    Map
        $ hello
        @ L_RESULT
    Clear
        @ L_RESULT
    Print
        @ L_RESULT
```

N-tests: normalisation



```
public void N_ClearBoolean()
{
    Rule root = FrontEnd.LoadFullRuleFromText(
        "dcl\n" +
        "   L_RESULT boolean;\n" +
        "enddcl\n" +
        "map true to L_RESULT\n" +
        "clea            +
        "prin  l_result\n  +
        "", F              indFromFile(@"C:\repositories\CSL\tools\siege\src\S_ClearBoolean.HpsBindFile"));
    Assert.AreEqual(2, root.Items.Count);
    var v1 = root.Items[0] as DeclarationBlock;
    Assert.IsNotNull(v1);
    Assert.AreEqual(1, v1.Declarations.Count);
    var v2 = v1.Declarations[0] as Declaration;
    Assert.IsNotNull(v2);
    var v3 = v2.TypeRef as TypeBoolean;
    Assert.IsNotNull(v3);
    Assert.AreEqual(1, v2.Names.Count);
    Assert.AreEqual("L_RESULT", v2.Names[0].Name);
    Assert.AreEqual(0, v2.Names[0].Occurs);
    var v4 = root.Items[1] as StatementBlock;
    Assert.IsNotNull(v4);
    Assert.AreEqual(3, v4.Statements.Count);
    var v5 = v4.Statements[0] as StmtMap;
    Assert.IsNotNull(v5);
    Assert.IsFalse(v5.New);
    var v6 = v5.Source as ExprLiteralBoolean;
    Assert.IsNotNull(v6);
    Assert.IsTrue(v6.Value);
    Assert.AreEqual(1, v5.Target.Count);
    var v7 = v5.Target[0] as FieldSpec;
    Assert.IsNotNull(v7);
    Assert.AreEqual(1, v7.Fields.Count);
    Assert.AreEqual("L_RESULT", v7.Fields[0]);
    Assert.IsFalse(v7.Brackets);
    Assert.AreEqual(0, v7.Indices.Count);
    var v8 = v4.Statements[1] as StmtClear;
    Assert.IsNotNull(v8);
    var v9 = v8.Target as FieldSpec;
    Assert.IsNotNull(v9);
    Assert.AreEqual(1, v9.Fields.Count);
    Assert.AreEqual("L_RESULT", v9.Fields[0]);
    Assert.IsFalse(v9.Brackets);
    Assert.AreEqual(0, v9.Indices.Count);
    var v10 = v4.Statements[2] as StmtPrint;
    Assert.IsNotNull(v10);
    var v11 = v10.Value as FieldSpec;
    Assert.IsNotNull(v11);
    Assert.AreEqual             s.Count);
    Assert.AreEqual "L_RESULT",  v11.Fields[0]);
    Assert.IsFalse(             ;
    Assert.AreEqual(0, v11.Indices.Count);
}
```

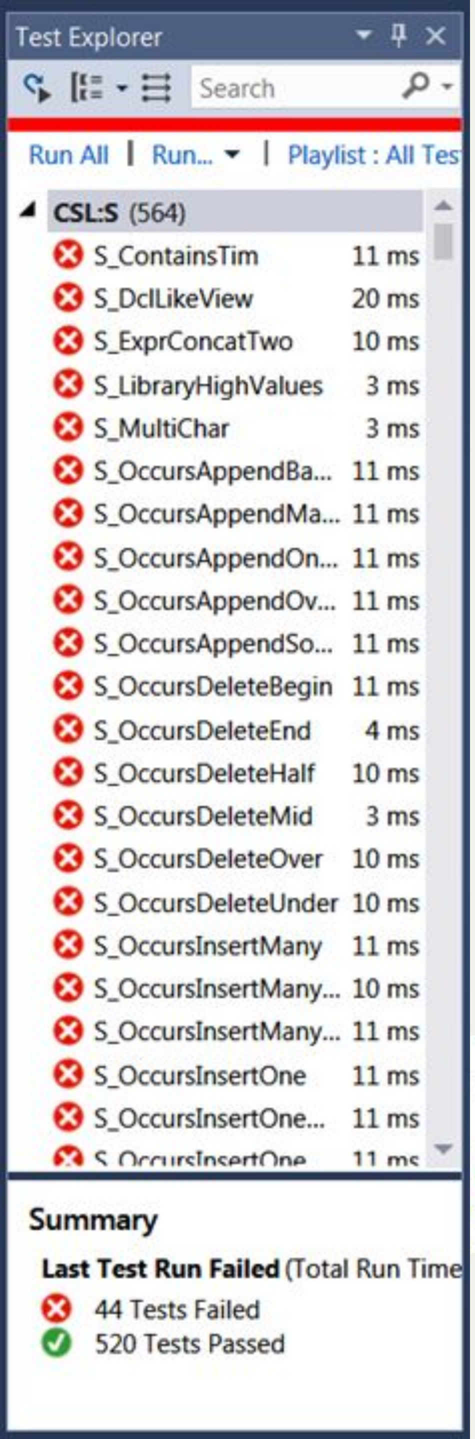
T-tests: typing



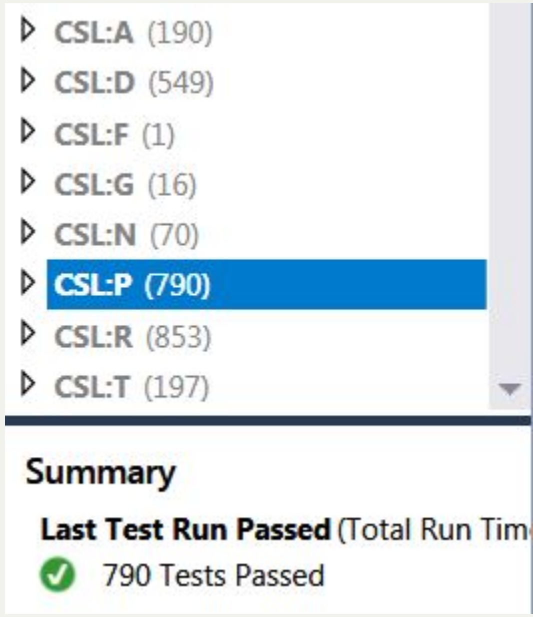
```
[TestMethod]
[TestCategory("typecheck"), TestCategory("rule"), TestCategory("clear")]
[TestCategory("CSL:T")]
✓ | 0 references | Vadim Zaytsev, 41 days ago | 1 author, 5 changes
public void T_ClearDec15()
{
    SymbolTable table = new SymbolTable();
    AbstractTypes.one().PurgeLocalViews();
    table.Populate(FrontEnd.LoadRuleFromText(
        "dcl\n" +
        "   L_RESULT dec(15);\n" +
        "enddcl\n" +
        "map 42 to L_RESULT\n" +
        "clear L_RESULT\n" +
        "print L_RESULT\n" +
        ""), null);
    Assert.AreEqual(1, table.Count);
    Assert.IsTrue(table.IsDeclaredStrict("L_RESULT"));
    Assert.IsTrue(table.GetTypeOf("L_RESULT") is CastleTypeDecimal);
    Assert.AreEqual(15, (table.GetTypeOf("L_RESULT") as CastleTypeDecimal).Length);
    Assert.AreEqual(0, (table.GetTypeOf("L_RESULT") as CastleTypeDecimal).Scale);
}
```

```
----------- rule ----------
dcl
    L_RESULT dec(15);
enddcl
map 42 to L_RESULT
clear L_RESULT
print L_RESULT
----------- ast ----------
...
----------- type ----------
decimal[15,0] L_RESULT
```

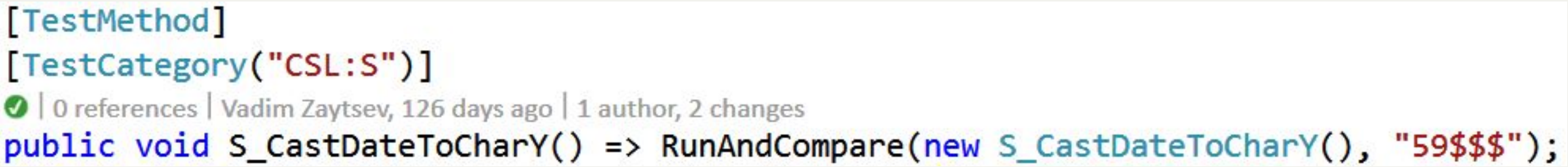
S-tests: successful execution



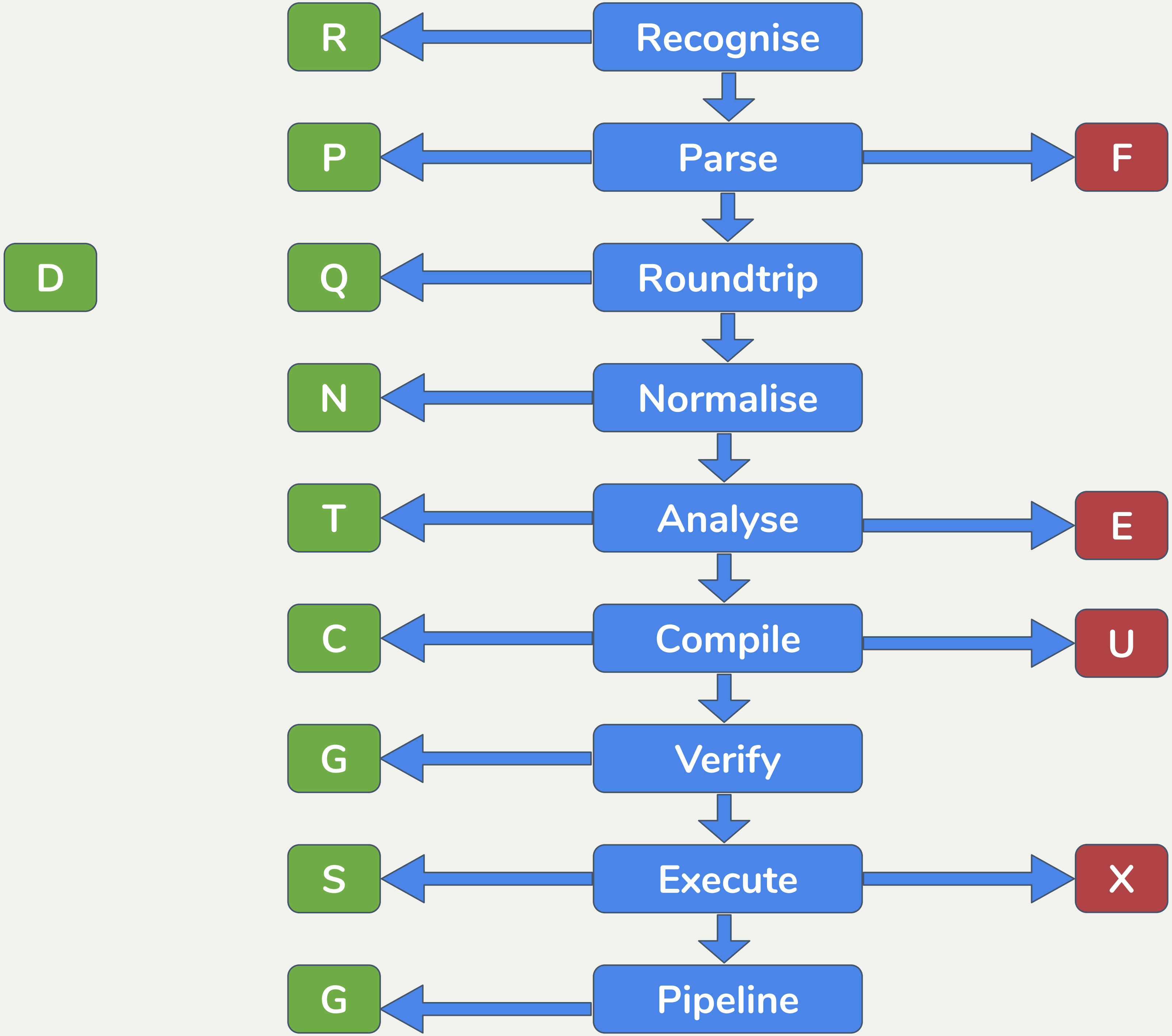
```
CSL:S (564)
S_ContainsTim          11 ms
S_DclLikeView          20 ms
S_ExprConcatTwo        10 ms
S_LibraryHighValues     3 ms
S_MultiChar             3 ms
S_OccursAppendBa...    11 ms
S_OccursAppendMa...    11 ms
S_OccursAppendOn...    11 ms
S_OccursAppendOv...    11 ms
S_OccursAppendSo...    11 ms
S_OccursDeleteBegin    11 ms
S_OccursDeleteEnd       4 ms
S_OccursDeleteHalf     10 ms
S_OccursDeleteMid       3 ms
S_OccursDeleteOver     10 ms
S_OccursDeleteUnder    10 ms
S_OccursInsertMany     11 ms
S_OccursInsertMany...  10 ms
S_OccursInsertMany...  11 ms
S_OccursInsertOne      11 ms
S_OccursInsertOne...   11 ms
S_OccursInsertOne      11 ms

Summary
Last Test Run Failed (Total Run Time
44 Tests Failed
520 Tests Passed
```



```
CSL:A (190)
CSL:D (549)
CSL:F (1)
CSL:G (16)
CSL:N (70)
CSL:P (790)
CSL:R (853)
CSL:T (197)

Summary
Last Test Run Passed (Total Run Tim
790 Tests Passed
```



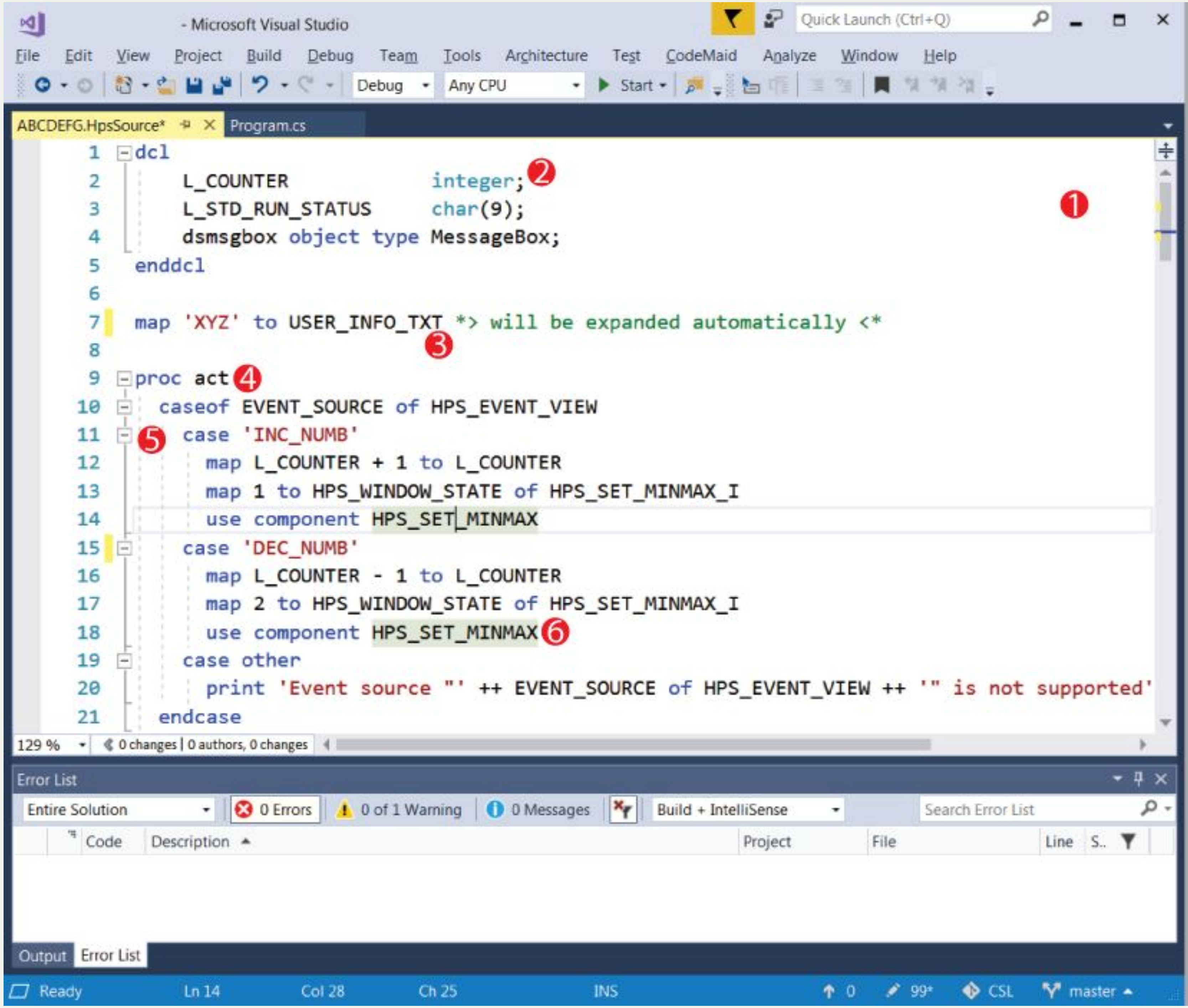
```
[TestMethod]
[TestCategory("CSL:S")]
✓ | 0 references | Vadim Zaytsev, 126 days ago | 1 author, 2 changes
public void S_CastDateToCharY() => RunAndCompare(new S_CastDateToCharY(), "59$$$");
```



raincode LABS
compiler experts

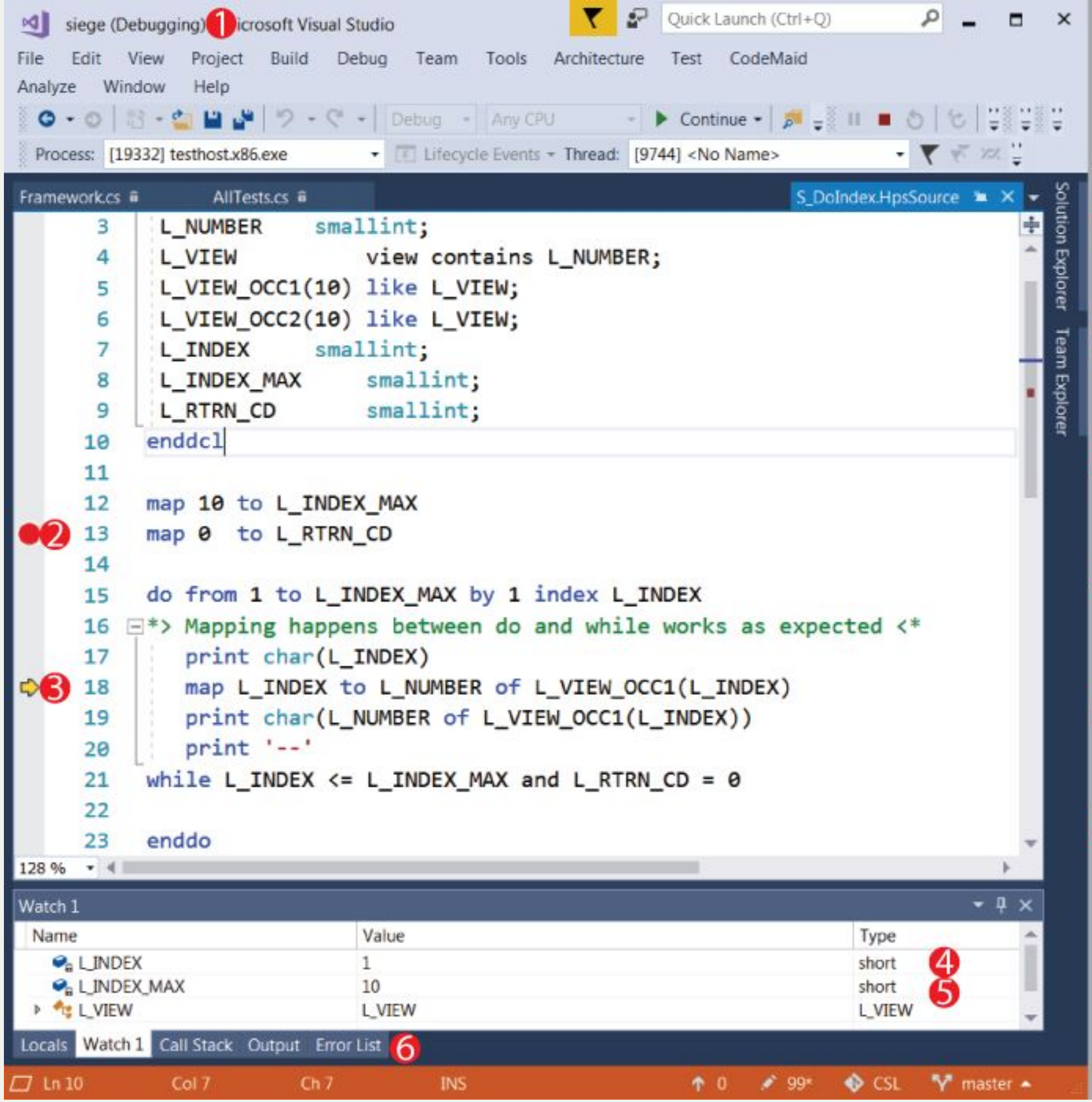
Tests were used:

- Mostly during middle stages of the project
  - too little info early on
  - easy to test bugfixes for regression later on
- To measure progress internally
  - project planning
  - work distribution
- To report to the customer
  - challenging to communicate a CC process



```
dcl
    L_COUNTER            integer;
    L_STD_RUN_STATUS     char(9);
    dsmsgbox object type MessageBox;
enddcl

map 'XYZ' to USER_INFO_TXT *> will be expanded automatically <*

proc act
  caseof EVENT_SOURCE of HPS_EVENT_VIEW
    case 'INC_NUMB'
      map L_COUNTER + 1 to L_COUNTER
      map 1 to HPS_WINDOW_STATE of HPS_SET_MINMAX_I
      use component HPS_SET_MINMAX
    case 'DEC_NUMB'
      map L_COUNTER - 1 to L_COUNTER
      map 2 to HPS_WINDOW_STATE of HPS_SET_MINMAX_I
      use component HPS_SET_MINMAX
    case other
      print 'Event source "' ++ EVENT_SOURCE of HPS_EVENT_VIEW ++ '" is not supported'
  endcase
```

1. Native syntax
2. Data types
3. Symbol expanstion
4. Procedures
5. Code folding
6. System components

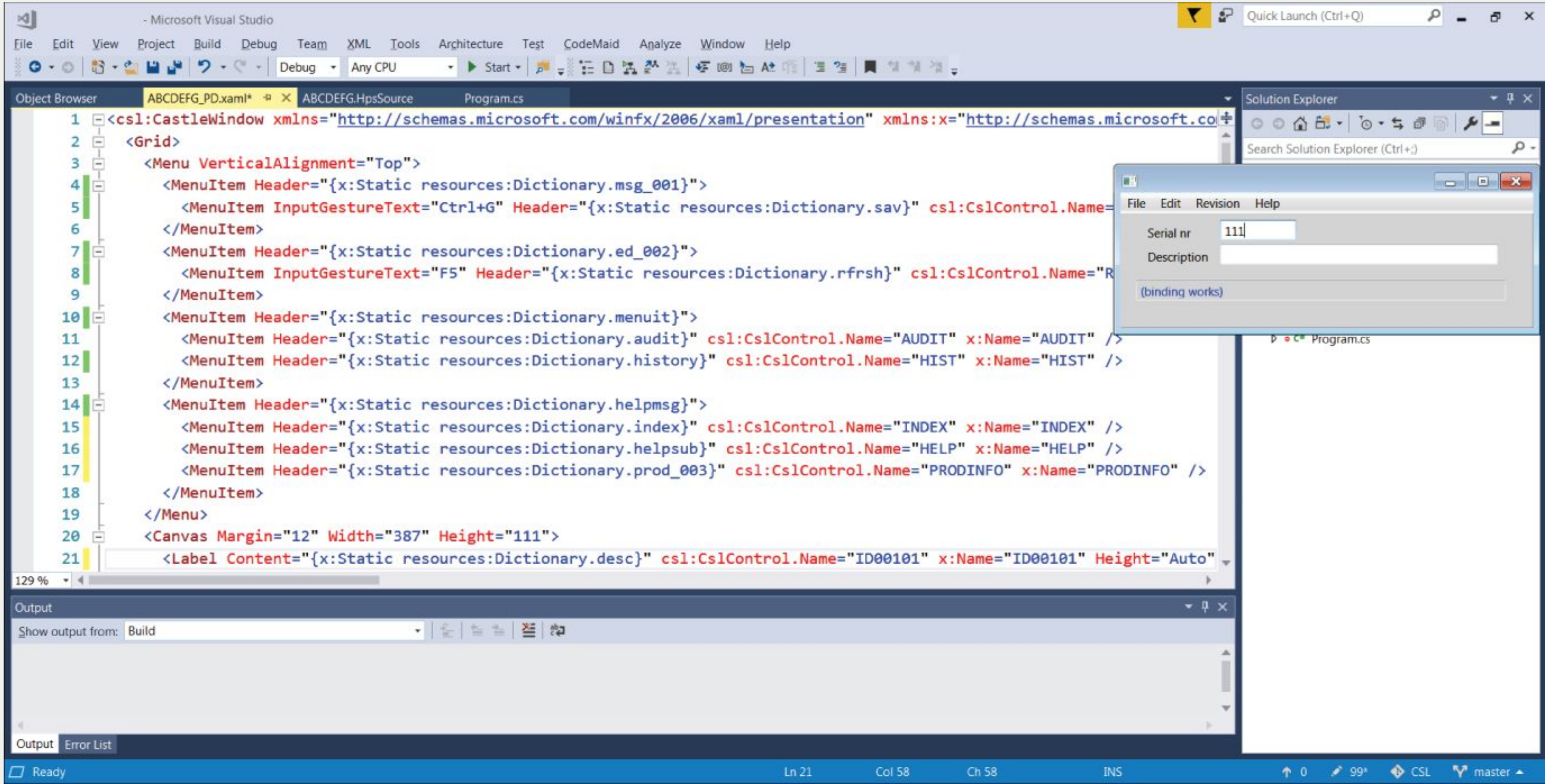


```
 L_NUMBER    smallint;
 L_VIEW          view contains L_NUMBER;
 L_VIEW_OCC1(10) like L_VIEW;
 L_VIEW_OCC2(10) like L_VIEW;
 L_INDEX     smallint;
 L_INDEX_MAX     smallint;
 L_RTRN_CD       smallint;
enddcl

map 10 to L_INDEX_MAX
map 0  to L_RTRN_CD

do from 1 to L_INDEX_MAX by 1 index L_INDEX
*> Mapping happens between do and while works as expected <*
   print char(L_INDEX)
   map L_INDEX to L_NUMBER of L_VIEW_OCC1(L_INDEX)
   print char(L_NUMBER of L_VIEW_OCC1(L_INDEX))
   print '--'
while L_INDEX <= L_INDEX_MAX and L_RTRN_CD = 0

enddo
```

Name	Value	Type
L_INDEX	1	short
L_INDEX_MAX	10	short
L_VIEW	L_VIEW	L_VIEW

1. Debugging
2. Breakpoint
3. Currently executing
4. Watch on a field
5. Watch on a view
6. Locals show known views

TIALAA supports both client & server



```
<csl:CastleWindow xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.co
  <Grid>
    <Menu VerticalAlignment="Top">
      <MenuItem Header="{x:Static resources:Dictionary.msg_001}">
        <MenuItem InputGestureText="Ctrl+G" Header="{x:Static resources:Dictionary.sav}" csl:CslControl.Name=
      </MenuItem>
      <MenuItem Header="{x:Static resources:Dictionary.ed_002}">
        <MenuItem InputGestureText="F5" Header="{x:Static resources:Dictionary.rfrsh}" csl:CslControl.Name="R
      </MenuItem>
      <MenuItem Header="{x:Static resources:Dictionary.menuit}">
        <MenuItem Header="{x:Static resources:Dictionary.audit}" csl:CslControl.Name="AUDIT" x:Name="AUDIT" />
        <MenuItem Header="{x:Static resources:Dictionary.history}" csl:CslControl.Name="HIST" x:Name="HIST" />
      </MenuItem>
      <MenuItem Header="{x:Static resources:Dictionary.helpmsg}">
        <MenuItem Header="{x:Static resources:Dictionary.index}" csl:CslControl.Name="INDEX" x:Name="INDEX" />
        <MenuItem Header="{x:Static resources:Dictionary.helpsub}" csl:CslControl.Name="HELP" x:Name="HELP" />
        <MenuItem Header="{x:Static resources:Dictionary.prod_003}" csl:CslControl.Name="PRODINFO" x:Name="PRODINFO" />
      </MenuItem>
    </Menu>
    <Canvas Margin="12" Width="387" Height="111">
      <Label Content="{x:Static resources:Dictionary.desc}" csl:CslControl.Name="ID00101" x:Name="ID00101" Height="Auto"
```

Serial nr: 111

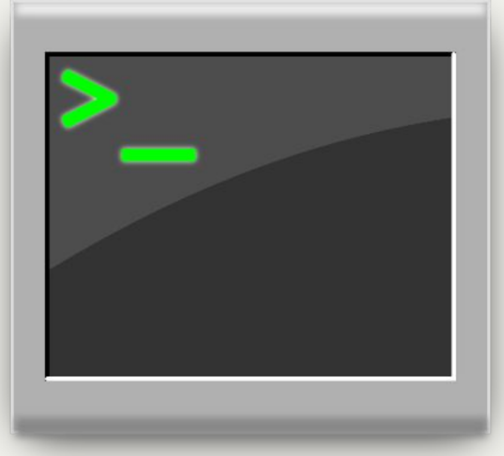
Description:

(binding works)

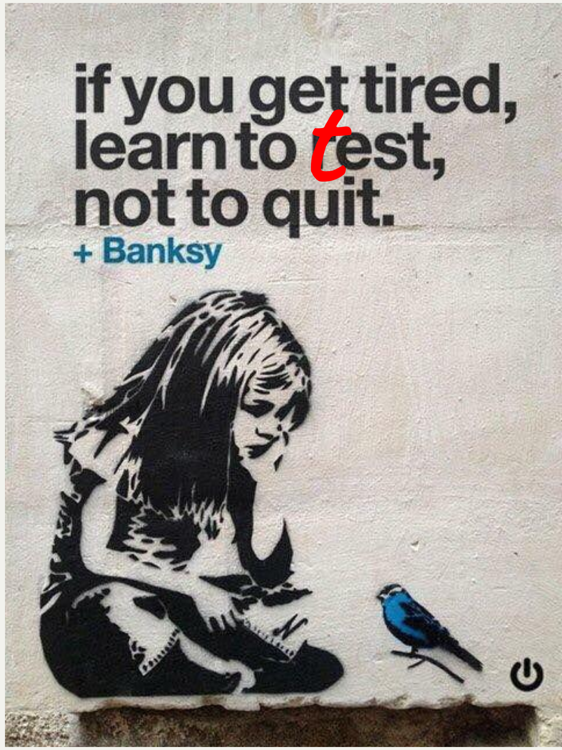
Pilot study results

- ~4 MLOC rules,
  ~13 MLOC bind files,
  ~4 MLOC sets,
  ~3 MLOC panels
- ~42k data structures, ~17k programs
- 100% compilation & verification
- Integration testing ongoing
- Go into production in a few months

Conclusion



- Follow @grammarware
- 4GLs are bad DSLs
- TIALAA is there to replace AppBuilder
  - https://www.raincode.com/technical-landscape/tialaa/
- Testing a compiler is a lot of work
- No out of the box solution
- No out of the box comprehensive methodology
- Existing papers are scarce and focused
- Request for SLEBoK!
  - http://slebok.github.io
- Thanks! Questions?

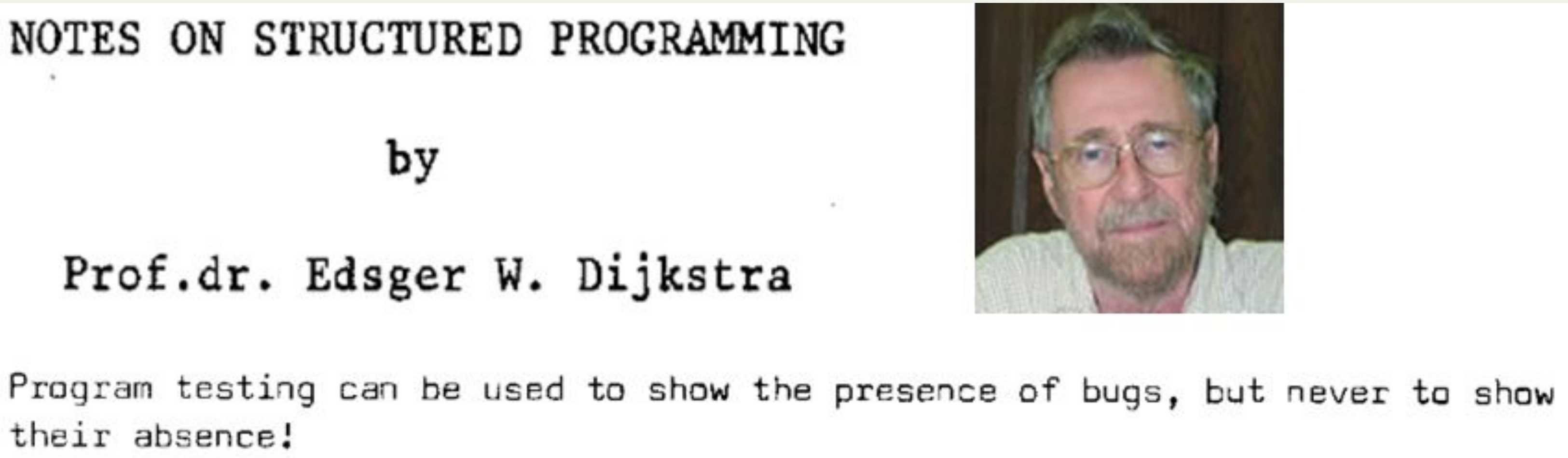


raincode LABS
compiler experts

Testing in TIALAA

- G-tests: can the compiler handle the customer’s codebase?
- R-tests: can the parser recognise this input?
- F-tests: can the parser rightfully reject this input?
- P-tests: can the parser construct a good tree from this input?
- N-tests: can the normaliser rewrite this tree well?
- E-tests: can this input error be fixed automatically?
- T-tests: can this program be typed correctly?
- A-tests: can this program be rejected by static semantic analysis?
- C-tests: can this program be successfully compiled to produce a DLL?
- V-tests: can this program be compiled to a verified DLL?
- U-tests: can this problem be rightfully rejected during compilation?
- S-tests: can this program successfully execute to produce output?
- X-tests: can this program throw the right exception?
- D-tests: does this runtime library function work?

Dijkstra vs Goodenough

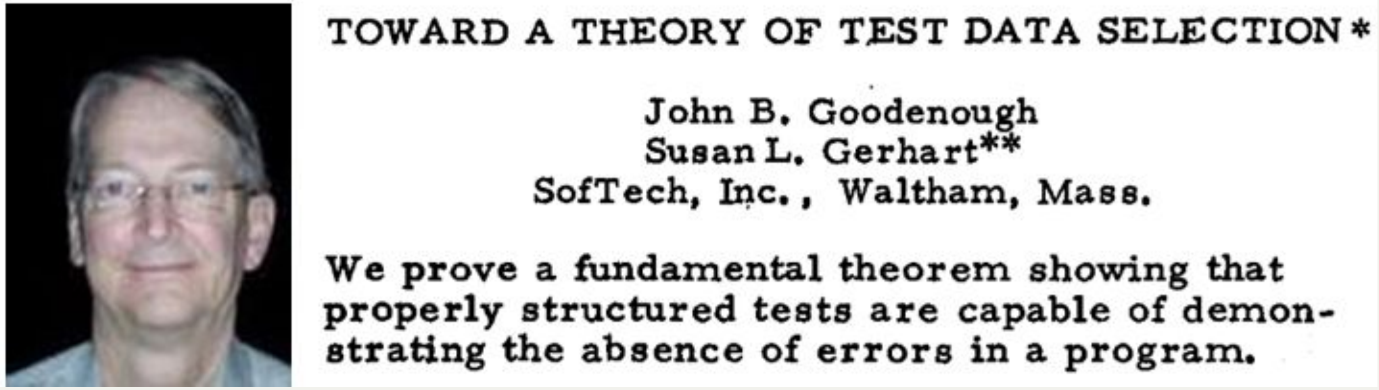


NOTES ON STRUCTURED PROGRAMMING

by

Prof.dr. Edsger W. Dijkstra

Program testing can be used to show the presence of bugs, but never to show their absence!



TOWARD A THEORY OF TEST DATA SELECTION*

John B. Goodenough
Susan L. Gerhart**
SofTech, Inc., Waltham, Mass.

We prove a fundamental theorem showing that properly structured tests are capable of demonstrating the absence of errors in a program.