michel.loehr@saxsys.de www.saxsys.de

Michel Löhr

Highspeed Test Automation

Dresden · Frankfurt/Main · Leipzig · München · Hamburg · Görlitz · Berlin

Highspeed Test Automation

Domain Specific Languages (DSLs)

→ Dedicated languages for a specific purpose like SQL for handling data in databases

→Macros such as SUM() in Spreadsheets (EXCEL)

→Raised abscration level and (re)using domain knowledge

→Opposite of general purpose languages as Java or C#

→Internal / External DSLs

michel.loehr@saxsys.de www.saxsys.de

Highspeed Test Automation

Goal of presentation

→Using a Domain Specific Language approach with automatic testing will increase productivity and quality in creating and maintaining testscripts

→I will showcase this with two examples

Highspeed Test Automation

Software Testing History

→ Bis 1956	- Debugging oriented
→ 1957-1978	- Demonstration oriented
→ 1979-1982	- Destruction oriented
→ 1983-1987	- Evaluation oriented
→ 1988-2000	- Prevention oriented (*)
→ 2000-	- Automation oriented (**)

A study conducted by NIST in 2002 reports that software bugs cost the U.S. economy $59.5 billion annually. More than a third of this cost could be avoided if better software testing was performed.

Quellen:
(*) wikipedia
(**) M. Löhr

Highspeed Test Automation

Rationale

In software development practice, as much as 50% of the total cost is spent on testing to ensure its quality. To reduce such cost, it is imperative to have a solution to automate the testing process.

Automating testing within a software development project shouldn't be itself a software development project within its original project

→ROI

Highspeed Test Automation

Test automation types

- White box testing
  - Code-driven testing
- Black box testing
  - (G)UI testing

Platform and OS independence
Data driven capability (Input Data, Output Data, Meta Data)
Customizable Reporting (DB Access, crystal reports, etc..)
Email Notifications (Automated notification on failure or threshold levels)
Easy debugging and logging
Version control friendly – minimum or zero binary files
Extensible & Customizable (Open APIs to be able to integrate with other tools)
Common Driver (Ant or Maven)
Headless execution for unattended runs (For integration with build process or batch runs)
Support distributed execution environment (distributed test bed)
Distributed application support (distributed SUT)
...

Highspeed Test Automation

Testautomation Approaches

- →Data-driven testing
- →Modularity-driven testing
- →Keyword-driven testing
- →Hybrid testing
- →Model-based testing

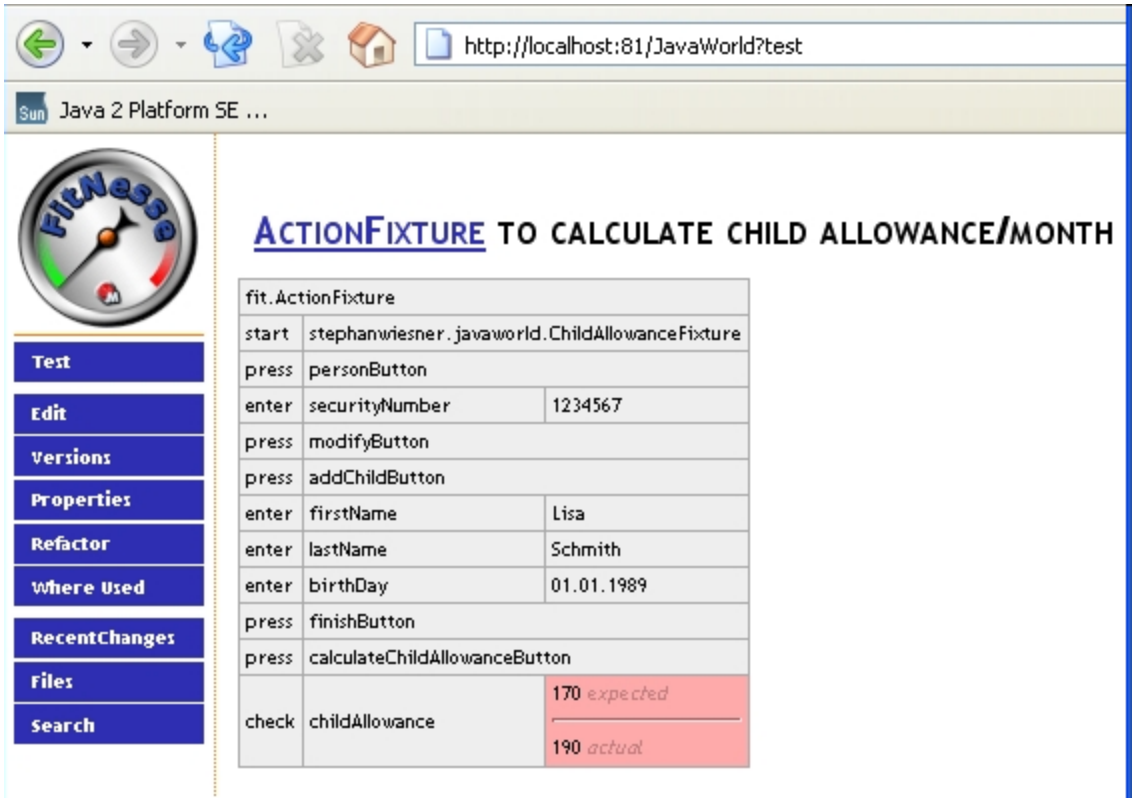
Highspeed Test Automation

Keyword-driven testing

Object	Action	Data
Textfield(username)	Enter Text	„Max“
Textfield(password)	Enter Text	„Secret“
Button(login)	Click	One left click

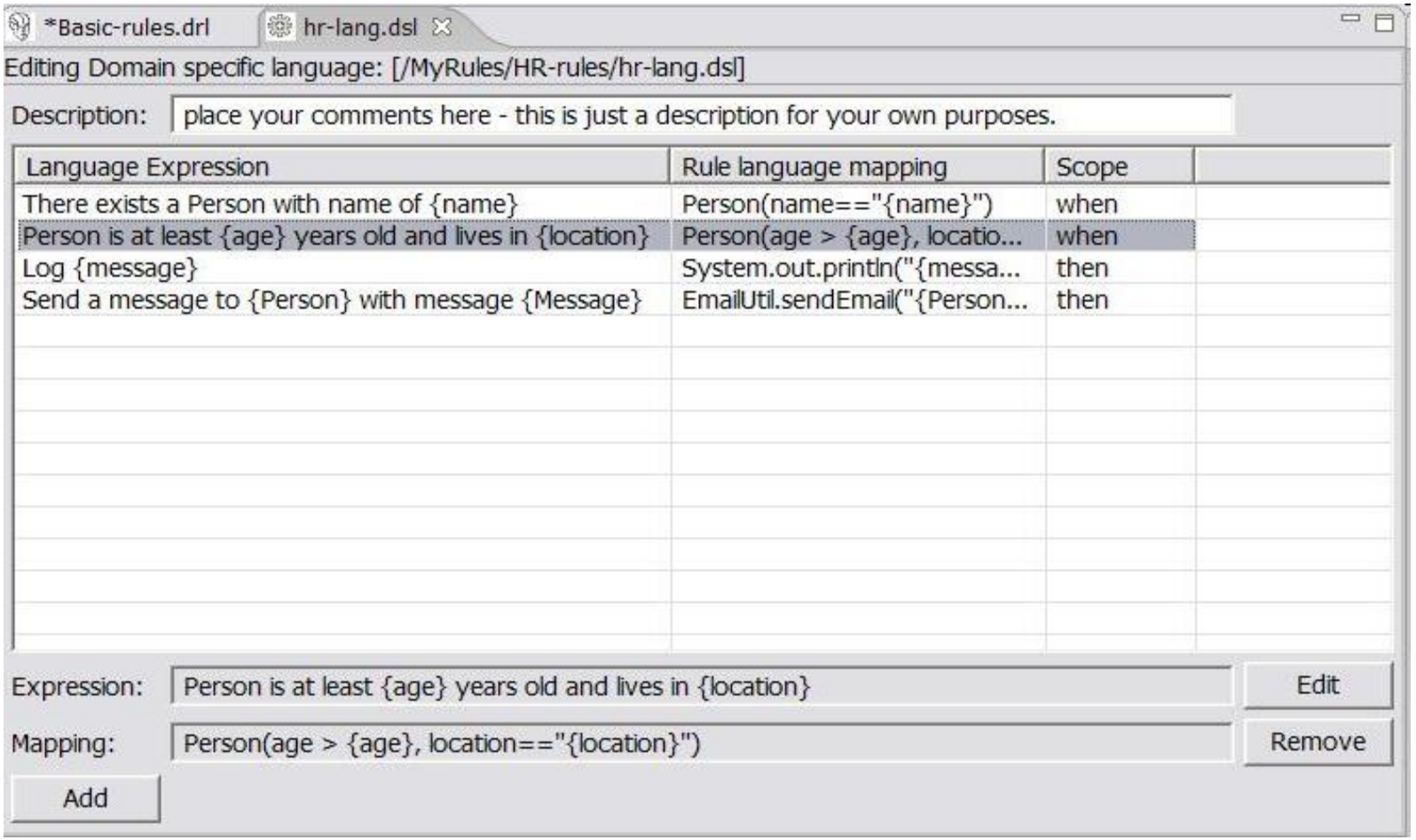
Highspeed Test Automatision

Keyword-driven testing



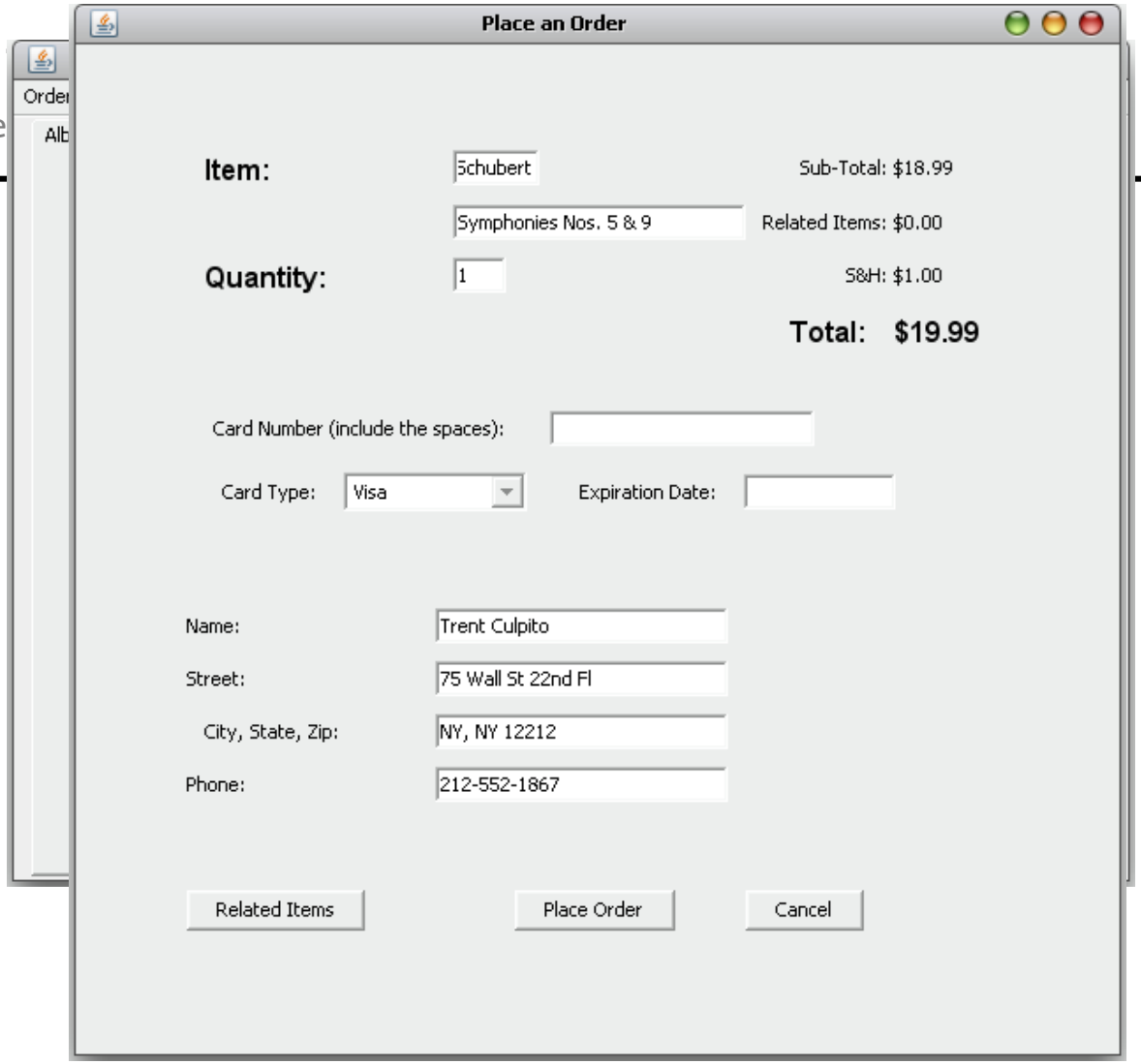
Highspeed Test Automation

Domain Specific Languages (DSLs)



Highspeed

DSL Example



Example

```
1.
2.
3.
4.
5.
6.
7.
8.
9.
```

```
34 1234",
.
al}).
).
```

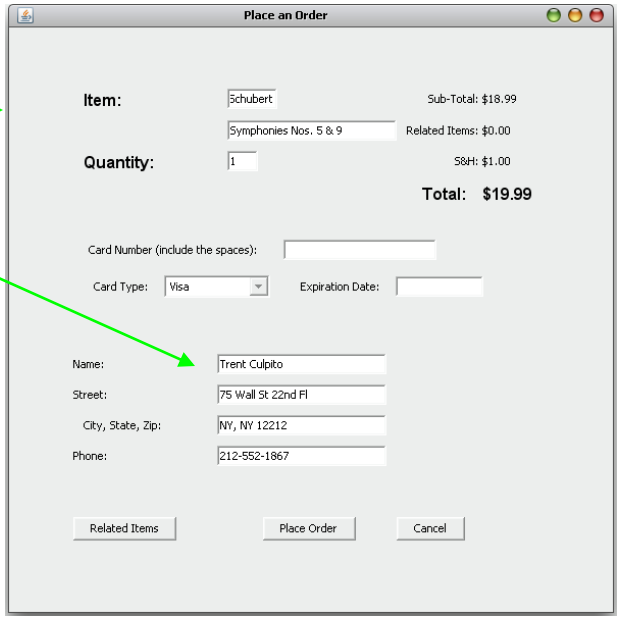
```
21    public void testMain(Object[] args) throws IOException, BSFException
22    {
23        startApp("ClassicsJavaB");
24
25        // Frame: ClassicsCD
26        placeOrder().click();
27
28        // Frame: Member Logon
29        nameCombo().click(atText("Bill Wu"));
30        passwordText().click(atPoint(6,9));
31        memberLogon().inputChars("password");
32        rememberThePassword().clickToState(SELECTED);
33        ok().click();
34
35        // Frame: Place an Order
36        cardNumberIncludeTheSpacesText().click(atPoint(18,5));
37        placeAnOrder().inputKeys("1234 1234 1234 1234{TAB}");
38        creditCombo().click();
39        creditCombo().click(atText("Mastercard"));
40        expirationDateText().click(atPoint(9,10));
41        placeAnOrder().inputChars("09/09");
42        placeOrder2().click();
43
44        //
45        yourOrderHasBeenReceivedYourOrderNumberIs54().performTest(YourOrderHasBeenReceived_textVP());
46        ok2().click();
47
48        // Frame: ClassicsCD
49        jmb().click(atPath("Order"));
50        jmb().click(atPath("Order->View Existing Order Status..."));
51
52        // Frame: View Order Status
53        nameComboB().click();
54        nameComboB().click(atText("Bill Wu"));
55        passwordText2().click(atPoint(27,8));
56        viewOrderStatus().inputChars("password");
57        rememberPassword().clickToState(SELECTED);
58        ok3().click();
59
60        // Frame: View Existing Orders
61        existingTable().performTest(existingTable_contentsVP());
62        existingTable().click(atCell(atRow("ORDER ID", "11", "ORDER DATE",
63                                           "25/08/08", "STATUS",
64                                           "Order Initiated"),
65                                     atColumn("ORDER ID")),
66                              atPoint(47,11));
67        cancelSelectedOrder().click();
68        close().click();
69    }
```

How does this Testing DSL looks like?

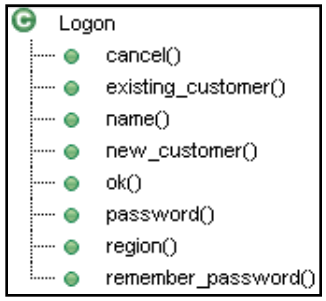
```
clerk.starts_app.
  places_order.for_customer(bill)
```

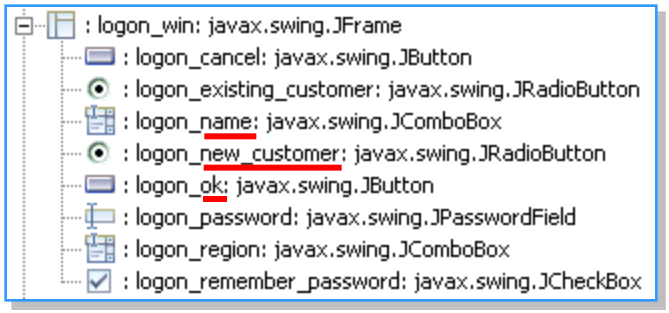
```
5  class Main < Window
6
7    def places_order
8      place_order.click
9      Logon.new
10   end
```

```
7  def for_customer(customer)
8    new = ! name.has?(customer[:name])
9    remember("customer", customer)
10
11   if new
12     new_customer.select
13   else
14     name.select(customer[:name])
15   end
16
17   ok.click
18   order = Order.new({:actor => actor})
19   order.sets(customer) if new
20   order
21 end
```



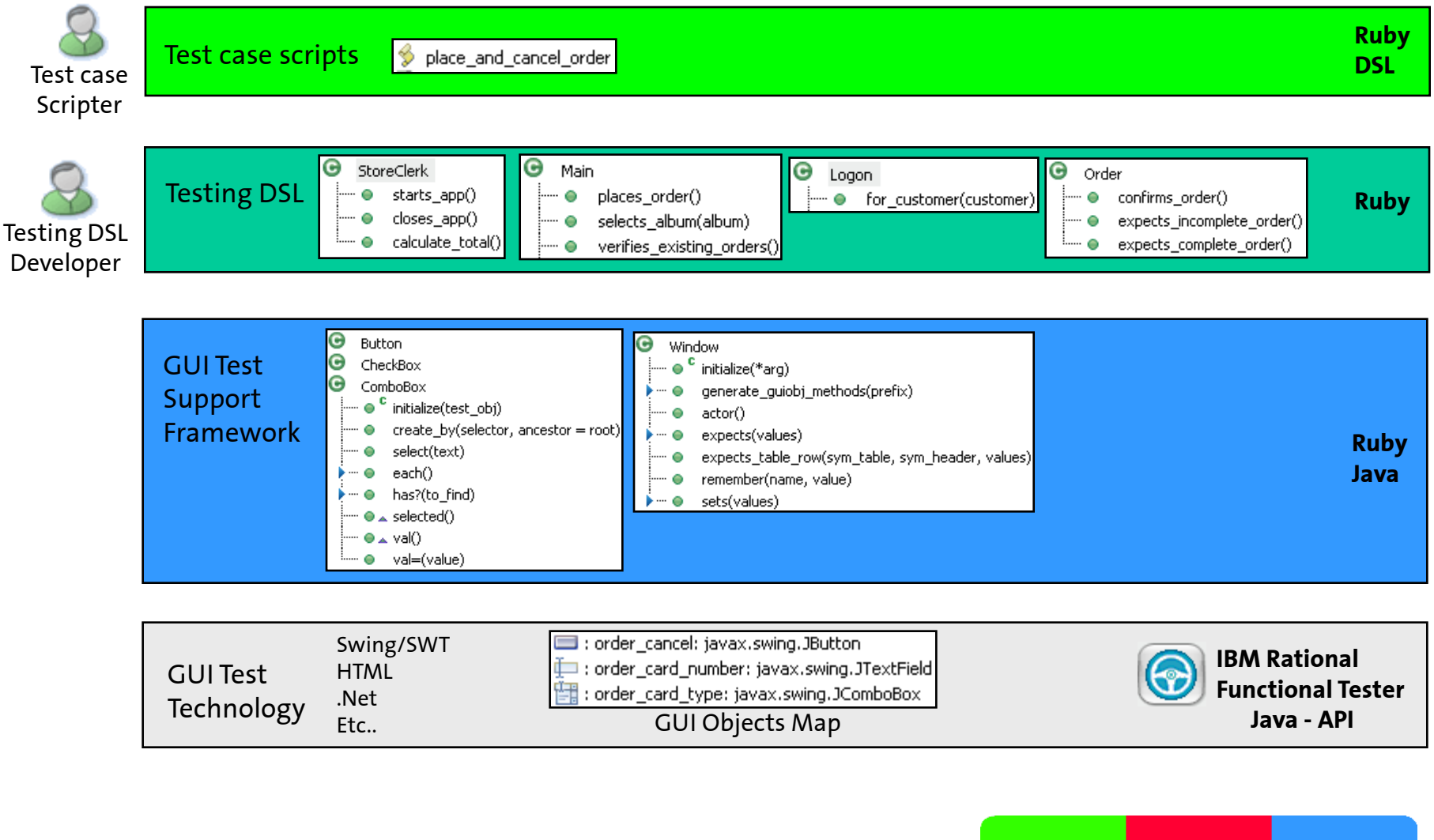
Metaprogramming





michel.loehr@saxsys.de www.saxsys.de

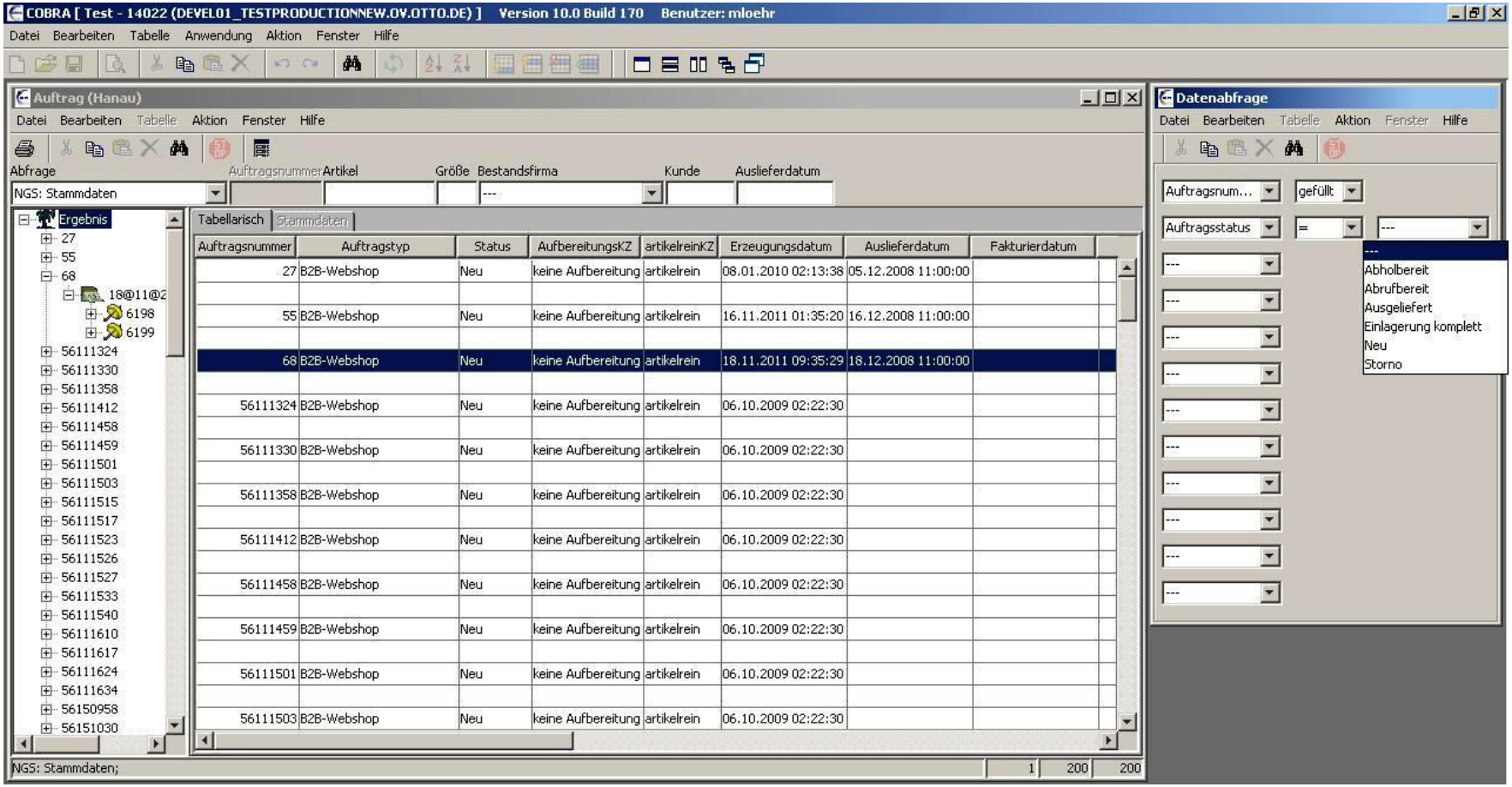
How does such an architecture look like?



michel.loehr@saxsys.de www.saxsys.de

Highspeed Test Automation

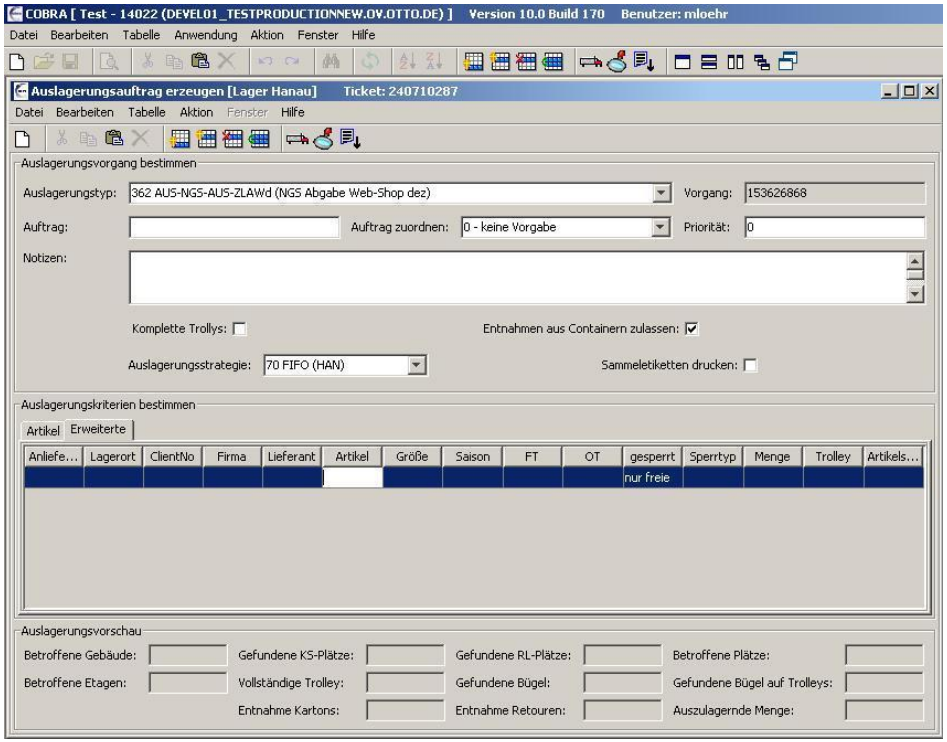
DSL Example: Inventory Management



Highspeed Test Automation

Test automation using Groovy & internal DSL

```
AuslagerungsAuftragErzeugen(lager){
    super("Auslagerungsauftrag erzeugen ."+lager+".*")
    set_fixtures {
      auslagerungstyp        : {comboBox(name: "baseCom
      auftragszuordnung      : {comboBox(name: "baseCom
      auftrag                : {textbox(name:"jTextFiel
      vorgang                : {textbox(leftlabel:  "Vo
      fachentnahmen          : {checkBox(name: "jCheckB
      auslagerungskriterien  : {tabbedPane()},
      auslagerbare_menge     : {textbox(name:"jTextFiel
      artikel_kriterien      : {table("jTableItemData")
      ...
    }
}
```



Highspeed Test-Automation

DSL Example: Groovy Testscript snippets

```
def artikeln = [[artikel:6199, größe:38, menge: 12, trolli: [10, 2]],
                [artikel:6198, größe:38, menge: 12, trolli: [10, 2]]
               ]
vorgang = erzeugeAuslagerung([auslagerungstyp    : "362 .*",
                              auftrag            : auftragNo,
                              auftragszuordnung  : "1 - Auftrag zuordnen",
                              notizen            : "Guitest: Hanau -> Webshop",
                              sammeletiketten    : false
                             ],
                             [artikeln],
                             WLS
                             )

cobra.auslagerungsauftrag_erzeugen().with {
  erzeuge(auslagerung)
  artikeln.each { artikel ->
    artikel_kriterien([ [Artikel: artikel.artikel,
                         Größe: artikel.größe,
                         Menge: artikel.menge]
                      ])
  }

def artikel_kriterien(List arg) {
    auslagerungskriterien.selectTab("Artikel")
    artikel_kriterien.addRows(arg, 0, {it.pressAndReleaseKey(key_enter)})
}
```

Highspeed Test Automation

Conclusion using DSLs

Benefits

→Higher abstraction > better productivity in testscripting

→Better flexiblity over Keyword-driven testing (graybox)

Drawbacks

→Need some development experience of (host language) to create testscripts

→Technology dependencies

Highspeed Test Automatisierung

Vision: data-, keyword-, model driven testing

```
Testplan: sample # 1
    login user: „foo“, password „bar“
    buy_cds datapool[„cd-pool1“]

Scenario: login user, passwd
    login_screen {
                enter username:user, password:passwd
                [advanced]
                click login
    }
    alternative: advanced {
                check advanced_login
    }

Context: login_screen
    fixture: {title: „Login .*“}
    guimap: [username, password: textbox,
                advanced: checkbox,
                login: button]
```

```
Testplan: sample # 2
    login user: „foo“, password „bar“ using advanced
```

michel.loehr@saxsys.de www.saxsys.de

Thank you for your attention!

```groovy
static String takeScreenShot(String name = "") {

    def screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize()
    def rect = new java.awt.Rectangle(0, 0, screenSize.width.toInteger(), screenSize.height.toInteger())
    def image = new java.awt.Robot().createScreenCapture(rect)

    if (!log.dir.exists()) {
        log.dir.mkdir()
    }

    //Save the screenshot as a png
    def imgname = (name)?name:"screen${++log.scrn_count}"
    def imgfile = new File("${log.dir}/${imgname}.png")
    ImageIO.write(image, "png", imgfile)

    def htmlfile = new FileWriter("${log.dir}/${imgname}.html")
    def x = new groovy.xml.MarkupBuilder(htmlfile)
    x.html{
      body{
          img(src:"${imgname}.png")
      }
    }
    imgname
}
```

```
def set_fields(Map args) {
    args.each { key, value ->
        def comp = fixtures[key.toLowerCase()]()
        switch (comp) {
          case JCheckBoxFixture:                                    ıcheck() ;  break
                if (Eval.me("$value") && !comp.isSelected()) {      :omp.selectItem(0) ;  break
                  comp.check()                                      :xt() ; break
                } else {
                  if (!Eval.me("$value") && comp.isSelected()) {
                      comp.uncheck()
                    }
                }
                break
          case JComboBoxFixture:
                comp.select(value.toString())
                break
          case JTextComponentFixture:
                comp.setText(value.toString())
                break
        }
    }
}
```

```
def methodMissing(String name, args) {
    if (name.startsWith("fenster_")) {
        def action = name.replaceFirst("fenster_","")
        switch (action) {
            case ~/bewege.*/   : name = "moveTo"
                                 args = new Point(args[0], args[1]) ; break
            case ~/breite.*/   : name = "resizeWidthTo"   ; break
            case "groesse"     :
            case ~/größe.*/    : name = "resizeTo"
                                 args = new Dimension(args[0], args[1]) ; break
            case "herstellen" : name = "normalize"       ; break
            case ~/hoehe.*/    :
            case ~/höhe.*/     : name = "resizeHeightTo" ; break
            case "maximieren" : name = "maximize"        ; break
            case "minimieren" : name = "iconify"         ; break
            case "schliessen" :
            case "schließen"   : name = "close"          ; break
            default: fail("unknown action: "+action)
        }
    }
    fixture.invokeMethod(name, args)
    this
}
```

```
Closure addRows = { delegate, rows, col_offset, afterRow, customEditors ->
    rows.each { row ->
        int    lastrow = delegate.rowCount() - 1
        String description = ""

        // when key '_order' is present take this to be the order to set the cells:
        def order = (row["_order"]) ?: row.keySet()
        order.each { entry ->
            def value = row[entry]

            description += "$entry=$value; "
            def col_entry = -1
            delegate.component().getColumnModel().getColumns().eachWithIndex { header, i ->
                if (header.headerValues.grep(entry)) {
                    col_entry = i
                }
            }
            if (col_entry < 1) {
                throw new Exception("Column $e
            }
            def cell = new TableCell(lastrow,
            def cf = new JTableCellFixture(del
            def editor = cf.editor()
            def done = false
            customEditors.each { type, edit ->
                if (editor.class.name =~ ".*${
                    edit(delegate, cell, value
                    done = true
                }
            }
            if (!done) {
                delegate.enterValue(cell, valu
            }
        }
        afterRow.call(delegate)
    }
}
JTableFixture.metaClass.addRows = {rows -> addRows(delegate, rows, 0, {}, [:]) }
JTableFixture.metaClass.addRows = {rows, col_offset, afterRow ->
        addRows(delegate, rows, col_offset, afterRow, [:]) }
JTableFixture.metaClass.addRows = {rows, col_offset, afterRow, customEditors ->
addRows(delegate, rows, col_offset, afterRow, customEditors) }
```

```
Closure tabLastCol = { delegate ->
    def row = delegate.component().getSelectedRow()
    def col = delegate.component().getColumnCount()-1

    def cell = new TableCell(row, col)
    delegate.selectCell(cell)
    delegate.pressAndReleaseKey(key_tab)
}

Closure comboBoxEditor = { delegate, cell, value ->
    delegate.selectCell(cell)
    // be sure to get only the codenumber
    def code = value.toString() =~ /(\d+)/
    // press every num key of the code:
    code[0][1].getChars().each { key ->
        delegate.pressAndReleaseKey(keyCode((key as int)))
    }
}
```

```
telnet = new Teln
telnet.connect(ho
rin = telnet.getI
rout = telnet.get

setupGUI("Scanner

server = new XMLR
server.isReady =
  def isReady = r
  ready = false
  return isReady
}

socket = new Serv
server.startServe
telnet()
```

```
static def telnet() {

    def writer = Thread.start {
        BufferedReader br = new BufferedReader(new
            InputStreamReader(telnet.getInputStream(), "ISO-8859-1"))

        char[] buff = new byte[1024]
        int ret_read
        while (  (ret_read = br.read(buff,0,1023)) )
        {
          if (ret_read > 0) {
            def output = new String(buff, 0, ret_read)

            def parts = output.split(/[\x1b]\x5b\x32\x4a\x1b\x5b\x48/)
            parts.each { part ->
              if (part.size() > 2) {
                text.text = ""
                println "-----EOS-----"
              }
              text.append(part)
              println "${part}"
            }
            // mark output as ready
            ready = true
          }
        }
    }
}

remote = new XMLRPCServerProxy("http://localhost:8008/")

while (!remote.isReady()) {
  sleep(200)
}
```