iSAQB Advanced DSL - Motivation

Michael Sperber

Created: 2024-06-30 Sun 19:14

Michael Sperber

- CEO, Active Group GmbH, Tübingen
- researcher on programming languages 1995-2003
- PhD in 2001, DSL for stage lighting
- designed DSL in the late 1980s
- functional programming / functional software architecture

Foundation Level

CPSA Certified Professional for Software Architecture®

International Software Architecture
Qualification Board

iSAQB

International Software Architecture Qualification Board

"The purpose of the association is to standardise the training of software architects internationally."

- founded in 2008
- develops standard curricula
- foundation level
- advanced level
- product and technology-neutral

CPSA Advanced Level

- design medium-sized to large IT systems independently and in a methodologically sound manner,
- assume technical and content-related responsibility within IT systems of medium to high criticality,
- plan, design and document appropriate measures to meet non-functional requirements,
- accompany development teams in the process of implementing these measures, and
- manage and direct architecture-related communication processes within medium-sized to large development teams.

CPSA Advanced Level Certification

- Methodology, Technology Communication
- need 70 credit points, 10 credit points each
- exam is take-home architecture exercise, ~40hours

DSL

"A data structure is just a stupid programming language."

– Bill Gosper

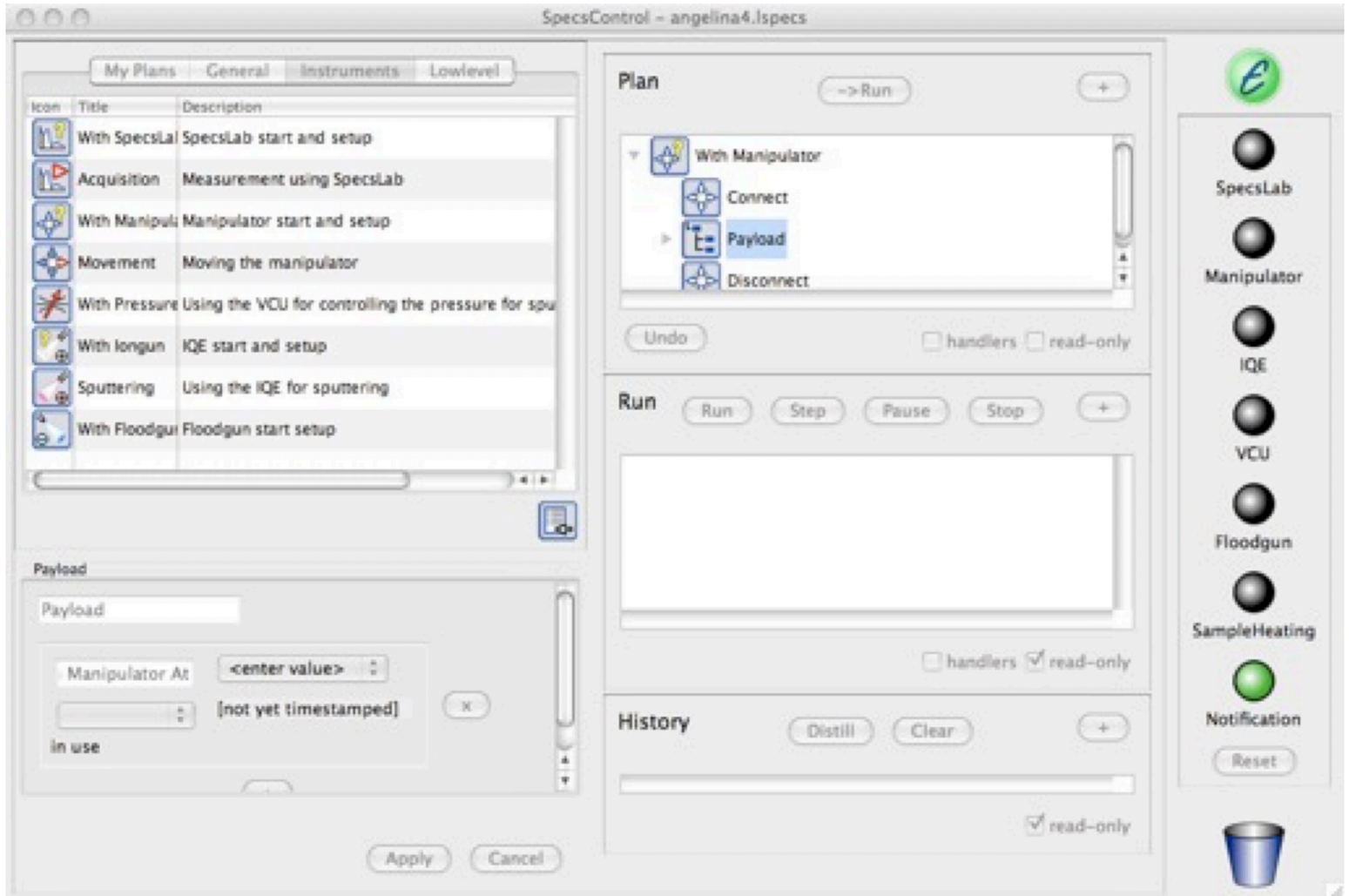
DSL Examples

- SQL
- HTML
- CSS
- PostScript
- PHP
- YAML
- Emacs Lisp
- LaTeX
- VBA
- DOT
- PlantUML
- Gherkin
- BPMN
- Frink
- …

Example: AUTOSAR

```
<SWC-TO-ECU-MAPPING
   UUID="dc6b203d-f3b8-48a1-b1bb-ebe611639946”>
   <SHORT-NAME>abd8b270ad2074978a759aa82135687</SHORT-NAME>
    <COMPONENT-IREFS>
     <COMPONENT-IREF>
       <SOFTWARE-COMPOSITION-REF DEST="SOFTWARE-COMPOSITION">
         /TutorialProject/SYS_Indicator/SWA_Indicator
        </SOFTWARE-COMPOSITION-REF>
        <TARGET-COMPONENT-PROTOTYPE-REF
         DEST="COMPONENT-PROTOTYPE">
          /TutorialProject/SWA_Indicator/FrontLeftActuator
       </TARGET-COMPONENT-PROTOTYPE-REF>
      </COMPONENT-IREF>
  </COMPONENT-IREFS>
  <ECU-INSTANCE-REF DEST="ECU-INSTANCE">
    /TutorialProject/HWT_Indicator/FrontLeftIndicatorEcu
  </ECU-INSTANCE-REF>
</SWC-TO-ECU-MAPPING>
```

Example: Surface Analysis Process DSL



Example: Emacs

```
(defun gnus-summary-save-article-spam (&optional n)
  "Append the current article to the SPAM file.
If N is a positive number, save the N next articles.
If N is a negative number, save the N previous articles.
If N is nil and any articles have been marked with the process mark
save those articles instead."
  (interactive "P")
  (let ((articles (gnus-summary-work-articles n)))
    (if (and gnus-novice-user
             (not (gnus-yes-or-no-p
                   (format "Do you really want to delete %s forever
                           (if (> (length articles) 1)
                               (format "these %s articles" (length
                             "this article")))))
        ()
      (gnus-summary-save-articles-spam articles)
      (gnus-summary-delete-articles articles))
    n))
```

Example: Questionnaire DSL

```
(defn verstoesse-editor
   [verstoesse date]
   (let [edit-verstoss (edit-verstoss-x verstoesse date)
         verstoesse-map (apply conj {} verstoesse)]
     (elements/within :verstoesse
       (lists/list-editor
        (lists/list-item-editor
          (lists/with-add-button-editor "28px" edit-verstoss))
        (lists/list-item (lists/list-column (loc/L "Gesetzesverstos

                                            (lists/list-item-field
                                             (elements/value-of (lens/++ (lens/pos
                                                                          (fn [[id sonstiges]
                                                                            (dom/div
                                                                             (if (= id sonsti
                                                                               sonstiges
                                                                               (utils/spec op
                                                                (lists/list-column txt/Datum
                                                                                   (lists/list-item-field
                                                                                   :width "20%")
                                                                bearbeiten-column
                                                                löschen-column
```

(LG 4-1) Example: Factory Automation

```
PROC SEND_SPA (SHORT _a, SHORT _m, SHORT _id, LONG _id_no)
#*************************************************************
#  Functional Description :
#      Handles SPA init aministration.
#-------------------------------------------------------------
#  Called from :
#      Triggered by spa_init_trig
#      This trigger is set from counter enable_spa_init
#-------------------------------------------------------------

BEGIN

DECLARE LONG   _old_time
DECLARE SHORT  _busy
DECLARE SHORT  _time_out

  # Check if channel is available, check spa_ssad_snd_msg_state for
  # Yes: SSAD sets this tag to OFF, when a send table is busy!
  _old_time = UTC_SECTIME
  _busy = 1
  WHILE _busy
    # Loop as long the channel is in use for a maximum of 2 seconds
```

Exercise: Requirements and Conditions

Under what requirements and conditions does a DSL make sense?

(LG 1-1) Condition: Insufficiently Expressive Implementation Language

```sql
SELECT
  first_name,
  last_name,
  salary
FROM employees
WHERE salary > 3800;
```

(LG 1-1) Requirement: Complex User-Defined Rules

```
blah: blah.o
        cc blah.o -o blah # Runs third

blah.o: blah.c
        cc -c blah.c -o blah.o # Runs second

# Typically blah.c would already exist, but I want to limit any add
blah.c:
        echo "int main() { return 0; }" > blah.c # Runs first
```

(LG 1-1) Requirement: Restrict Behavior

```
%PDF-1.7
7 0 obj
<<
/Type /XObject
/Subtype /Form
/Resources <<
/ProcSet [ /PDF /Text /ImageB /ImageC /ImageI ]
/ExtGState <<
/G3 4 0 R
/G12 8 0 R
/G13 11 0 R
/G14 12 0 R
>>
/XObject <<
/X7 17 0 R
>>
>>
/BBox [ 0 0 543 251 ]
/Group <<
/Type /Group
/S /Transparency
/I true
```

(LG 1-1) Requirement: Analyzability

```
machine todoItem {
  state idle {
    delete => deleting
  }
  state deleting {
    invoke deleteTodo {
      done => deleted
    }
  }
  final state deleted {}
}

machine app {
  state idle {
    new => assign(todo, spawn(todoItem))
    delete => send(todo, delete)
  }
}
```

Matthew Phillips: Announcing Lucy

(LG 1-1) Exercise: Consequences

Can you describe example projects where the introduction of a DSL had far-reaching consequences?

Answer (Good)

The company LexiFi offers a DSL for describing complex financial products.

Products described using the DSL replace millions of lines of code modelling only limited aspects of such products. Using the DSL ensures the consistency of the views of multiple department on a given product.

Answer (Bad)

FactoryLink was an infrastructure for production-process automation.

Users could describe processes using a BASIC-like DSL, where events in the factory would trigger BASIC routines. Writing these routines was cumbersome because the only interaction with the outside world was through a "Real-Time Database", which mirrored sensors and actuators.

Processes that spanned multiple events had to save their state in the RTDB, often leading to BASIC routines with many numbered entry points.

One particular customer had used FactoryLink to automate a semiconductor factory, amassing 6 millon LOCs.

(LG 1-1) Possible Disadvantages

- vendor lock-in
- implementation effort
- only maintainable by few
- frustrating usability

(LG 3-1) Exercises: Specification of DSLs

What notations and mechanisms do you know for expressing the specification of a language?