Programming-in-the-many: SLIME

Summer 2002

Karsten Stahl Martin Steffen

Institut. für Informatik u. Prakt. Mathematik

Christian-Albrechts Universtität zu Kiel

Slime

Sequential Function Charts Modeling Environment

- SFC
  - one of various description languages for micro controllers
  - international standard (IEC 61131)
  - Petri-net like semantics
  - here: “poor man’s SFCs”: simplified, but with formal operational semantics

Results

- runnable tool, all modules integrated, executable under jdk-1.4
  - graphical interface for editing
  - checks (type checking, well-formed checking)
  - parser
  - simulator
- CD-Rom with jar'ed tool (+ doc + sources + repos . . .)

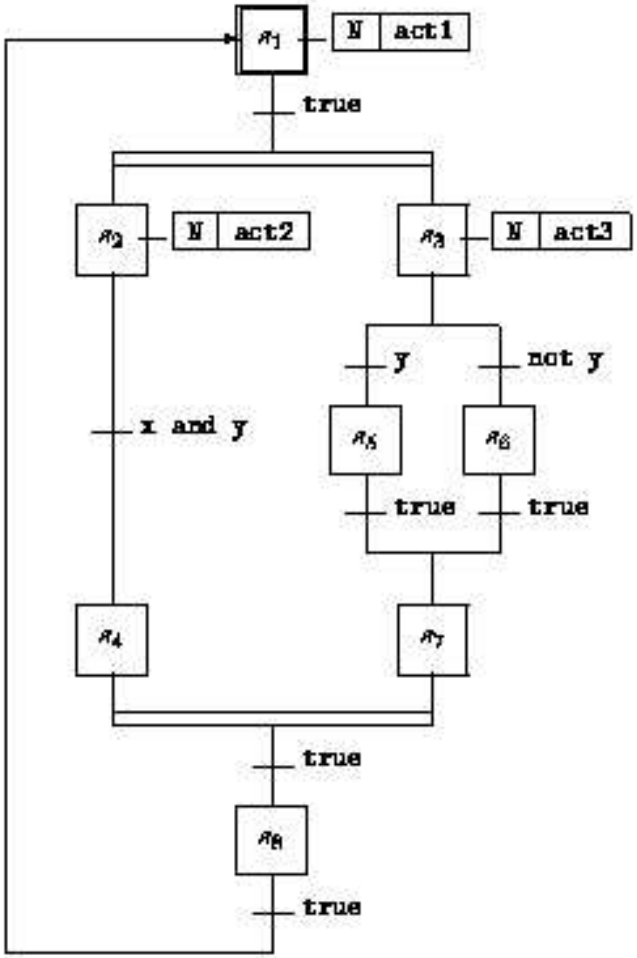
SFC example

Deklarationen

x	bool	false
y	bool	false
z	bool	false

Aktionen

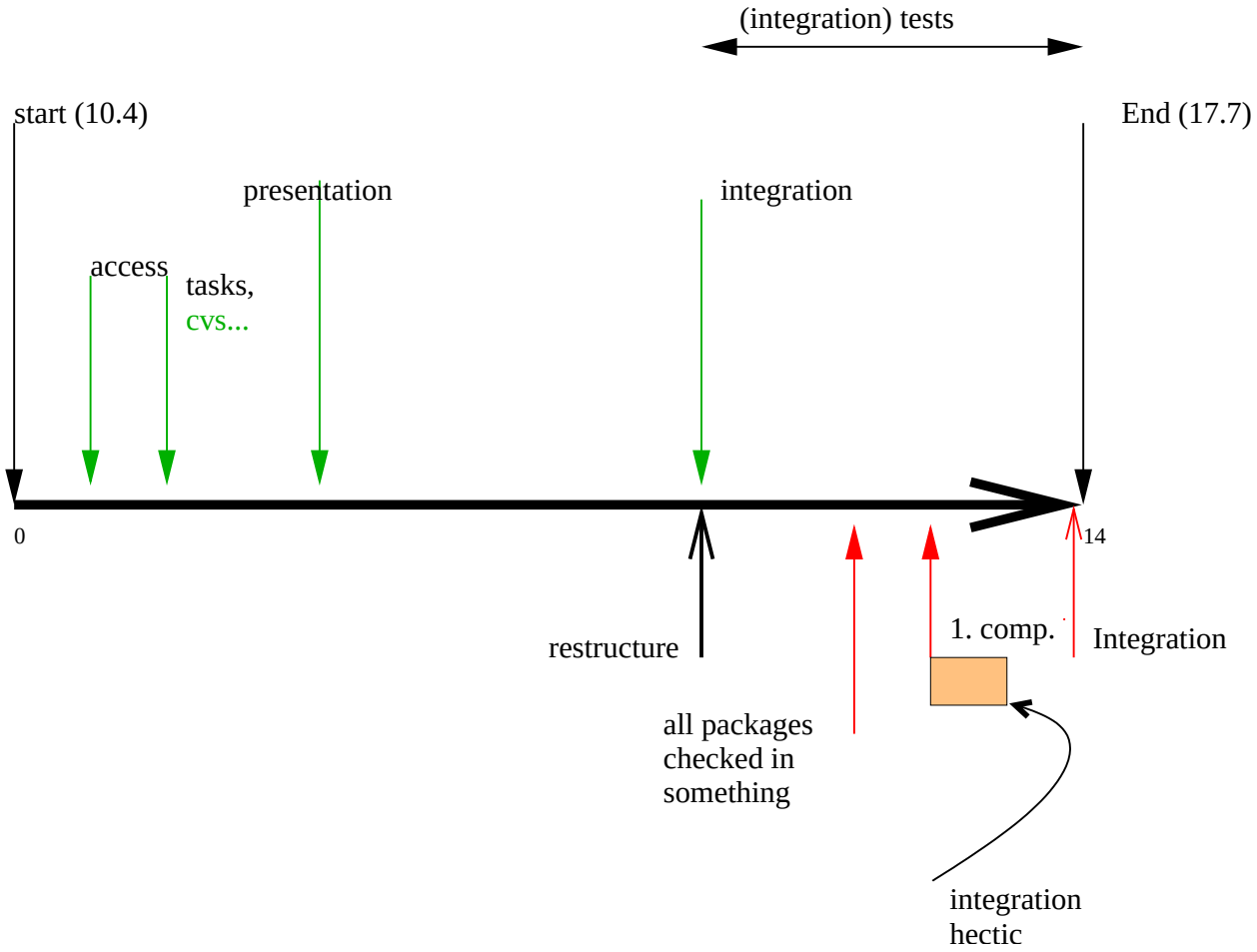
act1	x := false
act2	y := x
act3	x := not x; y := x



Develoment process

- CVS, modules as packages
- Error-list, Status list
- email-list
- public web-page including JAVADOC documentation
- weekly progress report
- 3 review meetings, including this one.

Timeline (planned and actual)



Error reporting

```
-------------------------------------------------------
 Error <nr>: <short description>
   package:  <in which package/class does it occur
   status:   reported|confirmed|non-confirmed|repaired
             repaired-confirmed

            + <date> + <author>

   class:    fatal|non-fatal|
             feature-request|coding convention violation ....

   description: <longer description, hints for repair>
  -----------------------------------------------------
```

Statistics

- 13 official meetings
- 4 iterations of the requirement specification
- > 500 emails concerning SLIME in my mailbox [a]
- approximately
  - 100 officially reported errors [b]
  - 170 Java files
  - 200 class files, i.e. 200 public classes
  - 50 LaTeX-files (doc, web-pages, requirements)
  - handfull of other files (Makefiles, Error lists etc.)

[a] including those exchanged directly with the participants, but without the more than 700 cvs-log emails.

[b] none confi rmed . . .

Good

- it’s over
- we have a running tool ready
- nice result for so few people
- task distribution
- good specification: formal operational semantics

Neutral/beyond our control

- not much people,
- lot of (late) drop outs, [a] and lately announced

[a] people at the beginning: 11 (except coaches), at the end: 4

Less good

- Attracting students
  - another topic?
  - stressing collaborative work over programming in JAVA?
- laaaate first code delivery (26. June) /compilation, laaate integration (with all the consequences)
- we always had quite some breaches of interfaces, but: this year was the first time, I had to discuss why this is should be avoided without much discussion
- communication
- no test group, no Error ever `confirmed`

Next time

- first *Readme* or first written plan required to be checked-in in after 2 weeks
- stricter, enforced cvs-strategy?: enforced compilability for checking-in?
- user logging (currently, I don’t know how, the official university’s server can do it, but there are other disadvantages of that solution)?
- stricter surveillance (e.g. for absynt), watches
- no separation between gui and editor? But an explicit test group.
- other means of communication? (*news-group?*, cvs-logs?)