Eiffel Standardization

Dipl. Inform. Arno Wagner
Computer Engineering and Networks Laboratory, ETH Zürich

Overview

- The Eiffel Library Kernel Standard (ELKS)
- How eiffel-nice-library works
- The specification style
- Examples
- Issues and concerns
- Remarks

The Eiffel Library Kernel Standard (ELKS)

A Standard by NICE (Nonprofit International Consortium for Eiffel) to improve interoperability

- Specifies required classes
- Specifies required ancestry links
- Is periodically revised
- Current complete Standard: ELKS'95
- Finished revision: ELKS 2000 ARRAY
- Current work: ELKS 2001 STRING

The Standardization Process

Discussion on eiffel-nice-library

⇓

Vote on eiffel-nice-library by NICE members

⇓

Proposal to NICE Board

⇓

Approval by NICE Board

⇓

Standard

eiffel-nice-library

The mailing list for discussion about ELKS

- Open mailing list, currently 96 members
- Active members with various background

Typical discussion process

- Moderator (Roger Browne) starts discussion about a feature
- Discussion produces one or more options / specifications
- Moderator sums up and calls vote
- Vote by email, open 3 days

Typically values: 8 - 15 voters, 3 - 14 days discussion time

The Specification Style

Classes have

- Name
- Head comment
- Set of features
- Set of invariants

Features have

- Name
- Signature
- Head comment
- Precondition (optional)
- Postcondition (optional)

Details of the Specification Style

- Uses a mimimal set of 'basic specifiers'
- Basic specifiers have no postconditions
- Other features specified relative to the
- Assertions completely specify a feature (where possible)
- Recursive postconditions for recursive specifications
- Alternative specification methods only if necessary
- All assertions are named

Recursive Specification: Example

```
has (c: CHARACTER): BOOLEAN
   -- Does Current contain c?
   ensure
      false_if_empty:  count = 0 implies not Result
      true_if_first:   count > 0 and then item (1) = c implies Result
      recurse:         (count > 0 and then item (1) /= c) implies
                       (Result = substring (2, count).has (c))
```

Recursive Specification: Properties

- High expressive power
- Allow complete specifications in amy cases
- Reasonable readability
- Executable
- Not completely executable
- Sometimes very complex
- No full first-order power (no 'binding')

Other Specification Methods Used: Example

```
is_integer: BOOLEAN
   -- Does 'current' represent an INTEGER?
   ensure
      syntax_and_range:
       -- 'result' is true if and only if the following two conditions
       --  are satisfied:
       --
       -- 1. In the following BNF grammar, the value of
       --    'Current' can be produced by "Integer_literal":
       --
       -- Integer_literal = [Sign] Integer
       -- Sign            = "+" | "-"
       -- Integer         = Digit | Digit Integer
       -- Digit           = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
       --
       -- 2. ... within range ...
```

Issues and Concerns

Vendor acceptance:

- Incompatibilities with existing libraries
- Incompatibilities with older ELKS
- Features that may be difficult to implement

Usability:

- Names: Intuitive and consistent naming needed
- Completeness of feature set
- Feature bloat: What to include and what not?
- Readability of the specification
- ...

Remarks

- eiffel-nice-library works quite well
  - The moderator is critical
  - The discussions are interessting
  - Contributions have a high standard
- Complete assertion based specifications are often feasable and apropriate
- Completely executable assertion based specifications are sometimes infeasable
- ELKS gained considerably in exactness, simplifying implementation