Forth System Hooks
Metaobject Protocol in Forth Style

Ulrich (Ulli) Hoffmann
uh@fh-wedel.de

EuroForth 2008, 2008-09-26

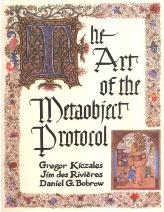
Overview

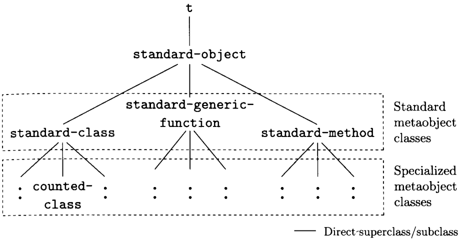
1 Motivation

2 Standard Hooks?

3 Summary

Meta Object Protocol





t

standard-object

standard-class

standard-generic-function

standard-method

Standard metaobject classes

counted-class

Specialized metaobject classes

Direct-superclass/subclass

- Describes the Idea of a *meta object protocol*
- Lisp centric, for CLOS (Common Lisp Object System)

```
(defmethod make-instance ((class standard-class) &rest initargs)
  (let ((instance (allocate-instance class)))
    (apply #'initialize-instance instance initargs)
    instance))
```

Hooks in EMACS

- Hooks are very often used in EMACS and its extensions.

EMACS — diff.el

```
(defun diff-delete-if-empty ()
  ;; An empty diff file means there’s no more diffs to integrate, so we
  ;; can just remove the file altogether.  Very handy for .rej files if we
  ;; remove hunks as we apply them.
  (when (and buffer-file-name
     (eq 0 (nth 7 (file-attributes buffer-file-name))))
    (delete-file buffer-file-name)))

(defun diff-delete-empty-files ()
  "Arrange for empty diff files to be removed."
  (add-hook ’after-save-hook ’diff-delete-if-empty nil t))
```

Hooks in some Forth implementations

- Many Forth systems already have hooks for one purpose or the other.
- They are system specific.

volksForth

```
\ .status push load                            ks 29 oct 86

  Defer .status   ' noop Is .status

  : (load  ( blk offset -- )   isfile@ >r
     loadfile @ >r   fromfile @ >r   blk @ >r   >in @ >r
     >in !   blk !  isfile@ loadfile !  .status  interpret
     r> >in !   r> blk !   r> fromfile !   r> loadfile !
     r> isfile ! ;
```

Hooks in some Forth implementations

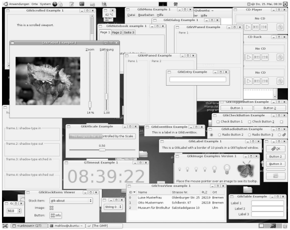
- Many Forth systems already have hooks for one purpose or the other.
- They are system specific.

Open Firmware

```
headers
defer fm-hook  ( adr len phandle -- adr len phandle )
' noop is fm-hook

: find-method  ( adr len phandle -- false | acf true )
   fm-hook  (search-wordlist)
;
```

Manfred Mahlow's cspForth



GTK+ binding for cspForth

- Very elegant object oriented extension to Standard Forth
- `http://www.forth-ev.de/wiki/doku.php/en:projects:csp4th:cspforth`
- Words can have a `prelude`–action which is executed everytime the word is interpreted.
- Requires modifications (=hooks) in `create` and `interpret`.

Where to put Hooks?

- *Boot Hooks*
  Define what action a system is eventually doing when booting.
- *IO Hooks*
  Define how system IO is done. Often its a collection of *Defer*ed words, sometimes an array of vectors.
- *Create Hooks*
  Define what can happen, among standard things, when creating words.
- *Interpreter Hooks*
  How the interpreter works.

Create Hooks

What happens additionaly to creating a word?

- Lay down `view`-information
- Room for statistical data about the definition
- Additional semantics to be associated with a word

```
Hook preCreate ( c-addr len -- )   ' 2drop is preCreate
Hook postCreate ( c-addr len -- )  ' 2drop is postCreate

: "Create ( c-addr len -- )
    2dup preCreate   2dup 2>R
    %create
    2r> postCreate ;
```

A notfound Interpreter Hook

- Hook notfound (c-addr len -- i*x)
- Default behavior (similar to):
  `: complaint ( c-addr len -- ) type ."?" abort ;`

```
: interpret ( -- )
  BEGIN
    BL WORD  DUP COUNT DUP C@
  WHILE ( c-addr )
    FIND ?DUP
    IF  OVER STATE @ 0<> = IF  COMPILE,  ELSE  EXECUTE THEN
    ELSE
      COUNT NUMBER? ?DUP  IF  0< IF  ?literal  THEN  ?literal
      ELSE
        notfound
      THEN
    THEN
  REPEAT
  DROP ;
```

An extension which uses notfound

- Hex number input using $

```
: hexnum? ( c-addr len -- n -1 | c-addr len 0 )
   dup 0= IF 0 EXIT THEN
   over c@ [char] $ = IF
        base @ >r hex
        2dup  0 0 2swap >number swap drop
        r> base !
        IF 2drop 0 EXIT THEN
        2swap 2drop  d>s  -1 EXIT THEN
   0 ;

: ?hexnum ( c-addr len -- n | )
   hexnum? IF EXIT THEN
   complaint ;

' ?hexnum is notfound
```

Lots of Interpreter Hooks

- What if we take this to the extreme?

```
: Hook Defer ;

Hook prompt    ( -- )
Hook status    ( -- )
Hook number    ( c-addr u -- n 1 | d -1 | c-addr u 0 )
Hook notfound  ( c-addr u -- )
Hook error     ( x -- )
Hook prolog    ( xt -- )
Hook epilog    ( xt -- )
Hook filter    ( c-addr len -- )
```

Lots of Interpreter Hooks

```
: interpreter ( -- )
  BEGIN status
        ['] interpret CATCH
        ?DUP IF  error  ELSE  prompt  THEN
        REFILL 0=
  UNTIL ;
```

Lots of Interpreter Hooks

```
: ?literal ( n -- )  STATE @ IF  POSTPONE LITERAL  THEN ;

: interpret ( -- )
  BEGIN
    BL WORD  DUP COUNT filter DUP C@
  WHILE ( c-addr )
    FIND ?DUP
    IF  OVER DUP >R prolog
        STATE @ 0<> = IF  COMPILE,  ELSE  EXECUTE THEN
        R> epilog
    ELSE
      COUNT number ?DUP  IF  0< IF  ?literal  THEN  ?literal
      ELSE
        notfound
      THEN
    THEN
  REPEAT
  DROP ;
```

Zusammenfassung

- Forth System Hooks have been used for a long time.
- They are valuable and offer another dimension of extensions.
- To be more useful we should agree on (some of) them.
- Obviously we have to find the right mixture.

Zusammenfassung

- Forth System Hooks have been used for a long time.
- They are valuable and offer another dimension of extensions.
- To be more useful we should agree on (some of) them.
- Obviously we have to find the right mixture.

- Discussion