SOM – Present and Future

Simon Nash
IBM Austin

Pages 220-235

SOM - Present and Future

Simon C. Nash

IBM Corporation
Austin, TX

nash@austin.ibm.com

What is SOM?

- ★ System Object Model
- ★ Part of the OS
- ★ Language-neutral
- ★ Language bindings (toolkit)
- ★ Compiler support (DTS)
- ★ Distributed objects (DSOM, CORBA)

Why SOM?

- ⋆ OO language interoperability
- ⋆ Binary format for objects
- ⋆ Release-to-release binary compatibility
- ⋆ Procedural language access
- ⋆ Support for distribution

Platforms

Available:

OS/2, AIX, Windows, Macintosh

Announced:

MVS, AS/400

Other ports in progress

Languages

Available:

C, C++, Smalltalk

Beta:

Object REXX

Announced:

OO COBOL

SOM Releases

1992: SOM 1.0

(C, OS/2 WPS)

1993: SOMobjects 2.0

(C++, IDL, CORBA, DSOM)

1994: SOMobjects 2.1

(Warp, DTS C++)

1995: ???

SOM Components

- ★ kernel
- ★ toolkit:
  - SOM compiler, language bindings
- ★ class libraries:
  - collections
- ★ frameworks:
  - persistence, replication, events, metaclass, IR, emitter
- ★ distribution
  - DSOM (workstation and workgroup)

A SOM Example: stack.idl

```
#include <somobj.idl>

interface Stack: SOMObject
{
  void push (in SOMObject element);

  SOMObject pop ();

  long size ();

  implementation
  {
    SOMObject contents[100];
    long top;
    somDefaultInit: override;
  };
};
```

A SOM Example: stack.c

```
#include "stack.ih"

SOM_Scope void  SOMLINK push(Stack *somSelf,
            Environment *ev, SOMObject* element)
{
    StackData *somThis = StackGetData(somSelf);
    StackMethodDebug("Stack","push");

    _contents[_top++] = element;
}

SOM_Scope SOMObject*  SOMLINK pop(Stack *somSelf,
            Environment *ev)
{
    StackData *somThis = StackGetData(somSelf);
    StackMethodDebug("Stack","pop");

    return _contents[--_top];
}
```

A SOM Example: stack.c

```
SOM_Scope long  SOMLINK size(Stack *somSelf,
            Environment *ev)
{
    iiackData *somThis = StackGetData(somSelf);
    StackMethodDebug("Stack","size");

    return _top;
}

SOM_Scope void SOMLINK somDefaultInit(Stack *somSelf,
            somInitCtrl* ctrl)
{
    StackData *somThis; /* set in BeginInitializer */
    somInitCtrl globalCtrl;
    somBooleanVector myMask;
    StackMethodDebug("Stack","somDefaultInit");
    Stack_BeginInitializer_somDefaultInit;
    Stack_Init_SOMObject_somDefaultInit(somSelf, ctrl);

    _top = 0;
}
```

A SOM Example: test.c

```
#include <stdio.h>
#include <som.h>
#include "stack.h"

void main (void)
{
  Stack *stack1, *stack2;
  Environment *ev;
  SOMObject *obj1;

  stack1 = StackNew();
  stack2 = StackNew();
  ev = somGetGlobalEnvironment();
  _push(stack1,ev,stack2);
  printf("stack1 size is %li\n",_size(stack1,ev));
  obj1 = _pop(stack1,ev);
  printf("stack2 size is %li\n",_size(obj1,ev));
  _somFree(stack1);
  _somFree(stack2);
}
```

A SOM Example: test.exe

```
sc -sc stack.idl

/* code the method implementations */

sc -sh;ih stack.idl

icc test.c stack.c som.lib

test

/* output is:
     stack1 size is 1
     stack2 size is 0
*/
```

A SOM Example: stack.dll

```
sc -sc stack.idl

/* code the method implementations */

sc -sh;ih;def stack.idl

icc /Ge- stack.c som.lib stack.def

implib stack.lib stack.def

icc test.c stack.lib som.lib

test

/* output is:
     stack1 size is 1
     stack2 size is 0
*/
```

SOM Features

- ★ static compile/link time binding
  - for high performance
- ★ name lookup, programmable dispatch
  - for flexibility, dynamic languages
- ★ class and metaclass objects
- ★ multiple inheritance
- ★ transparent proxies

SOM Challenges

- ⋆ scaleability: many fine-grained objects
- ⋆ performance: approaching native C++
- ⋆ shared objects: multi-process
- ⋆ run-time footprint: class metadata
- ⋆ local/remote transparency
- ⋆ OMG services, CORBA 2
- ⋆ dynamic language support