21

Visual Animation of LOTOS using SOLVE

K. J. Turner[a], A. McClenaghan[b]*

[a]Computing Science, University of Stirling, Stirling FK9 4LA (*kjt@compsci.stirling.ac.uk*)

[b]Philips Research Labs., Redhill, Surrey RH1 5HA (*ukrmccl@prl.philips.co.uk*)

SOLVE is an object-oriented approach and toolset based on LOTOS to allow formal requirements capture and visual animation, particularly for interactive systems and digital logic design.

Classification: D.1.5, D.1.7, D.2.1, D.2.2, I.6.2

Keywords: object-oriented programming, visual programming, requirements/specifications, tools and techniques, formal languages, LOTOS (Language Of Temporal Ordering Specification)

1 Overview

The requirements phase in LOTOS-based development has received little attention previously, and was the focus of the project SPLICE I (*Specification and Prototyping for a LOTOS Interactive Customer Environment – Phase I*). Unlike traditional software engineering methods, the SOLVE approach developed by the project allows manipulation of *formal* requirements through *visual animation*. SPLICE studied requirements capture for OSI services, digital logic, neural networks and interactive systems.

SOLVE (*Specification using an Object-oriented, LOTOS-based, Visual language* [1]) uses familiar object-oriented programming concepts translated automatically into LOTOS, thus ensuring a precise basis. The visual animation requires no knowledge of LOTOS, and is suitable for customers/clients, analysts or designers/programmers. SOLVE is partly oriented towards interactive systems, but applications such as digital logic design have also been investigated. SOLVE would benefit from extensions such as more data types, inheritance, dynamic object creation and deletion, dynamic modification of communication paths, and multi-media characteristics.

2 SOLVE Approach

The key concepts in SOLVE are object orientation, interactive animation, and formal specification. SOLVE is designed for those who are not familiar with formal languages (in particular LOTOS). SOLVE allows an analyst to write a requirements specification in an intuitive language that is automatically translated into LOTOS for exploration through visual animation.

A SOLVE specification consists of a number of concurrent objects that communicate via messages. The SOLVE language deliberately avoids the algebraic feel of LOTOS. However, SOLVE descriptions can be automatically translated into LOTOS specifications. An object declaration gives its name, instance parameter sorts and method parameter sorts; an object definition describes inner details. Behaviour within any single simple object is sequential. Method definitions use the following statements:

*The designer of SOLVE and XDILL, supported by the UK Science and Engineering Research Council on SPLICE.

D. Hogrefe et al. (eds.), *Formal Description Techniques VII*

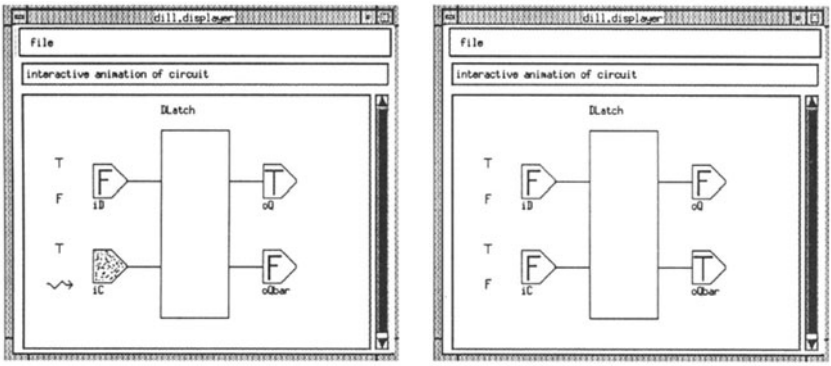
--	comments
Variables *Name : Sort, ...* **EndVariables**	local variables
Assign *(Variable, Value)*	variable bindings
If *Condition* **Then** *Statement* **Else** *Statement* **EndIf**	conditionals
While *Condition* **Do** *Statement* **EndWhile**	loops
AskWaitCall *ObjectName.MethodName (Parameter) (Results)*	blocking calls
TellCall *ObjectName.MethodName (Parameters)*	non-blocking calls

3 SOLVE In Action

SOLVE has been used, for example to define a system for setting the clock of a VCR (*Video Cassette Recorder*). This has an on-screen 24-hour clock that is set using on-screen cursor and control buttons. The cursor can be moved using left or right buttons to select one digit, which can then be adjusted with increment and decrement buttons. The mapping between informal requirements and SOLVE objects, attributes and methods is straightforward. For example, manual operation of a push-button is represented by clicking its icon. A skeleton of the SOLVE description for the VCR clock is given below; the full description is about 250 lines [1], and is translated automatically into about 1700 lines of LOTOS.

```
System VCRclock Is                                  -- system name
  PicDecls                                          -- icon declarations
    leftArrow, rightArrow                           -- icon filenames
  ObjectDeclarations                                -- object interfaces
    Object Cursor(Bool) Is                          -- cursor object, flashing parameter
      QueryXPos()(Int)                              -- check cursor x position method
      Left()()                                      -- move cursor left method
  ObjectDefinitions                                 -- object internals
    Object Cursor(Bool:flashingOn) Is               -- cursor object
      Method QueryXPos() Is                         -- return cursor x position
        Return(xPos)
      Method Left() Is                              -- move cursor left
        If (xPos Nei 1)                             -- position not 1 (i.e. leftmost)?
          Then
            Assign(xPos,xPos Minus 1)               -- decrement position
            TellCall Interface.SetIcon(xPos,yPos,iconPic)
                                                    -- asynchronous call to redisplay icon
          Else                                      -- move attempt ignored
        EndIf
        Return()
```

The SOLVE approach has also been extended to allow interactive visual animation of digital logic specifications written using DILL (*Digital Logic in LOTOS* [3]). DILL provides a library of pre-defined circuit components, combined using a macro language and LOTOS operators. XDILL (*X-based DILL* [2]) visually animates requirements for digital logic circuits. A stage in the visual animation of a D-Latch (a one-bit memory) is illustrated overleaf. In the left-hand animation window the user has clicked on the *F* (False) button to drop the input clock *iC*. The right-hand window shows that the D-Latch has stored its input data *iD*, reflected in the state of output *oQ* and inverted output *oQbar*.



The SOLVE toolset uses the X-Windows environment and is mainly written in C, with some code generated by *yacc* and *X-Designer*. An analyst or customer can use the toolset to explore and assess a requirements specification. Normally automatic selection of events is chosen, allowing direct interaction with the system by mouse clicks and drags.

Front-end tools comprise the editor and parser. SOLVE specifications are textual and can be prepared by a standard editor. However, the syntax-directed editor *syd* was developed with SOLVE in mind; the philosophy of *syd* is to enforce the syntax of the language down to a selected level. The *parser* tool is built using *yacc*, and translates SOLVE to LOTOS fairly straightforwardly [1]. Each object corresponds to a LOTOS process that communicates via a common intermediate process as a communication medium.

Animation tools comprise the simulator, displayer and animator. A modified version of the *hippo* tool is used to carry out LOTOS simulation. Although *hippo* is an old design, its attraction is simple communication with it via standard input/output. Other common tools are used to parse and check the LOTOS specification. To manage an interactive animation, the *animator* tool synchronises *hippo* and *displayer* event offers. The *displayer* displays object icons in response to requests from the *animator*, and passes mouse clicks and drags on object icons to the *animator*. The *animator* manages the interactive animation of a SOLVE description, communicating via Unix pipes to/from the standard input/output of *displayer* and *hippo*.

Overall, SOLVE has gone a long way to achieving the goals of SPLICE. Requirements capture and specification in disparate areas have been undertaken. It has also been possible to demonstrate visual animation of formal requirements specifications without requiring specialist LOTOS knowledge from the tool user.

References

1. A. McClenaghan. SOLVE: Specification using an object-oriented, LOTOS-based, visual language. Technical Report CSM-115, Computing Science, Univ. of Stirling, UK, Jan. 1994.
2. A. McClenaghan. XDILL: An X-based simulator tool for DILL. Technical Report CSM-119, Computing Science, Univ. of Stirling, UK, Apr. 1994.
3. K. J. Turner and R. O. Sinnott. DILL: Specifying digital logic in LOTOS. In R. L. Tenney, P. D. Amer, and M. Ü. Uyar, editors, *Proc. Formal Description Techniques VI*, pages 71–86. North-Holland, Amsterdam, Netherlands, 1994.