SPLICE I
(Specification using LOTOS for an Interactive Customer Environment – Phase I)
Final Report on Grant GR/H80811

Prof. Ken Turner
Department of Computing Science, University of Stirling

7th June 1994

1 Introduction

The SPLICE I project was an 18-month project running from August 1992 to March 1994, though the grant was available from June 1992. The work was undertaken by the principal investigator and two research assistants funded by the project for most of its lifetime. The project was the initial phase of the originally planned project that would have continued to full-scale tool development and application of the method. For brevity, 'SPLICE' in the following refers to Phase I of the project as performed under this grant.

The principal aim of the project was to bridge the gap between the customer, the specifier and the designer in specifying requirements. The objectives of SPLICE (summarised from the original project description) were:

- to develop a method for using the internationally-standardised formal specification language LOTOS (*Language Of Temporal Ordering Specification*, [1]) to capture software requirements in a specification
- to design prototype software tools for turning functional requirements into formal specifications written in LOTOS
- to investigate a method and prototype tools for explaining formal specifications to different levels of users, using animation of specifications and graphical interfaces to communicate the essence of specifications.

2 Work Undertaken

The following presentation of the work reflects the workplan set out in the original proposal. Of necessity only an overview is given here; see the referenced documents for more details.

2.1 Initial Surveys

Investigations were made of a number of areas in which requirements capture and animation could be undertaken. The problem domains studied were layered communications systems, digital logic design, neural networks, reactive systems, and planning in the construction industry. The last domain raised interesting questions of scheduling and interaction among many processes. However, it was decided that this was sufficiently well tackled by existing methods and tools in management science that it should not be pursued further. Work therefore concentrated on the other four domains. Relevant concepts for requirements capture in these areas were studied, leading to a strategy for tackling them from an architectural point of view. Only with layered communications systems was it possible to study a variety of existing LOTOS specifications for insight into specification style; the other areas were entirely new applications of LOTOS.

Existing LOTOS tools were also studied. The two toolsets available to the investigators were those produced by the ESPRIT SEDOS project and the ESPRIT LOTOSPHERE project. Although the LOTOSPHERE toolset, called LITE, was more recent and more advanced, it was still being developed during the lifetime of SPLICE and was somewhat unstable. More seriously, the LITE simulator (called *smile*) is graphically oriented. Although graphical simulation was a goal of SPLICE, *smile* has its own particular style of graphical operation that ran counter to the goals of SPLICE. In addition, *smile* suffers from the problems noted in the original proposal of being LOTOS-oriented and specifier-oriented. The decision was therefore taken to use the SEDOS tools. The SEDOS simulator (called *hippo*) has a simple standard input/output interface. This made it possible to incorporate *hippo* as a 'LOTOS calculator' within the SPLICE tools. SPLICE could then concentrate on developing an appropriate animation strategy and graphical interface.

A further question initially investigated was the most suitable windowing system to use. The original intention was to exploit the capabilities of the NeXTstep environment for NeXT workstations. Investigations showed that NeXTstep is a very suitable and congenial environment in which to specify and build graphical applications. In particular, the *InterfaceBuilder* tool offers considerable convenience and productivity gains. Unfortunately, NeXT Inc. ceased making hardware shortly after the project started and the future of NeXTstep became rather unclear. (Current indications for NeXTstep are positive.) It was decided to concentrate on prototyping of graphical interfaces using the X Window System. Although X is less sophisticated and tools such as *X-Designer* are less powerful than their NeXT equivalents, X is much more widely used and offers a stable environment.

2.2 Requirements Capture Method

2.2.1 Problem Domains

As reported in section 2.1 the problem domains studied in the project were layered communications systems, digital logic design, neural networks, and reactive systems. The original thesis of SPLICE was that requirements capture, specification and analysis would be greatly assisted by taking an architectural, component-based approach supported by a specification library.

Some work had already been undertaken prior to the project on layered communications systems such as OSI (Open Systems Interconnection). This work was consolidated during the project, and extended to service creation in telecommunications. A language SAGE (Service Attribute Generator) was defined to allow compact formulation of service requirements from a customer point of view. SAGE provides the building blocks (facilities) needed to define services in a modular and natural way. In addition, SAGE provides a range of combinators for building higher-level services from lower-level services and facilities. A specification component library was developed for SAGE, and also a translation tool that turns a SAGE specification into a LOTOS specification. The resulting LOTOS specification can be simulated in order to demonstrate its behaviour to the customer requiring the new service. The work on SAGE was reported in [13, 14].

Digital logic design offered a contrasting problem domain in which the same principles of architecture-driven requirements definition could be investigated, but at a much more concrete level. Hardware description is, of course, a well-investigated area. The emphasis in SPLICE was on specification architecture and graphical animation. A language DILL (Digital Logic In LOTOS) was defined to allow requirements for digital logic to be specified and animated in a component-based fashion. At the time of investigation, specification of digital logic in LOTOS had not previously been reported. The DILL approach parallels the SAGE approach in having a specification component library, a tool to translate DILL specifications into LOTOS, and simulation of the resulting specifications. Unlike layered communications systems, the combinators for digital logic components are rather simple: they correspond to wiring up a circuit. Although it proved surprisingly tricky to find the best ways of modelling components and their combinations effectively, the resultant approach works well. In particular, the composition and synchronisation features of LOTOS allow specification components to be 'wired up' very easily. The work on DILL was reported in [12, 14, 15].

Neural networks were selected for study as a rather different problem domain. Neural networks are highly concurrent and dynamic sets of largely similar components, combined in a highly structured way. They are parallel distributed processing systems that seem ideal as an application of LOTOS. The specification of neural networks and their design from requirements are active areas of research, and a new application for LOTOS. Library components were developed to support specification of aspects such as neurons, layers of neurons, fan-out units, etc. Learning functions, connectivity functions, neural states, etc. were modelled using the abstract data type facilities of LOTOS. In many neural networks, the environment is a time-varying stochastic function of inputs. Although there are timed (and stochastic) extensions to LOTOS that could have been used, it was felt that the project should concentrate on architectural, requirements-oriented specification of basic network structures. Perceptrons (simple, time-invariant neural networks) were therefore studied using two 'standard' examples: classifying pairs of bits related by *xor*, and classifying winning positions in noughts-and-crosses. Although useful progress was made during the project, much more work needs to be done. This includes extensions to other kinds of neural network (in particular to those than can learn without guidance), generating networks automatically, and developing user-oriented tool support for evaluation of networks. The work on neural networks was reported in [4, 6].

Reactive systems were chosen as the final area for study from the point of view of requirements definition. Reactive (or embedded) systems are widespread in modern equipment of all kinds. Their requirements often revolve around user interfaces and real-time issues. It was felt that dealing with real-time aspects would be a large investigation in its own right, and would divert the project from its concentration on requirements capture and animation. The user interface aspects were relevant to the project, complementing the other problem domains and fitting well with the graphical orientation of the tools to be developed. A less architectural and more pragmatic approach was taken to defining the requirements for reactive systems. A language SOLVE (Specification using an Object-oriented, LOTOS-based, Visual language) was defined in order to allow requirements for reactive systems to

be formulated. SOLVE is an object-oriented, visually-oriented language that is particularly suitable for specifying and animating interfaces to reactive systems. A rather basic library of specification components was developed, and used to model requirements for systems such as a VCR (Video Cassette Recorder) on-screen clock controller. SOLVE specifications are translated automatically into LOTOS and are then animated graphically. A set of tools supports this animation in a visual, user-oriented fashion. The strength of SOLVE is that it acts a bridge between customer requirements and the underlying formal specification in LOTOS. The same LOTOS specification can be used for graphical animation as well as for further system design. The work on SOLVE builds on PhD research by one of the investigators prior to the project [9]. Project work was reported in [10, 16].

2.2.2 Object-Oriented Requirements Capture and Analysis

The most promising means of easing requirements capture and specification was felt to lie in formal object-oriented approaches. Apart from the work on SOLVE (which was more animation-oriented), effort was invested in developing a LOTOS-based method for requirements capture and analysis using object-oriented principles. This is the method called ORCA (Object-oriented Requirements Construction and Analysis) that supports various kinds of static analysis (based on diagrammatic representations) and dynamic analysis (through animation).

Requirements capture and analysis methods tend to be analyst-oriented, allowing only limited customer involvement. The ORCA method was designed to be customer-oriented, allowing full involvement of customers at all stages. Apart from customer orientation, ORCA is object-oriented, formal, visual and constructive.

Customer orientation allows a flexible approach to analysis; a rigid framework was felt to discourage customers from communicating their needs effectively. ORCA also avoids the traditional emphasis on documentation as a basis for determining requirements, which can lead to isolation of the customer and the analyst. Instead, ORCA allows direct customer involvement in requirements definition. In particular, ORCA allows customers to alter the way in which requirements are presented so that they can be interpreted more readily in problem domain terms.

Object orientation allows requirements to be modelled in a natural way, i.e. natural to the customer. Although object-oriented and object-based techniques are widely used, there is a surprising variability in the way that even fundamental concepts are treated. Part of the project work was to develop a consistent framework of object-oriented concepts and to give these a precise basis.

ORCA is also formal. The method is underpinned by LOTOS, such that the end result of requirements analysis is a requirements specification in LOTOS. This offers the expected benefits of precision, analysability, and a foundation for tool support. Other work on the project concerned with animating requirements specifications in LOTOS complements ORCA.

ORCA is supported by various forms of visualisation. Graphical notations in requirements analysis are normal, but ORCA adds the extra ingredient of a formal basis for its graphical representations. A number of graphical views can be presented of requirements. These include navigation around large graphs, browsing and selection of classes, and choosing different views of the same structure (dynamic/static, hierarchical/flat). The same information can be presented with differing graphical emphasis, the intention being to allow the customer to see the requirements in the most appropriate form. This is done by defining mappings between the formal definitions of classes and how they might appear graphically. The range of mappings is provided by the analyst for selection by the customer.

Requirements modelling with ORCA might be described as constructive. The aim is to exploit a library of classes in the chosen application domains. The approach allows variations on this theme. Where a class is selected from a hierarchy, it is possible to be flexible about whether its superclasses and subclasses figure explicitly in the model. To promote component re-use, new classes can be aliased to old classes. Classes can be modified by extension/restriction and specialisation/generalisation. Various forms of class composition are permitted: union, sharing and internal connection. A template mechanism also allows classes to be built from their components in a more flexible manner than these forms of composition.

The work on ORCA builds on PhD research by one of the investigators prior to the project [5]. Project work was reported in [3, 7].

2.3 Communication of Specifications

Commercial tools to capture and display requirements tend to be sophisticated graphics editors. The descriptions that these tools produce are often inadequate models of the behavioural requirements, cannot be used as executable prototypes, and lack the rigour needed for testing and refinement. The project therefore carried out pilot studies to address these problems. The SOLVE approach uses graphical presentation and manipulation to convey the meaning of a specification. Unlike other approaches, SOLVE's primary concern is to deal with formal specifications.

The key concepts in SOLVE are object orientation, interactive animation, and formal specification (in LOTOS). SOLVE is a language for specifying and animating the requirements of reactive systems. These are typically human interface systems to devices such as VCRs and other domestic appliances.

SOLVE is designed to be used by people who are not familiar with formal languages (in particular LOTOS). It allows formal requirements specifications to be written using a simple object-oriented language, and to be explored using interactive animation. SOLVE is suitable for requirements capture in problem domains where systems are interactive (producing feedback in response to user input) and can be represented by visual animation. SOLVE has been tried out on a number of simple systems such as a VCR controller.

SOLVE allows the analyst to write a requirements specification in an intuitive, object-oriented language that is then automatically translated into LOTOS. The LOTOS translation is graphically animated, allowing the customer, analyst or designer to explore the requirements specification by interacting with it directly (e.g. by clicking or dragging object icons).

A key feature of SOLVE is visualisation. Each object is visualised as an icon. An object is responsible for displaying and modifying its own icon. An object icon can change to represent an abstraction of the state of an object or some part of the total system. The object icons visually inform the SOLVE user of what is happening in the system. Each simple SOLVE object is created with all the characteristics of a basic prototypical object. Once an object has been created it may be customised. In fact the specifier may build up a kit of predefined simple and composite objects.

A SOLVE specification consists of a number of objects that communicate via messages. A message either invokes an object method or returns the results of an invoked method. Objects may execute concurrently. Object declarations provide sort ('type') information used to check correct use of objects against their definitions. Each object has a list of instance parameters and a set of methods. The sorts of instance parameters are declared, as are the names and signature of the methods.

The SOLVE language is object-oriented and hopefully intuitive. It has been deliberately designed to resemble a programming language, avoiding the algebraic feel of LOTOS. However, a SOLVE description can be automatically translated into a LOTOS specification. Some of the mechanics of the SOLVE object-oriented system are automatically incorporated in the translation. This makes the LOTOS specification SOLVE-oriented, but makes it suitable for visual animation and for subsequent object-oriented development.

As well as use with reactive systems, SOLVE was also developed to allow requirements for digital logic circuits to be specified and animated; in this form the approach is known as XDILL (X-based Digital Logic In LOTOS). The work on SOLVE was reported in [10, 16] and that on XDILL in [11, 16]. Preliminary work was reported in [8, 9].

2.4 Tool Design

In the original proposal it was anticipated that tool *design* rather than *development* would be undertaken. In fact, the project made better progress than expected in tool development, though it should be said that the tools are only limited prototypes. Nonetheless, it has been possible to underpin most of the project approach with demonstration versions of tools.

Both SAGE (for layered communications systems) and DILL (for digital logic) are supported by specification component libraries. There is a tool for each language that translates it into LOTOS so that specifications can be animated. The specification libraries are actually complex macro packages written in the *m4* language and translated into LOTOS by the *m4* macro processor. The approach is simple but surprisingly powerful. Compact requirements declarations in SAGE or DILL are translated into LOTOS specifications that are 20 to 40 times larger. This gives some idea of the productivity gains possible in producing formal requirements specifications. The tool libraries also support specification component re-use, and allow the user to work at a more architectural level during requirements definition.

Preliminary investigations were carried out into supporting the ORCA approach with tools written in the Sather language running under SunTools. However, this work has not progressed beyond initial feasibility studies.

SOLVE (for reactive systems) and XDILL (for digital logic) are supported by a range of tools: *editor*, *parser*, *displayer*, *animator*, a modified version of the *hippo* simulator, *solve* and *xdill*. The tools use the the X Window environment and are mainly written in C, though some of the code is generated by *yacc* and *X-Designer*.

SOLVE specifications are purely textual and so can be produced using a standard text editor. However, the syntax-directed editor *syd* [2] was developed with SOLVE in mind. The *syd* editor is a novel compromise between a traditional text editor and a strict syntax-directed editor. The approach of *syd* is to enforce the syntax of the language down to a specified level. The syntactic level at which *syd* operates may be lowered or raised according to the requirements of the user. Another convenient feature is that the meta-syntax may specify the conventional textual layout of the language (e.g. the use of newlines and indentation). Although *syd* was developed for use with SOLVE, it is really a general-purpose, syntax-directed editor.

The *parser* tool was built using *yacc*. For a valid specification, the parser produces a LOTOS specification as well as a special control file for the *animator*. The translation from SOLVE to LOTOS is reasonably straightforward. Objects corresponds to LOTOS processes that communicate via an intermediate process as a communication

medium. The communication model permits dynamic modification of communication connections, although dynamic creation of objects and their connections are not currently supported by SOLVE.

The *displayer* tool displays object icons in response to requests from the *animator* tool, and it passes user requests to click or drag object icons to the *animator* tool.

The *animator* tool manages the interactive animation of a SOLVE specification, spawning *displayer* and *hippo* as child processes. The *animator* communicates via Unix pipes to/from the standard input/output of *displayer* and *hippo*. This simple means of communication is why the SEDOS simulator *hippo* was used rather than the later simulator *smile*, which is more difficult to interwork with. The purpose of *hippo* is to simulate the behaviour of the given system, yielding lists of possible events. The *displayer* turns user input (from the graphical interface) into event offers. This effectively yields lists of events offered by the environment. To manage an interactive animation, the *animator* synchronises the *hippo* event offers and the *displayer* event offers. During an animation there may be a choice of possible events; the *animator* may be set to resolve these automatically or the user may choose to do so.

Tool support for SOLVE and XDILL is essentially the same, differing largely in the command-line interface: *solve* or *xdill*. The whole toolset was reported in [10, 11, 16].

3 Resources Used

In the original proposal it was intended to employ one senior researcher. However, for circumstantial and technical reasons it was decided to employ two more junior researchers with expertise in the two main aspects of the project (object-oriented requirements analysis, graphical tool development). This change in staffing was agreed with SERC before the start of the project. Three taught MSc students on specialist conversion courses were also involved in project work as part of their studies.

In the original proposal it was intended to actively involve British Telecom and Hewlett-Packard in project discussions; both companies had expressed an interest in this at the proposal stage. However, changes in the organisation and direction of the Hewlett-Packard laboratories meant that the project was much less relevant to them than before; in fact almost no discussions took place. British Telecom maintained an interest in the project, but the number of discussions was less than expected. The principal investigator presented the work of the project at the PSTV XIII (*Protocol Specification, Testing, and Verification*) and FORTE VI (*Formal Description Techniques*) conferences. Some Departmental travel funding was provided for the PSTV conference. The net result of all these changes was an underspend on travel budget.

As anticipated in the original proposal, an existing SUN 4 workstation was used on the project. A NeXT workstation was also bought as anticipated, and used for investigation into graphical user interfaces. As expected, the NeXTstep system showed considerable promise for developing graphical interfaces. However, as reported in section 2.1 it was decided to make use of the X Window System rather than NeXTstep. A third workstation (an HP 9000 series 300 machine) was provided by the Department. LOTOS tools and X software were used on all three workstations to ensure a common development environment.

4 Deliverables

The tangible results of the project are papers and prototype tools. The main papers produced by the project are listed in the references. Considering that the project had very short time-scales, the productivity of the project in writing papers is good. The project was intended to focus more on the design than construction of tools. In fact, better progress than anticipated was made, and working tools (though still prototypes) were developed. Section 2.4 surveys the tools that were produced.

5 Future Work and Exploitation

The project has delivered satisfactory interim results. As anticipated in the original proposal, it would be productive to develop these further in a Phase II project. This would take the method and tools developed so far and turn them into practically usable form.

Formal requirements capture, specification and analysis using ORCA looks promising; the further work to be undertaken would be to 'industrialise' this as a method that could be used by engineers in real-life development. Architecture-based specification using libraries of specification components in various fields has also been validated. The existing libraries would need to be extended to deal with the range of concepts required by the relevant industrial sector. The most fruitful future applications would seem to be in the design of reactive systems, and in service creation and feature analysis for telecommunications.

The existing prototype tools would need to be developed further before they could be regarded as industrially usable. A major limitation at the moment is in the simulation capabilities of the LOTOS tools used. Further work might focus on exploiting more recent simulators such as *smile*, or in developing new and more practical simulation strategies. More effort would be required on graphical interfaces for creation and animation of specifications written in SOLVE.

Overall, the project is felt to have produced timely and worthwhile results. Productivity has been good, and reflects effective use of the resources available.

References

[1] ISO (1988). *Information Processing Systems – Open Systems Interconnection - LOTOS – A Formal Description Technique based on the Temporal Ordering of Observational Behaviour*, ISO/IEC 8807, International Organisation for Standardisation, Geneva.

[2] Philip A. Eccles. SyD-EG: A syntax-directed editor generator using standard Unix tools, M.Sc. thesis, University of Stirling, Scotland, March 1994.

[3] J. Paul Gibson. Formal object-based design in LOTOS. Technical Report CSM-113, University of Stirling, Department of Computing Science and Mathematics, April 1993.

[4] J. Paul Gibson. A LOTOS-based approach to neural network specification. Technical Report CSM-112, University of Stirling, Department of Computing Science and Mathematics, May 1993.

[5] J. Paul Gibson. Formal object-oriented development of software systems using LOTOS. Technical Report CSM-114, University of Stirling, Department of Computing Science and Mathematics, September 1993. *(A foundation for project work.)*

[6] J. Paul Gibson. A LOTOS-based approach to neural network specification. Submitted to *Formal Description Techniques VI*, Boston, October 1993.

[7] J. Paul Gibson. Object-oriented requirements construction and analysis (ORCA) environment: A conceptual tool. *In preparation for submission in 1994*.

[8] Ashley McClenaghan. RECAP-IS: A tool for capturing the requirements of interactive systems using LOTOS. Submitted to *Formal Description Techniques VI*, Boston, October 1993.

[9] Ashley McClenaghan. Distributed systems: Architecture-driven specification using extended LOTOS. Technical Report CSM-120, University of Stirling, Department of Computing Science and Mathematics, December 1993. *(A foundation for project work.)*

[10] Ashley McClenaghan. SOLVE: Specification using an object-oriented, LOTOS-based, visual language. Technical Report CSM-115, University of Stirling, Department of Computing Science and Mathematics, January 1994.

[11] Ashley McClenaghan. XDILL: An X-based simulator tool for DILL. Technical Report CSM-119, University of Stirling, Department of Computing Science and Mathematics, April 1994.

[12] Richard O. Sinnott. The formally specifying in LOTOS of electronic components, M.Sc. thesis, University of Stirling, Scotland, March 1993.

[13] Magdalene S. P. Teo. Component-oriented specification and analysis of communications services, M.Sc. thesis, University of Stirling, Scotland, September 1993.

[14] Kenneth J. Turner. An engineering approach to formal methods. In André A. S. Danthine, Guy Leduc, and Pierre Wolper, editors, *Proc. International Conf. on Protocol Specification, Testing, and Verification XIII*, pages 357–380, Amsterdam, June 1993. North-Holland. *Invited paper*.

[15] Kenneth J. Turner and Richard O. Sinnott. DILL: Specifying digital logic in LOTOS. In Richard L. Tenney, Paul Amer, and Ümit Uyar, editors, *Formal Description Techniques VI*, Amsterdam, October 1993. North-Holland.

[16] Kenneth J. Turner and Ashley McClenaghan. Visual animation of LOTOS using SOLVE. Submitted to *Formal Description Techniques VII*, Bern, October 1994.