Project: DRA FRONT END FILTER PROJECT

Title: Briefing for CLEF

Ref: DS/FMU/FEF/041 *Issue: Revision* : *2.2* *Date:* 5 June 2016

Status: Approved *Type:* Project Overview

Keywords:

Author:

Name	*Location*	*Signature*	*Date*
R. Stokes	WIN01		
R.B. Jones			

Authorisation for Issue:

Name	*Function*	*Signature*	*Date*
R. B. Jones	HAT Manager		

Abstract: This document provides a briefing suitable for a CLEF on the results of the formal methods work carried out by ICL for the DRA Front End Filter project RSRE 1C/6130.

Distribution: HAT FEF File

0 DOCUMENT CONTROL

0.1 Contents List

List of Tables

0.2 Document Cross References

[1] Michael J.C. Gordon and Tom F. Melham, editors. *Introduction to HOL.* Cambridge University Press, 1993.

[2] J.M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, 1989.

[3] DS/FMU/017. *Secure Database Technical Proposal.* High Assurance Team, ICL Secure Systems, WIN01, 21st January 1992.

[4] DS/FMU/FEF/002. *Errors in the Specifications.* G.M. Prout, ICL Secure Systems, WIN01.

[5] DS/FMU/FEF/003. *Formal Security Policy.* G.M. Prout, ICL Secure Systems, WIN01.

[6] DS/FMU/FEF/005. *Specifications of hide and updateState.* G.M. Prout, ICL Secure Systems, WIN01.

[7] DS/FMU/FEF/006. *Security Conjecture for the SSQL Abstract Machine.* G.M. Prout, ICL Secure Systems, WIN01.

[8] DS/FMU/FEF/014. *Specification of SSQL Semantics II.* G.M. Prout, ICL Secure Systems, WIN01.

[9] DS/FMU/FEF/015. *Proof of Security (IIe).* G.M. Prout, ICL Secure Systems, WIN01.

[10] DS/FMU/FEF/018. *Proposal for Phase 2.* G.M. Prout, ICL Secure Systems, WIN01.

[11] DS/FMU/FEF/021. *Specification of TSQL.* G.M. Prout, ICL Secure Systems, WIN01.

[12] DS/FMU/FEF/022. *SWORD Front End Architectural Model.* R.D. Arthan, ICL Secure Systems, WIN01.

[13] DS/FMU/FEF/024. *A HOL Specification of the SWORD Output Filter.* G.M. Prout, ICL Secure Systems, WIN01.

[14] DS/FMU/FEF/025. *Representation of an SSQL State as a TSQL State.* G.M. Prout, ICL Secure Systems, WIN01.

[15] DS/FMU/FEF/026. *Critical Requirements on the SWORD Query Transformations.* R.D. Arthan, ICL Secure Systems, WIN01.

[16] DS/FMU/FEF/029. *Specification of Query Transformations in HOL (II).* R.D. Arthan, ICL Secure Systems, WIN01.

[17] DS/FMU/FEF/032. *Table Computations for SWORD.* R.D. Arthan, ICL Secure Systems, WIN01.

[18] DS/FMU/FEF/034. *Phase II Proof Strategy.* R.D. Arthan, ICL Secure Systems, WIN01.

[19] DS/FMU/FEF/036. *Phase II Proof Finale.* R.D. Arthan and G.M. Prout, ICL Secure Systems, WIN01.

[20] DS/FMU/IED/USR014. *ProofPower Software and Services.* Lemma 1 Ltd., `http://www.lemma-one.com`.

[21] *SSQL Transformations.* Simon Wiseman, DRA, 14th January 1993.

0.3 Changes History

Revision : *2.2* **(11 January 1994)** Approved issue.

Issue 2.3 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document is a report on the results of the formal specification and verification work carried out by ICL as part of the development by DRA of SWORD, a secure relational database system. ICL's work on this project is generally referred to as the "FEF project" in the rest of this document.

This document is primarily produced in support of the intention of DRA to commission from a CLEF a further report. We understand from DRA that the latter report has a number of aims. The main aim of relevance here is to:

> Provide an assessment of whether the SWORD Front End approach is capable of achieving a useful ITSEC rating.

and the main relevant associated deliverable is taken to be:

> A report which gives an assessment of the assurance levels that the approach could attain, and the likely costs of such an evaluation.

The reference to evaluation costs is taken to be to costs recoverable by a CLEF in performing an evaluation, and is not in the scope of this report.

This report is written on the following assumptions derived from discussions with DRA:

- It is understood that DRA are interested primarily in the possibilities for an ITSEC Assurance Correctness Level of E3.
- It is assumed that evaluation would be as a *product*, (that is, according to ITSEC 1.4, a package which can be bought off the shelf), as opposed to a *system* or specific installation.
- DRA have a general interest in the value of Formal Methods in the development of secure systems, both for systems intended for evaluation at E5 or E6, and also for systems intended for evaluation at lower levels for which the use of formal methods is not mandated by ITSEC. This interest is not confined to, but does encompass, the contribution which Formal Methods might make to a successful evaluation at any particular level.

The intention in writing this report is to try to assist the CLEF, by providing a first introduction to the results of the FEF project. These results, or similar results, may be considered as possible evaluation deliverables, and so some comments are offered, in the hope that they may be useful to the CLEF, but not with the intention of trying to pre-empt the judgement of the CLEF.

1.2 Introduction

This report is divided into five further sections.

Section 2 describes the objectives of the formal methods work.

Section 3 describes the key achievements against the identified objectives.

Section 4 describes the results of the FEF project, particularly considering the significance of the various formal specifications and of the theorems which have been proved about them.

Sections 5 and 6 relates the results of the FEF project to possible deliverables for an evaluation at various levels of the ITSEC criteria. It is recognised that the FEF project is of the nature of research, while evaluation deliverables will be produced by a development project. Consequently, what is said in this section about the FEF results is meant to apply to some evaluation scenario in which the formal methods deliverables would be similar to those of FEF.

Section 7 offers observations and conclusions.

1.3 Terminology and Abbreviations

CLEF Commercial Licenced Evaluation Facility

DBMS Database Management System

FEF Front End Filter

This term is used to describe an implementation technique for SSQL. SSQL queries provided by a client user or process are transformed into SQL queries which are then executed on a standard commercial DBMS. Output from the SQL queries is filtered to remove sensitive information before it is returned to the client. In this document the phrase 'FEF project' is used to refer to ICL's work on the formal specification and verification of aspects of the FEF development being undertaken by DRA.

HAT High Assurance Team, the unit within ICL which carried out the formal methods work described in this document.

ITSEC Information Technology Security Evaluation Criteria

RDBMS Relational DBMS

SSQL Secure SQL

A variant of the SQL database query language with features supporting multilevel security. SSQL differs from SQL mainly in that there is an additional data type, *Class*, for security-classifications, which is used to classify the information held in tables and their constituent rows and columns.

SQL Structured Query Language

A standard relational query language for querying and updating a database.

TOE Target Of Evaluation

TSQL Target SQL

A variant of SQL, augmented with a data type *Class* which can be used to represent security-classifications, but which has no other special interpretation.

2 THE OBJECTIVES OF THE FORMAL METHODS WORK

The objectives of the formal methods work undertaken under contract RSRE 1C/6130 (as set out in Annex 9 to the ITT) were:

- To provide an assessment of the applicability of computerised mathematical theorem proving tools to the construction of secure systems of realistic size and complexity.
- To result, in combination with DRA in-house work, in the production of a prototype secure DBMS capable of being evaluated to a medium level of assurance.

The work completed to date has fallen into two distinct phases which had more specific objectives as follows.

In the first phase the primary objective (see [3] Section 2.1) was:

- The production of a machine-checked formal proof of the security of an abstract machine modelling a secure database supporting the query language *SSQL* as defined by DRA in Annex 2 to the ITT.

The second phase of the work was to involve formal modeling of the design of the front end filter rather than the SSQL semantics. After reviewing the requirements for Phase 2 in the light of the results of Phase 1, the following revised statement of objectives was agreed (see [10] Section 1.2):

- To use the proof process to discover flaws in the specifications.
- To evolve the specification during the course of the proof as necessary to render it satisfactory.

The main purpose of the requirement for Phase 2 in the original ITT was to verify the design of the Front End Filter. The most important (if not the only) concern was that the filter would result in a secure database.

This proposal for Phase 2 therefore attempts to provide the most effective way of employing formal modelling and proof to improve confidence in the security of the Front End Filter.

The specifications referred to in the Phase 2 proposal are those of the Front End Filter. The relevant notion of security is that in the formal security policy [5].

3 ACHIEVEMENTS IN RELATION TO OBJECTIVES

The key achievements in relation to the stated objectives are summarised in sections 3.1 and 3.2 below. A more detailed description of the results may be found in Section 4. A description of the errors detected in the specifications as a result of the formal methods work is given in Section 4.4.

3.1 Phase 1

The original objectives took the required Phase 1 proof to be concerned with *confirming* the correctness of the specifications (w.r.t. the formal security model), while acknowledging a risk that the specifications might prove not to be correct.

In fact, the formal work proved more valuable than had been anticipated. Significant numbers of minor errors were discovered in the specifications and corrected during the formalisation, syntax checking and type checking of the specifications.

During the proof work one error was discovered which rendered the SSQL abstract machine as specified insecure. This error was corrected and the formal proofs were completed.

Thus the primary objective of Phase 1 was fully satisfied, and in addition a number of errors were detected and corrected in the original specifications.

3.2 Phase 2

The objectives of the Phase 2 work were reviewed prior to the commencement of Phase 2, and re-oriented toward using the formal specification and proof work for detecting errors in the design specifications for the Front End Filter.

Partial formal proofs of the security of the SSQL system as implemented using the Front End Filter were then undertaken.

Though the greater complexity of the security-relevant aspects of the specifications (by comparison with Phase 1) prevented completion of the relevant proofs, partial proofs were accomplished as planned. The rate of detection of errors was similar to that in Phase 1, but the proportion of substantive errors in the security of the specifications was significantly higher in Phase 2 than in Phase 1.

4 DESCRIPTION OF RESULTS

4.1 Introduction

The work under the FEF project considered in this document was commissioned and carried out in two phases. The bulk of the work was carried out between March 1992 and December 1993.

The main technical aim of Phase 1 was to produce a formal specification of the semantics of SSQL in a form suitable for processing using a computerised theorem-proving system and to provide a machine-checked proof that the SSQL semantics provided information-flow security according to an agreed formalisation of the 'No Flows Down' security policy. An important feature of the approach taken was to structure the formal specification of the SSQL semantics so that security critical aspects were clearly separated out.

The original intention for Phase 2 had been to verify that the formal model of the Front End implementation of SWORD was a 'secure refinement' (as defined in [3] Section 6.1) of the SSQL abstract machine (as specified in Phase 1) and so to demonstrate that it was secure. However, the requirements for Phase 2 were reviewed towards the end of Phase 1, and it was agreed that the

complexity of the implementation was likely to make a full correctness proof prohibitively expensive. Moreover, complete informal specifications of the implementation were not expected to be available soon enough. The revised objectives for Phase 2 were therefore to use formal specification and proof to discover any security flaws in the informal specifications, and to evolve the formal specifications to eliminate any flaws discovered.

The deliverables of the FEF project mainly comprise documents containing specifications or proof scripts. Specifications and theorems are expressed in ProofPower-HOL, a version of Higher Order Logic supported by ProofPower [20]. ProofPower-HOL is a formal logic suitable for specifying models of systems, and for formulating and proving theorems about such models. The concrete syntax for ProofPower-HOL is in some ways similar to the Z notation [2], but the underlying abstract syntax and the logical system follows very closely that of HOL88 [1]. ProofPower is a tool which gives support for specification and proof in ProofPower-HOL (and other formal notations) via comprehensive programmable facilities for developing and interrogating a database of specifications and theorems. The database is organised as an extensible hierarchy of modules referred to as "theories". It also gives facilities for preparing documents containing a mixture of narrative and mathematical material. Theory listings may be automatically included in such documents, giving a precise summary of the mathematical content of specifications and of any theorems which have been proved. Most of the FEF project deliverables are documents containing both narrative and the formal material defining a theory together with the theory listing.

The results of the formal treatment from the two phases are discussed in more detail in sections 4.2 and 4.3 below.

In addition to the formal documents themselves, a key output of the work is the feedback to the DRA designers about security and other problems which were found during the course of the work. This is discussed briefly in Section 4.4. The document [4] gives a complete account.

4.2 Phase 1 Results

4.2.1 Overview

The main specification work in Phase 1 was the specification of a specific, although quite abstract, system to which the Security Model is formally applicable. This system, the SSQL Abstract Machine is intended to serve as a formal definition of the SSQL semantics. The main proof work in Phase 1 was to prove that the SSQL Abstract Machine is secure in the sense of the Security Model. The specification of the SSQL Abstract Machine is structured in such a way that the critical security features are separated out from the specification of detailed functionality. This makes clearer what is and is not essential to security and simplifies security proofs.

Sections 4.2.2 to 4.2.5 below discuss the Phase 1 results and their significance in greater detail.

4.2.2 Security Model

The Security Model[5] defines a property, *secure*, of behavioural models of systems. This is a 'non-interference' formulation in which it is required that highly classified inputs do not 'interfere' with the values of less highly classified outputs. This property is later applied to state transition models via a process of behavioural abstraction. The Security Model was actually formalised and agreed in

advance of Phase 1 and was taken as the common overall statement of critical security requirements for both phases of the work.

4.2.3 SSQL Semantics

The specification of the semantics of SSQL describes how the result of an SSQL query is computed from the state of the database, covering both the response returned to the user and the new state.

The specification of the semantics is factored into three components, two of which contain all the security-relevant details and one of which contains the features which are not relevant to security. This structure is described in [3] Section 6.3.4.

Of the two security-relevant parts, one (known as 'hide' [6]) determines the constraints on read access to the database which apply at the clearance of the query submitted, and the other ('update' [6]) determines the constraints on write access to the database at the relevant clearance. In the context of these constraints the detailed semantics of the query language ('process query' [8]) can be specified in a way which does not contribute to the complexity of the security proof.

The specification of 'process query' is incomplete (that is, loose), in certain respects. Of the four kinds of query (*insert*, *delete*, *update* and *select*), it is only for *select* that full details of the functionality are specified. The specifications together contain sufficient information to establish that the SSQL Abstract Machine of Section 4.2.4 will be secure however the omitted details are filled in, as demonstrated by the completion of the proof against the formal model of the security policy.

4.2.4 SSQL Abstract Machine

To establish conformance of the system to the policy a 'behavioural model' of the system is required. This is constructed in two stages from the components of the SSQL semantics.

In the first stage a model of the target system as a state transition system is constructed. This is known as the 'SSQL Abstract Machine'.

In the second stage an operation of 'behavioural abstraction' is performed on this abstract machine to give a behavioural model of the kind required in the security model.

The SSQL Abstract Machine consists of two parts:

- The transition function.
- The initial state of the system.

The transition function is constructed from the three components of the SSQL semantics. A view of the database state is constructed which hides anything not in the user's clearance (using the 'hide' component of the semantics). The query is then processed against this view (and thus with no 'read up') using the 'process query' component. The results of the query are then applied to the database using the 'update' component of the semantics, which also computes the output to be supplied to the user. The 'update' component is responsible for ensuring that there is no 'write down'.

The initial state is loosely specified as conforming only to essential security invariants on the state.

The SSQL abstract machine is documented in [7] under the name of *SSQLam*.

4.2.5 Theorem about *SSQLam*

The following theorem is proved as the key result in Phase 1:

$$\vdash behaviours\ SSQLam \in secure$$

A theorem is expressed in ProofPower-HOL as an optional list of assumptions followed by the turnstile symbol, "⊢", followed by a formula, called the conclusion of the theorem. The theorem is the assertion that the conclusion is true provided all the assumptions are. The ProofPower system ensures that the only theorems which can be computed are ones which have been derived from an identifiable list of axioms and definitions via mathematically sound rules of reasoning. In this case, there are no assumptions, and the conclusion expresses a relationship between three named objects whose definitions form part of the Phase 1 specification work. The theorem asserts that the result of applying the behavioural abstraction operation, *behaviours*, to the SSQL Abstract Machine, *SSQLam* produces a system which belongs to the set, *secure*, of systems which conform to the Security Model.

The proof is documented in [9] under the name *secureSSQL*. The proof comprises a formal validation of the fact expressed by the above theorem, viz., that the behaviours of the abstract-state machine *SSQLam* satisfy the information-flow constraints imposed by the Security Model of Section 4.2.2.

4.3 Phase 2 Results

4.3.1 Overview

In the Phase 2 specifications, various aspects of the actual Front End implementation of SWORD were modelled with a view to doing relevant proof work to discover security problems in the design. The specifications were structured to expose security critical aspects of the design so that proof work could be done in the areas where most benefit was likely to be derived. At the top level a formal model is given of the overall system architecture. This gives rise to a decomposition of the critical security properties into properties of the top level subsystems of the actual Front End design; it also gives an opportunity to verify that the critical properties on the subsystems are sufficient for overall system security. However, these subsystems do not partition conveniently into security-critical and non-critical parts. At the next level down the interactions between the subsystems were investigated by relating them to a slightly more abstract view of query execution. At the third level, the constraints on query execution are re-expressed in terms of an explicit model which can be directly related to the SSQL syntax and which specifies precisely the security-relevant computations which are to be performed when a query is executed.

The main proof work was done at the third level, to prove that the model satisfies information-flow constraints which, at least intuitively, bear a close relationship with the overall Security Model. Proof work was also done to give a partial proof that these information-flow constraints are sufficient to ensure conformance of the system to the security requirements. This amounts to showing that satisfaction of the information-flow constraints, together with some plausible assumptions about other aspects of the system, does entail that a system constructed according to the architectural model will conform to the Security Model. A small amount of pilot proof work was carried out at the other two levels (however, the proof work at the top-level is not further discussed below).

Sections 4.3.2 to 4.3.9 below discuss the Phase 2 results and their significance in greater detail.

4.3.2 Specification of Architecture

A formal top level description of the architectural construction of the Front End implementation of SWORD is given in [12]. This construction defines how the following subsystems are combined to form the SWORD system:

- *TSQLtf*, [11], corresponding to a "conventional DBMS", implementing *TSQL*. This may or may not be implemented in reality by a "standard commercial DBMS", and so to avoid taking a position on this issue this architectural component will here be called the "TSQL engine".
- *STP* ("SSQL Transformation Processor", [16]) which transforms queries input in SSQL to queries in TSQL to be submitted to the TSQL engine.
- *outputFilter*, [13], which sanitises results from the TSQL engine before passing them back to the user.

The construction also requires a state-representation function, specified as *reprState* in [14].

4.3.3 Specification of Security-Critical Properties

The security-critical properties of the top level subsystems of the Front End implementation are documented in [12] under the names:

subsys_secure
subsys_secureA
subsys_secureB
subsys_secureC
subsys_secureD
subsys_secureE

4.3.4 Subsystem Lemmas

The document [18] identifies some lemmas, which may be called the 'subsystem' lemmas. Taken together, these lemmas assert that the security-critical properties (Section 4.3.3 above) are true of the subsystems of the architecture. A lemma is also identified asserting that the architecture reduces the critical property of the the system as a whole to the critical properties of the subsystems.

Thus a proof of security according to the policy of [5] assuming the lemmas will be a proof of the correctness of the architectural structuring as a first design-step. Since the lemmas are stated in terms of specifications of the individual subsystems of the architecture, they represent proof-obligations which are carried forward to later design-steps. The subsystem lemmas are documented in [18] under the names:

Architecture_Secure
Subsys_SecureA
Subsys_SecureB
Subsys_SecureC
Subsys_SecureD
Subsys_SecureE

4.3.5 Specification of *FE_SWORD_SYSTEM*

FE_SWORD_SYSTEM is a DBMS; it is the architectural construction applied to the architectural subsystems. It is defined in [18].

4.3.6 Proof of *Theorem1*

Theorem1, stated and proved in [18], is

$$\begin{array}{l} \quad Architecture\_Secure, \\ \quad Subsys\_SecureA, \\ \quad Subsys\_SecureB, \\ \quad Subsys\_SecureC, \\ \quad Subsys\_SecureD, \\ \quad Subsys\_SecureE \\ \vdash \; FE\_SWORD\_SYSTEM\_secure \end{array}$$

Here there are 6 assumptions, namely the subsystem lemmas described in Section 4.3.4 above. The conclusion of the theorem here is the boolean term $FE\_SWORD\_SYSTEM\_secure$. This is defined in [18] to be equivalent to $FE\_SWORD\_SYSTEM \in secure$, (and so the theorem has a similar form to the main theorem, *secureSSQL*, of Phase 1, discussed in Section 4.2.5). This result shows that the architecture specified is correct with regard to the Security Model. In other words, the security-critical requirements on the subsystems and the construction have been correctly identified in that they are indeed sufficient to ensure the security of the constructed DBMS.

4.3.7 Specification of Execution Model

What is called the 'Execution Model' represents a conceptual step in the development process following on from the Architecture definition. The Execution Model may be described as a specification of a new constraint on the architectural subsystems *TSQLtf* and *STP*. The purpose of the constraint is to provide an alternative to the somewhat intractable semantics of TSQL for use in the further formal modelling.

The Execution Model is documented in [15] through the specification of three new objects: EM_1, *compile* and *upd*. EM_1 is a constant, while *compile* and *update* are variables (representing subsystems whose detailed specification is not available). EM_1 is a construction function for composing the two subsystems *compile* and *upd*. *compile* models compilation a query to give a formal representation of a database operation; *upd* models the use of the result of *compile* to update a database state. EM_1 composes the two subsystems by compiling and executing a query, using *update* to modify the database state or "outputting" the result of a select query as appropriate. A loose specification for *compile* and *upd* is given by a definition of correctness relative to the TSQL semantics:

$$Correct\_Compile \; = \{(compile,\; upd) \mid EM_1 \; compile \; upd = TSQLtf\}$$

4.3.8 Reduction of Proof-Obligations: *Theorem4*

Further development, on the familiar recursive pattern, consists of a design-decomposition of a subsystem into components, together with further correctness-proof work to reduce the subsystem proof-obligations to component proof-obligations.

In this case, the subsystems in question are *TSQLtf* and *STP*, taken together, and the components in question are EM_1, *compile* and *upd*. Use is made of the definitions of the Execution Model to show that the lemma *Subsys_SecureE* can be reduced to three lemmas, called:

EM_SecureE
Correct_Compile_OkSTP
TableComputationsSecure

These lemmas are documented in [18], and the relevant theorem is stated and proved in [18] as *Theorem4*:

Architecture_Secure,
Subsys_SecureA,
Subsys_SecureB,
Subsys_SecureC,
Subsys_SecureD,
EM_SecureE,
Correct_Compile_OkSTP,
TableComputationsSecure
$\vdash$ *FE_SWORD_SYSTEM_secure*

which may be compared with *Theorem1* above. The approach here is that of so-called "partial proofs", in which a partial attack is made on a verification problem by successively attacking the assumptions of an initial theorem (typically vacuous: $P \vdash P$). The goal being to replace each assumption of the initial theorem with zero or more assumptions which are more plausible, or which constrain fewer or simpler components of the system.

The significance of this theorem is that assumption *Subsys_SecureE* of *Theorem1* has been shown to be reducible to a condition, *EM_SecureE* on EM_1, provided that:

- *STP* is related in a certain way to *compile*, the relationship being that specified in the definition of *Correct_Compile_OkSTP*. In essence, this says that the input-output behaviour of a select query in any state could equally well be computed by interpreting the query in the state according to a set of rules captured in the definition of a set of computations called *TableComputations*. Subject to various modelling assumptions discussed in detail in [17], the details of the definition of this set are based closely on the semi-formal definition of the query transformations specified informally in [21]. In a certain sense, *TableComputations* embodies a description of SSQL in terms of a relational algebra augmented with security features.
- All the computations in the set *TableComputations*, satisfy certain natural information-flow constraints, as specified in the definition of *TableComputationsSecure*.

4.3.9 *Theorem5*

Further proof work results in reducing *Theorem4* above to *Theorem5* below, which is documented in [19]. This result is "the best partial proof of the overall system security" achieved under the FEF project.

$$
\begin{array}{l}
\quad Architecture\_Secure, \\
\quad Subsys\_SecureA, \\
\quad Subsys\_SecureB, \\
\quad Subsys\_SecureC, \\
\quad Subsys\_SecureD, \\
\quad View_t\_secureE, \\
\quad outputFilter\_secureE, \\
\quad Correct\_Compile\_OkSTP, \\
\quad \vdash \ FE\_SWORD\_SYSTEM\_secure
\end{array}
$$

Theorem5, in comparison with *Theorem4*, represents progress in the following respects:

- The assumption *TableComputationsSecure* has disappeared, since it has been proved as a theorem, as documented in [19]. This indicates that the bottom level of proof work as discussed in Section 4.3.1 is complete — the SSQL relational algebra satisfies the information-flow constraints imposed upon it.
- The assumption $EM\_SecureE$ has been replaced by two new assumptions $View_t\_secureE$ and $outputFilter\_secureE$. This indicates partial progress on the second level of proof work and reflects a reduction of the assumption $EM\_SecureE$ of *Theorem4* to conditions, $View_t\_secureE$ and $outputFilter\_secureE$ on two of the constituents of EM_1.

4.4 Issues Detected

The main issues with the SSQL and Front End designs which were detected during the project are described and discussed in [4]. For present purposes these may conveniently be classified and summarised as follows:

- **Syntax and type checking errors:** the bulk of the design was conveyed to ICL in a semi-formal notation; transcription of this into a machine-checked formal language revealed errors such as misspellings of names, failure to pass arguments correctly, and omission of auxiliary definitions. While none of these problems cause the system as specified to be demonstrably insecure, they are likely to prevent a proof of security from going through, and make a proper judgement about security difficult. In some cases the correction is obvious and the problem would therefore be unlikely to result in an insecure implementation, but in other cases it is less clear how the problem should be resolved. In these cases there is a greater risk that a misunsdertanding might arise which would cause the implementation to be insecure.

  This type of problem is detected automatically as the design is transcribed and checked using the ProofPower tool.

  Approximately 30 problems of this class were reported in Phase 1 and about the same number in Phase 2.

- **Incompleteness:** for example, security of the query transformations of [21] relies on the system not loading into its symbol tables information about directories and tables which the client is not cleared to see; this was not made explicit in the design material. The detection of this type of error depends on whether an object is just referenced and not defined (in which case the tool will flag the omission) or whether an object is inappropriately given a dummy definition (in which case the significance of the omission may not become apparent until proof is attempted).

  Some 13 problems of this class were reported in Phase 1 and 7 in Phase 2. Perhaps 5 or 6 of the 7 Phase 2 problems were directly relevant to security in that a security check was omitted from the design in a way which would lead directly to a violation of the Security Model if reflected in the implementation.

- **Algorithmic errors:** for example, security of the overall system is crucially dependent on clearances of fields within derived tables being computed in a way which respects the possible flow of information into the field. The functional requirements on SWORD mean that some of these computations are quite subtle. This type of error, if it is security-relevant, is certain to be exposed during a full proof of security, and has a good chance of being exposed by partial proof work as carried out in Phase 2 if the area of attack is carefully chosen.

  1 problem of this type was discovered in Phase 1 and 5 were discovered in Phase 2. All but one of these problems were discovered during the proof work rather than during formulation of the specifications, and all would lead directly to a violation of the Security Model if reflected in the implementation.

- **Modelling errors:** these are errors where the assumptions used to model a system themselves give rise to problems which would not actually arise in an implementation: an example in the FEF work was in the Phase 2 model of select query execution (*TableComputations*), which, in a sense, tries to recover from errors which cause execution to be aborted (or never initiated) in the implementation.

  Only 1 significant problem of this type arose (viz., the above one).

5 RESULTS AND EVALUATION DELIVERABLES (E4/E5/E6)

This and the following section attempts to relate the results of the FEF project to the deliverables required for an evaluation at various assurance levels according to the ITSEC criteria. This section is in support of the aim of DRA to commission a report from a CLEF which will 'provide an assessment of whether the SWORD Front End approach is capable of achieving a useful ITSEC rating'.

The architecture of the SWORD database, as implemented using a 'Front End Filter', is not expected to permit evaluation at high assurance levels, since security of the database would depend upon correct operation of the commercial database, which would not be highly assured. Nevertheless formal methods have been extensively used on the FEF project, even though fully formal specifications are not required below assurance level E4.

These methods were applied because they were thought to be beneficial in their own right, irrespective of the target evaluation level. A second factor motivating the use of formal methods was the desire of DRA to assess the feasibility of formal specification and proof in such applications.

Formal methods were exploited at a level similar to that which would have been required for evaluation at E6. The emphasis in undertaking the formal treatment was on formally proving the security of the system, and on using the formal work to eliminate flaws in the design, rather than on conforming to the formal aspects of the requirements for certification.

This section gives some indication of how the formal methods work may be related to the requirements for formal treatment at E4/E5/E6. The next section considers how the formal treatment might be interpreted in relation to the requirements for E3.

The requirements for documentation at level E6 in the ITSEC (E6.1) identify the following formal specifications:

- A formally specified model of security.
- A formal description of the architecture of the TOE.

The formal security model is also required at E3 and E4 though a fully formal description of the architecture of the TOE is not required at these levels. A number of informal documents are required which relate these formal specifications to other aspects of the system documentation. The FEF project has not attempted to provide any of the associated informal documentation.

A formalisation of the "No Flows Down" policy in Z was provided as a part of the proposal to DRA for the FEF contract, and was subsequently transcribed from Z to HOL with some minor adjustments for the purposes of the FEF project[5]. This is offered as a "formally specified model of security".

The main bulk of the formal specifications produced under the FEF contract contributed to various aspects of the specification of the architectural design of the TOE. They are largely derived from semi-formal specifications developed at DRA Malvern. The specifications produced in Phase 1 concerned the external interfaces. The structure present in these specifications was intended to make visible the security characteristics of the required behaviour rather than to reflect the architecture of the proposed implementation. The specifications produced in Phase 1 were complete in their treatment of the security aspects of the external (SSQL) interface, but omit some details of other aspects of the interface (e.g. aspects of the behaviour of update commands which are not relevant to security).

Phase 2 of the FEF project was primarily concerned with modelling the proposed architectural structure of the TOE, and with validating its security. The approach adopted in Phase 2 was to construct models of the key algorithms employed in the Front End Filter, in order to validate the correctness of these algorithms. These models do not fully reflect the architecture of the TOE, since some of the key algorithms are closely integrated with less critical aspects of the functionality of the filter. This approach to modelling key features of the architecture of the TOE was taken to maximise exposure of the aspects of the proposed architecture which are most sensitive for security and to permit validation of these aspects by formal proof of their required properties.

The proofs undertaken under the FEF project may suffice as the formal part of the requirements for evidence of consistency of the TOE with the formal security policy model as specified in E6.6 of the ITSEC. No attempt was made in the FEF project to supply the informal explanations required for evaluation at E6 or below.

6 RESULTS AND EVALUATION DELIVERABLES (E3)

This section addresses the possible relevance of the formal treatment undertaken in the FEF project to an evaluation with a target assurance level of E3.

The approach will be to consider selected deliverables, and the requirements on each, and then to discuss how results from the FEF project may be used in satisfying those requirements.

The list of E3 deliverables which are taken to be within the scope of this report is as follows:

1. Security Target [Ref E3.1, E3.2, E3.3].
2. Informal Description of the Architecture.[E3.1, E3.5, E3.6].
3. Informal Description of the Detailed Design. [E3.1, E3.8, E3.9].
4. Information on the Acceptance Procedure of the Configuration Control System.

Additionally, there is a requirement for an informal description of the correspondence between source code or hardware drawings and the detailed design [Ref E3.1, E3.11]. This is relevant only in so far as it characterises a detailed design as that with which a source-code correspondence can be adequately established.

Each of these three deliverables is now considered in more detail. The ITSEC term 'Target of Evaluation' (TOE) is interpreted as 'DBMS'.

6.1 Security Target

The Security Target deliverable is the statement of security requirements. The requirements described under the ITSEC headings 'Requirements for Content and Presentation' and 'Requirements for Evidence' are:

1. It shall describe the security enforcing functions to be provided by the DBMS.
2. It shall include a rationale identifying the method of use, the intended environment and the assumed threats within that environment.
3. The security-enforcing functions within the security target shall be specified in an informal style.
4. It shall describe how the functionality is appropriate for that method of use and is adequate to counter the assumed threats.

The FEF Security Model of [5], Section 4.2.2 above, can reasonably be part of an offering to satisfy the first of these requirements. It describes only the security-enforcing function of the DBMS which is the prevention of down-flows.

Of the eight generic headings identified by the ITSEC [ref 2.31, 2.32] for possible security-enforcing functions, the Security Model concerns only *Access Control*. The Security Model used was a straightforward formalisation of the 'No Flows Down' policy agreed prior to commencement of the FEF work.

As agreed at that time, the policy treats inputs to and outputs from the system as single class data values. The policy is formulated in terms of input/output behaviour without reference to the internal state of the system, but is intended for applications (such as SWORD) where the state must be regarded as containing information at a variety of classifications. In the SWORD system inputs are actually multi-level structured objects. The treatment of inputs is therefore a simplification of the intended characteristics of the system. The effect of this simplification is that the formal policy model places slightly weaker constraints on the allowable information flows than those intended for the system. In particular, information from substructures of the input which are intended to be more highly classified than the input as a whole, may be permitted to flow to any output whose classification dominates that of the input. In addition, because of the limitations of pure non-interference formulations of information flow, information in the initial state of the system is not protected from disclosure (this may be treated as an implicit constraint that there be no sensitive information in the initial state).

Some pilot work is currently in progress on formulating an appropriate treatment to strengthen the flow constraints imposed on structured multi-level inputs and outputs.

With regard to the third requirement, the ITSEC mandates that an informal style of presentation shall be used for all security targets, any formal specification being in addition [2.66]. [5] contains a mixture of informal and formal text. The opinion is offered that although [5] may not be quite suitable for someone to understand without special training, it is not far from it.

6.2 Architecture

The requirements for the Description of the Architecture, under the headings of Content and Presentation and Evidence, are as follows. Again writing 'DBMS' for 'Target of Evaluation', then the Description of Architecture shall describe:

1. the general structure of the DBMS.
2. the external interfaces of the DBMS .
3. any supporting hardware or firmware with statement of the functionality of supporting protection mechanisms.
4. the separation of the DBMS into security-enforcing and other components.
5. how the security-enforcing functions will be provided.
6. how the separation into security-enforcing and other components is achieved.

6.2.1 Requirement 1: General Structure

The FEF project result which is the Specification of the *FE_SWORD_SYSTEM* can reasonably be offered in satisfaction of (part of) the first requirement, the description of the general structure.

This specification is relatively abstract. It is described as an Architectural Model, wherein relationships between subsystems are modelled by the composition of functions (in the mathematical sense). At more concrete level, a description of general structure might be in terms of processes, flows of data

between processes and separation between processes (e.g. issues of physical separation or network topology).

The basis of the approach to security is to separate the user from the TSQL engine by interposing the Front End. An account of how this separation is concretely achieved, thereby establishing the relevance of the abstract Architectural Model, is evidently fundamental for evaluation. Further issues arise in that, because the *FE_SWORD_SYSTEM* is specified relatively abstractly, there may be further structural components not identified. Possible examples are:

- Processing of the concrete representations of input queries and output tables.
- Database administrator's interface.
- Trusted functions e.g. hardcopy labelling.

6.2.2 Requirement 2: External Interfaces

The specification of the *SSQLam* (Section 4.2.3 above) is a candidate for the specification of the main external interface of the DBMS. This specification is not currently complete.

6.2.3 Requirement 3

No statement is available with regard to dependence on protection mechanisms in supporting hardware/firmware.

6.2.4 Requirement 4

With regard to the separation of the DBMS into security-enforcing and other components, there are various positions which might be taken, depending upon the rationale of modes of use and threats which would be supplied for evaluation as part of the security target.

Evidently one possibility is to draw the dividing line between the Front End and the TSQL engine. The latter is one of the subsystems described by the Architecture. Since the TSQL engine has no functionality specific to multilevel security, then it may be asserted to be not a security-enforcing component. However, this does not mean that the TSQL engine may be entirely untrusted, since its correct functioning is security-critical.

6.2.5 Requirements 5 and 6

Regarding these two requirements, there is nothing to add to the discussion in Section 6.2.1 above.

6.3 Detailed Design

The requirements under the headings of Content and Presentation and Evidence, for the Description of the Detailed Design, are as follows:

1. specify all basic components.

2. describe the realisation of all security-enforcing mechanisms and security-relevant functions.
3. map security enforcing functions to mechanisms and components.
4. document all interfaces of security-enforcing and security-relevant components, stating their purpose and parameters.
5. provide specifications/definitions for mechanisms, which shall be suitable for the analysis of interrelationships between the mechanisms employed.
6. show a clear and hierarchical relationship between levels of specification (if more than one level is provided).
7. describe how the security mechanisms provide the security enforcing functions specified in the security target.
8. describe why components for which no design information is provided cannot be either security-enforcing or security-relevant.

The formal modelling and proof work in Phase 2 has been targetted at validating security critical algorithms in the filter design. This focus has been achieved using methods for developing formal partial proofs which could be extended to yield a complete formal verified detailed design.

The assumptions of *Theorem5* show what still remains to be accomplished before a full formal proof would be complete. The discharge of these assumptions would also require completion of the detailed design.

The task would be to produce specifications for the remaining set of components such that:

- The assumptions of *Theorem5* are provably true of the specified components.
- The specifications are sufficiently concrete that they can form a basis for an informal correspondence argument showing correctness of the implemented software (as required by ITSEC).

7 SUMMARY AND CONCLUSIONS

The SWORD multi-level secure RDBMS has been subject to extensive formal modelling to validate the security aspects of the design, including the completion of a number of formal proofs about aspects of the specification and design.

This formal treatment has been valuable in providing precise specifications as a basis for implementation. The processing of these specifications by appropriate tools has been beneficial in removing minor errors, and the completion of formal proofs relating to these specifications has detected errors of a deeper nature.

The relevance of this kind of work to the evaluation of the system against the ITSEC criteria is of interest to DRA, and this report provides an overview of the formal methods work to assist in an assessment of its relevance to evaluation.