Retrenching the Purse: Finite Sequence Numbers, and the Tower Pattern

Richard Banach[1], Michael Poppleton[2], Czeslaw Jeske[1], and Susan Stepney[3]

[1] School of Computer Science, University of Manchester,
Manchester M13 9PL, UK,
{banach,cj}@cs.man.ac.uk
[2] Department of Electronics and Computer Science,
University of Southampton, Highfield,
Southampton SO17 1BJ, UK,
mrp@ecs.soton.ac.uk
[3] Department of Computer Science, University of York,
Heslington, York YO10 5DD, UK,
susan.stepney@cs.york.ac.uk

Abstract. The Mondex Electronic Purse system [18] is an outstanding example of formal refinement techniques applied to a genuine industrial scale application, and notably, was the first verification to achieve ITSEC level E6 certification. A formal abstract model including security properties, and a formal concrete model of the system design were developed, and a formal refinement was hand-proved between them in Z. Despite this success, certain requirements issues were set beyond the scope of the formal development, or handled in an unnatural manner.

Retrenchment is reviewed in a form suitable for integration with Z refinement, and is used to address one such issue in detail: the finiteness of the transaction sequence number in the purse funds transfer protocol. A retrenchment is constructed from the lowest level model of the purse system to a model in which sequence numbers are finite, using a suitable elaboration of the Z promotion [21] technique. We overview the lifting of that retrenchment to the abstraction level of the higher models of the purse system. The concessions of the various retrenchments generated, formally capture the dissonance between the unbounded sequence number idealisation and the bounded reality. Reasoning about when the concession can become valid influences the actual choice of sequence number bound. The retrenchment-enhanced formal development is proposed as an example of a widely applicable methodological pattern for formal developments of this kind: the *Tower Pattern*.

1 Introduction

The Mondex Electronic Purse [18], produced by the NatWest Development Team, is a system of Smartcard-based electronic purses carrying currency for electronic commerce applications. Clearly, this is a security-critical application. For this reason, the developers of Mondex (formerly a part of NatWest Bank, lately NDS Group), employed state of the art methods to ensure the implementation was as robust as possible. At the time of its creation (in the late 1990s), the Mondex Purse achieved an ITSEC [14] rating

of E6. This requires a formal abstract model, a formal concrete model and a nontrivial refinement between them, formally proved to be correct. This is the the highest possible ITSEC level (corresponding these days to a Common Criteria EAL7 rating), and the development was a trailblazer for showing that fully formal techniques could be applied within realistic time and cost limitations to industrial scale applications.

The abstract model of the Mondex Purse system describes a world of purses which exchange value through atomic transactions, and specifies the security properties: purse authentication, preservation of overall system value, and correct processing of both transferred value and lost value. The concrete design model describes a distributed system of purses, transferring value via an insecure and lossy medium using an n-step protocol. Security features are implemented locally on each purse. In the field the purse is self-sufficient, recording any lost value from failed transactions locally, for intermittent central logging.

The Mondex Purse verification of the security properties remains an impressive achievement, both as a landmark industrial case study, and as a contribution to the theory of refinement. The separation between the abstract and concrete levels is significant, in a logical as well as a functional sense. The refinement is a composition of two simpler refinements, a "backward" refinement to, and a "forward" refinement from, an intermediate "between" model. [17] gives a readable account of how the then existing forward refinement rules in Z were insufficient to prove refinement, and how certain backward rules from the more general theory of refinement, e.g. [9], had to be implemented in Z, in order to deliver the two-stage proof. The clue to the need for this was the fact that the concrete, n-step value transfer protocol resolved certain non-determinism later than the abstract system; this is an instance of a classical counterexample by Milner [9] showing the incompleteness of the forward rules as a proof method for refinement.

Nevertheless, the necessity of having a refinement, meant that a number of requirements issues, legitimately the concern of the formal development, had to be passed over in silence, since they would strictly speaking have broken the validity of the refinement had they been incorporated in the models that were used. One can argue that curtailing the ideal scope of the refinement to some extent *always* happens: for example, it is never practical to prove refinement all the way to the physical hardware. The refinement might be pushed to source code level, then to machine code (if the compiler is not trusted); if that were insufficient one could try to refine down to the hardware design and even to the physics of the constituent devices.

Retrenchment [3, 4] has been proposed as a theory that generalizes refinement, essentially in allowing the refinement relation (classically, an invariant) to be weakened in the postcondition by a defined *concession* clause. This is inevitably a more intricate theory, offering less than the simulation property of refinement, unless extra application-specific assumptions are made. It was motivated originally by the impossibility of refining infinite to finite types, or the continuous variables of real-world physical models to discrete ones. Further work has revealed the utility of retrenchment both as anticipated [15], and as a vehicle for the flexible layering in of contrasting, even conflicting requirements in a formal development [5].

We regard the requirements issues identified in the Mondex verification – those set beyond the scope of the formal development, or handled in an unnatural manner – as

"retrenchment opportunities". The aim of this paper is to show that by incorporating retrenchment as a formal transformation of models, one can broaden the scope and accuracy of the formal modelling in a manner sympathetic to the existing refinement-based development. This yields a way of getting the best of both worlds: the clarity and rigour of the original refinement-based development, without an artificial denial of the existence of the attendant other issues.

The rest of the paper is structured as follows. In section 2 we give an overview of the Mondex development, and identify the requirements issues that motivate the application of retrenchment. Section 3 reviews the proof rules for refinement and retrenchment in a Z setting. Section 4 focuses on one of these aspects, the finiteness of the sequence number. A retrenchment is defined between the concrete purse and a new purse model, identical to the former in all but making the sequence number finite. This retrenchment is then extended to the world of purses by a suitable adaptation of the Z promotion used in [18]. Section 5 overviews how this retrenchment of the concrete model of the Purse system can be lifted to the abstract model of the system. Section 6 gives a probabilistic validation of the lifted retrenchment, assessing the risk of the purse sequence number breaching its bound, deriving an acceptable value for the bound thereby. Section 7 concludes and recapitulates. It is observed that the structure of refinements and retrenchments derived in the present paper is more widely applicable than just the present work, and we elevate it to status of a generally applicable methodological pattern for widening the remit of formal developments using retrenchments: the *Tower Pattern*.

2 The Mondex Purse: from Refinement to Retrenchment

The Mondex Electronic Purse described in [18] consists of three models: A(abstract), B(between), and C(concrete). The A model is a highly abstract expression of atomic value transfer between purses, allowing an atomic notion of loss in transit. It is a model targetted purely at the security properties of the system; it does not capture all the many other system requirements. Model B captures the elements of the value transfer protocol, and is thus nonatomic; it is also enhanced with extra structure and constraints needed to achieve a backward refinement from model A. Model C is model B without the extra structure and constraints. These can be established by an induction on the length of the execution, leading to a forward refinement between models B and C. It is thus shown that model C is a refinement of model A.

We have indicated that [18] is a development of the security properties of the Mondex Purse, not a full specification of the system. Even within these limitations, some requirements aspects, in principle deserving to be included within the formal development, were omitted or handled unnaturally in modelling, in order to establish the refinement between models. One of the aims of this paper is to show that by incorporating retrenchment into the formal development armoury, the tension that arises about whether some feature should be included or not in the refinement-based development is eased. This is because versions with and without the feature may be formally related via a retrenchment and the development paths with and without the feature may be drawn together, in part automatically. Here then is a brief summary of the Mondex "retrenchment opportunities":

– Sequence Number: The integrity of the protocol depends partly on the sequence number of the transaction in progress. Sequence numbers occur in the B, C models where they are naturals; in reality they are bounded numbers.
– Log Full: Transfers completing abnormally are logged by purses. The concrete model implements the abstract "lost value" component in terms of an off-card exception archive into which purses' log contents are saved. A purse needs to be assured that the data is safely in the archive before it can clear it from its own, highly constrained, log memory. Logs occur in the B, C models where they are unbounded; in reality they are finite.
– Hash Function: Clearing a purse's log after its contents are centrally archived is done via a message containing a "clear" code. The purse log contents are assumed to be in total injective correspondence with the clear codes, as that property is required in the proof. In reality of course a cryptographic hash function is used, which is neither total, nor injective, but is informally argued to be "sufficiently injective".
– Balance Enquiry: Each purse has a balance enquiry operation. If this is invoked at a particular point in the middle of a B model value transfer, a discrepancy can occur between the model A and model B balances due to differences in where nondeterminism is resolved in the two models. This is handled formally by a modelling trick, using finalisation instead of the enquiry operation to observe the state.

In this paper we focus on the sequence number in detail, leaving the others to be explored elsewhere. Our strategy is to build a tower of models D, E, F which retrench C, B, A respectively, and so that the obvious refinement/retrenchment squares commute. See Fig. 1 which shows the models, the epithets that accompany them, and their interrelationships. Since the variables involved in the sequence number retrenchment do not appear in the A model, if we take a sufficiently "noninvasive" approach to the construction of model D, it turns out that models A and F can subsequently be identified, though this is a fragile property.

Briefly, model D retrenches model C to take into account the boundedness of actual sequence numbers. Model D is then lifted to the abstraction level of model B, yielding model E; this is essentially model D but with the additional invariants. Noting that model E is refinable from model A, yields model F (the top of the tower) as a copy of model A.

3 Refinements and Retrenchments

In this section we briefly review the notions of refinement and retrenchment used in this paper. For refinement, we adopt the formulation in [8] as used in the Mondex development. We give only the forward rules for refinement, since these formed the basis for the definition of retrenchment [3]; we do not need to consider the backward rules further here. The nomenclature in our definitions will be in line with that needed for the various models in our discussion of the Mondex development below.

The A to B refinement is a backward refinement. The B to C refinement is a forward refinement. We call the between model B "abstract" in this context; it is given

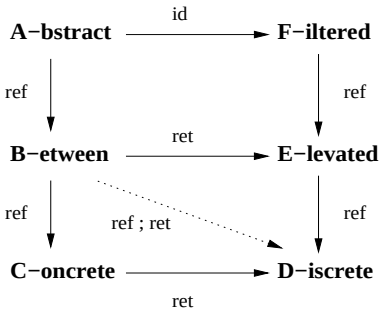

Fig. 1. A development pattern for refinement with retrenchment

by the ADT $(B, BInit, \{BOp \mid BOp \in \mathsf{Ops}\})$, and the concrete model C is given by the ADT $(C, CInit, \{COp \mid COp \in \mathsf{Ops}\})$. So schemas B, C give the abstract and concrete state spaces, and the corresponding per-operation I/O spaces are given by schemas BI_{Op}, BO_{Op} and CI_{Op}, CO_{Op}. We assume a retrieve relation $R_{BC} : [B; C]$ between the two state spaces, and for each operation Op, input and output mapping relations $RI_{BC,Op} : [BI_{Op}; CI_{Op}]$ and $RO_{BC,Op} : [BO_{Op}; CO_{Op}]$. Forward refinement is given by three proof obligations (POs), *initialization, applicability* and *correctness*:

$$\forall C' \bullet CInit \Rightarrow \exists B' \bullet BInit \wedge R'_{BC} \tag{1}$$

$$\forall B;\ BI_{Op};\ C;\ CI_{Op} \bullet R_{BC} \wedge RI_{BC,Op} \wedge \mathrm{pre}\, BOp \Rightarrow \mathrm{pre}\, COp \tag{2}$$

$$\begin{aligned}\forall B;\ BI_{Op};\ C;\ CI_{Op};\ C';\ CO_{Op} \bullet R_{BC} \wedge RI_{BC,Op} \wedge \mathrm{pre}\, BOp \wedge COp \\ \Rightarrow \exists B';\ BO_{Op} \bullet BOp \wedge R'_{BC} \wedge RO_{BC,Op}\end{aligned} \tag{3}$$

Note that (1)-(3) do not mention finalisation. We deal with the issue of observation, and specifically of relating the outputs of the abstract and concrete models (normally handled via finalisation) "on the fly", in line with the tack taken in retrenchment.

The C to D development step is a forward retrenchment. For this, the abstract model is the C ADT, and the concrete model is given by ADT $(D, DInit, \{(DOp, DI_{Op}, DO_{Op}) \mid Op \in \mathsf{Ops}\})$. Similar notational conventions apply. The retrenchment is given by firstly a *retrieve* relation $R_{CD} : [C; D]$ between the state spaces; and secondly we have the within, output and concedes relations on a per-operation basis. The *within* relation is between the input-state spaces $W_{CD,Op} : [CI_{Op}; C; DI_{Op}; D]$. The *output* and *concedes* relations are normally defined over both full input-state-output frames with types $O_{CD,Op}; C_{CD,Op} : [CI_{Op}; C; C'; CO_{Op}; DI_{Op}; D; D'; DO_{Op}]$, though in practice, we often omit such parts of these signatures as are not needed. We call these three relations the *retrenchment data*.

Two POs define a retrenchment between two models: *initialisation* as for refinement (1), and *correctness* which is analogous to refinement correctness (3); note that applicability issues are understood to be subsumed in (5) via the within relation:

$$\forall D' \bullet DInit \Rightarrow \exists C' \bullet CInit \wedge R'_{CD} \tag{4}$$

$$\forall C;\ CI_{Op};\ D;\ DI_{Op};\ D';\ DO_{Op} \bullet R_{CD} \land W_{CD,Op} \land DOp$$
$$\Rightarrow \exists C';\ CO_{Op} \bullet COp \land ((R'_{CD} \land O_{CD,Op}) \lor C_{CD,Op}) \quad (5)$$

4 The Sequence Number Retrenchment

The starting point for the sequence number retrenchment is the C model of the world of purses. We concentrate on just the simplest operation of [18] that is nontrivially affected, the single purse *CConPurse* operation *CIncreasePurseOkay*. This is an abstraction (for an individual purse and at C level) of a number of lower level operations that do or do not need to increment the purse transaction sequence number. It is implicit that sequence number *CnextSeqNo* is a natural, thus unbounded.

We retrench *CIncreasePurseOkay* to operation *DIncreasePurseOkay* of new D model purse *DConPurse*[1], where *DConPurse* 'is as' *CConPurse* apart from *DnextSeqNo*, which is of finite type $BN == 0..BIGNUM$. We assume the usual arithmetic operations for *BN* defined by restriction, where we liberally coerce when necessary, and where *BIGNUM* is a matter of implementation choice. *CConPurseIncrease* hides the C purse sequence number *CnextSeqNo*, thus $\Xi CConPurseIncrease$ denotes skip, i.e. no change on all state apart from *CnextSeqNo*. *DConPurseIncrease* 'is as' *CConPurseIncrease*.

$$CConPurseIncrease == CConPurse \setminus (CnextSeqNo)$$

CIncreasePurseOkay

$\Delta CConPurse$
$Cm?, Cm! : CMESSAGE$

$\Xi CConPurseIncrease$
$CnextSeqNo' \geq CnextSeqNo$
$Cm! = \bot$

DIncreasePurseOkay

$\Delta DConPurse$
$Dm?, Dm! : DMESSAGE$

$\Xi DConPurseIncrease$
$(DnextSeqNo < BIGNUM \Rightarrow$
$\quad DnextSeqNo' \geq DnextSeqNo \land$
$\quad Dm! = \bot)$
$(DnextSeqNo = BIGNUM \Rightarrow$
$\quad DnextSeqNo' = DnextSeqNo \land$
$\quad Dm! = DpurseBlocked\ Dname)$

$\bot$ is a general purpose message used in Mondex which is of-no-concern here, and *DpurseBlocked Dname* is a special message emitted when *BIGNUM* is reached, identifying the purse in question, *Dname*.

The above constitutes a minimally invasive retrenchment of *CIncreasePurseOkay*, in that not only is it the case that inside a guard the C model behaviour is preserved, but even outside the guard there are no new purse states to threaten the validity of the reasoning about the refinements in [18]. Far more aggressive D model designs are obviously possible.

[1] Z employs a convention of pre-capitalizing only the names of types (schema and other): we augment this convention by prefixing a single character $A, B, \ldots$ to a name as required, to denote the model in question. Thus *CThing* is a schema or other type in the C model, whereas *Dthing* is a variable, usually a schema component, in the D model. A further lexical schema convention we employ, to save space, is to say *DSchema* 'is as' *CSchema* to indicate that the text of *DSchema* can be generated from that of *CSchema* by replacing all *Cthing*s by *Dthing*s.

We come to the retrenchment data itself. *CDConPurseIncreaseEquality* is shorthand for equalities between corresponding C and D variables in *CConPurseIncrease* and *DConPurseIncrease* respectively. Under the vacuous *within* $W_{CD,IncreasePurseOkay}$ constraint on before states and inputs, the pair of operation instances *CIncreasePurseOkay*, *DIncreasePurseOkay* establishes either $R'_{CD} \wedge O_{CD,IncreasePurseOkay}$ (*retrieve* and *output* relations), or $C_{CD,IncreasePurseOkay}$ (*concedes* relation). This concession states the possible inequality between C, D sequence numbers and uses the D level error message.

R_{CD}

$$CConPurse;\ DConPurse$$
$$CDConPurseIncreaseEquality$$
$$\hline$$
$$CnextSeqNo = DnextSeqNo$$

$W_{CD,IncreasePurseOkay}$

$$CConPurse;\ DConPurse$$
$$Cm? : CMESSAGE$$
$$Dm? : DMESSAGE$$

$O_{CD,IncreasePurseOkay}$

$$Cm! : CMESSAGE$$
$$Dm! : DMESSAGE$$
$$\hline$$
$$Cm! = Dm!$$

$C_{CD,IncreasePurseOkay}$

$$CConPurse';\ DConPurse'$$
$$CDConPurseIncreaseEquality'$$
$$Cm! : CMESSAGE$$
$$Dm! : DMESSAGE$$
$$\hline$$
$$CnextSeqNo' \geq DnextSeqNo'$$
$$Cm! = \bot$$
$$Dm! = DpurseBlocked\ Dname$$

4.1 Promotion of the Purse Retrenchment

We review the Z technique of promotion [21, 10] of a local-state (purse) to a global-state (world) operation. The global state schema, say *World*, is defined as an indexing function from some index set *Ind* to the space of all possible local state elements, these being given by schema *LS*. To enable concise world-level description of an operation working only a single copy of the local state, the promotion framing schema $\Phi LSOp$ is defined. $\Phi LSOp$ contains both a global state schema *World* and a local state schema *LS*, and also an input parameter $i?$ of type *Ind*, identifying the required local state element for access or update. An equality identifies the target *LS* element $f(i?)$ through the index function f with the local state binding θLS. The final predicate ensures that all elements other than $f(i?)$ remain unchanged.

$\Phi LSOp$ is generic insofar as it allows the mechanical definition of a world-level operation *WorldOp* corresponding to a local operation *LSOp* without constraining the behaviour of that local operation in any way:

World

$$f : Ind \pfun LS$$

$$WorldOp == \exists \Delta LS \bullet \Phi LSOp \wedge LSOp$$

$\Phi LSOp$

$$\Delta World$$
$$\Delta LS$$
$$i? : Ind$$
$$\hline$$
$$i? \in \mathrm{dom}\, f$$
$$\theta LS = f(i?)$$
$$f' = f \oplus \{i? \mapsto \theta LS'\}$$

The above is the classical, index-function-based form of promotion. Recently certain promotion *patterns* [19, 20] have been proposed for various forms of local-to-global structuring, some having been based on promotion use in Mondex.

As in the C world, individual D model purses are promoted to the D world of purses, as given in the schemas that follow. *DConWorld* 'is as' *CConWorld*. Beyond the (purse-NAME-)indexed map of purses, the world contains the *Dether* of all messages ever sent between purses, and the *Darchive* of all transaction exception logs uploaded from purses. There are two *DConWorld* constraints: we equate each internal purse name to its corresponding index, and we ensure each archive entry identifies its originating purse.

Promotion of the D model 'is as' that of the C model of [18]: ΦDOp 'is as' ΦCOp, where $Dm?, Dm!$ are the input and output messages to and from *DIncreasePurseOkay*. *DIncrease*, the promoted and wrapped operation, 'is as' *CIncrease*. N.B. *DIgnore* 'is as' *CIgnore*, and just skips at world level.

DConWorld

$$DconAuthPurse : NAME \pfun DConPurse$$
$$Dether : \mathbb{P}\, DMESSAGE$$
$$Darchive : \mathbb{P}\, DLogbook$$

$$\forall n : \dom DconAuthPurse \bullet (DconAuthPurse\ n).Dname = n$$
$$\forall nld : Darchive \bullet first\ nld \in \dom DconAuthPurse$$

ΦDOp

$$\Delta DConWorld;\ \Delta DConPurse$$
$$Dm?, Dm! : DMESSAGE$$
$$Dname? : NAME$$

$$Dm? \in Dether$$
$$Dname? \in \dom DconAuthPurse$$
$$\theta DConPurse = DconAuthPurse\ Dname?$$
$$DconAuthPurse' = DconAuthPurse \oplus \{Dname? \mapsto \theta DConPurse'\}$$
$$Darchive' = Darchive$$
$$Dether' \subseteq Dether \cup \{Dm!\}$$

$$DIncrease == DIgnore \lor (\exists \Delta DConPurse \bullet \Phi DOp \land DIncreasePurseOkay)$$

Having defined the D model, the next job is to promote the retrenchment of individual purse operations such as *CIncreasePurseOkay* to a retrenchment at the *CConWorld*-to-*DConWorld* level, between *CIncrease*, (not quoted but with the same syntax as) *DIncrease*. Any theory of promotion of retrenchments must be grounded in the promotion of refinements. A good treatment is given by [10], including presentation of a simple world-level retrieve relation resulting from the distribution of the local retrieve relation through promotion. We base our approach on this form. Given a retrieve relation R between local states *Abs* and *Conc*, the promoted retrieve relation R^P [10] between *AbsWorld* and *ConcWorld* (with index functions *Absf*, *Conf* respectively) simply asserts the local one for all local state elements:

$$
\begin{array}{|l}
\hline
R^P \\
AbsWorld;\ ConcWorld \\
\hline
\mathrm{dom}\, Concf = \mathrm{dom}\, Absf \\
\forall\, n : \mathrm{dom}\, Concf \bullet \exists\, R \bullet \theta Abs = Absf(n) \land \theta Conc = Concf(n) \\
\hline
\end{array}
$$

The promotion of retrenchments offers a choice of approaches, depending on what one wishes to emphasise. In [6] we explore this in some detail, but space limitations here do not permit us to show the full variety of possibilities on the present example. Instead, we apply just one of the approaches, perhaps the most interesting one: *precise promotion*.

The essential point is this. Let us imagine the system has been running for some time and that some or many elements have already engaged in operations. In terms of the retrenchment, some elements will be in the local state element retrieve relation R, while others may have already conceded (and so may no longer be in R). Assuming all elements are in R (as for refi nement) thus gives an unduly restricted syntactic picture of the correspondence between the dynamics of the abstract and concrete worlds. In precise promotion, we introduce an extra world variable *good* to keep track of which elements are doing what, regarding the retrenchment.

Since there are two worlds, there are two obvious places in which to put the extra variable, the abstract or the concrete world. For most retrenchments the concrete world is the most natural place to put the extra information, and we do so here; so the extra variable is *Dgood*. Moreover, for this to work effectively, we require a separability axiom (6) to hold for all common operations *Op*. Given a concrete D model step, $DEstRet^{PP}_{DOp}$/$DNotEstRet^{PP}_{DOp}$ assert the existence/non-existence respectively of an abstract C world step that witnesses the refi nement. Given a D step, $DEstCon^{PP}_{DOp}$ asserts the existence of a C step that witnesses the concession. The separability axiom is:

$$DEstRet^{PP}_{DOp} \land DEstCon^{PP}_{DOp} \Leftrightarrow \mathsf{false} \tag{6}$$

where

$$
\begin{array}{l}
DEstRet^{PP}_{DOp} == D;\ DI_{Op};\ D';\ DO_{Op} \mid DOp \land \\
\quad (\exists\, C;\ CI_{Op};\ C';\ CO_{Op} \bullet R_{CD} \land W_{CD,Op} \land COp \land (R'_{CD} \land O_{CD,Op})) \\
DEstCon^{PP}_{DOp} == D;\ DI_{Op};\ D';\ DO_{Op} \mid DOp \land \\
\quad (\exists\, C;\ CI_{Op};\ C';\ CO_{Op} \bullet R_{CD} \land W_{CD,Op} \land COp \land C_{CD,Op}) \\
DNotEstRet^{PP}_{DOp} == D;\ DI_{Op};\ D';\ DO_{Op} \mid DOp \land \\
\quad \neg(\exists\, C;\ CI_{Op};\ C';\ CO_{Op} \bullet R_{CD} \land W_{CD,Op} \land COp \land (R'_{CD} \land O_{CD,Op}))
\end{array}
$$

Given a C-to-D retrenchment (5), and axiom (6), it can be deduced from a given concrete step alone, whether R_{CD} is reestablished or $C_{CD,Op}$ holds. This allows the concrete promotion to accurately maintain the *Dgood* variable as follows.

We need suitable enhancements to: the promoted operations (which become DOp^{PP}), to the promoted world construction itself (which becomes $DConWorld^{PP}$), and to the framing schema (which becomes ΦDOp^{PP}). The latter differs from ΦDOp only in the

replacment of *DConWorld* by $DConWorld^{PP}$, so we do not reproduce it in full.[2]

$DConWorld^{PP}$

$DConWorld$

$Dgood : \mathbb{P}\, NAME$

$Dgood \subseteq \mathrm{dom}\, DconAuthPurse$

$$DIncrease^{PP} == DIgnore \lor \big(\exists\, \Delta DConPurse \bullet \Phi DOp^{PP} \land DIncreasePurseOkay \\ \land (DEstRet^{PP}_{DIncrease} \Rightarrow Dgood' = Dgood) \\ \land (DNotEstRet^{PP}_{DIncrease} \Rightarrow Dgood' = Dgood \setminus \{Dname?\})\big)$$

It is clear that $DIncrease^{PP}$ is a refinement of *DIncrease* via a retrieve relation that simply projects away *Dgood*, as $DIncrease^{PP}$ arises from *DIncrease* by the addition of *Dgood*, whose value is never used in the update of any *DIncrease* variable.

With these details in place, we can write down the precisely promoted retrenchment between the *CConWorld* and $DConWorld^{PP}$ *Increase* operations. For this, it is easy to see that (6) holds, in particular, by examining whether the output of *DIncrease* is $\bot$ or *DpurseBlocked Dname*?. For $\bot$, $R'_{CD} \land O_{CD,IncreasePurseOkay}$ is established by the identity of outputs, and the 'is as' identity of the seqence number predicates. For *DpurseBlocked Dname*?, $C_{CD,IncreasePurseOkay}$ is established by definition of the outputs and by the skip on *DnextSeqNo*.

The retrenchment below employs a *focused* pattern of precise promotion, in that the within, output, concedes relations only refer to the named local state element *Dname*?. Since the promoted operation acts on only one element, implicitly all other elements in *Dgood* maintain the local retrieve relation R'_{CD}. An *inclusive* pattern is also available which covers all *Dgood* elements, explicitly claiming R'_{CD} in the concession for the elements in $Dgood \setminus \{Dname?\}$; for brevity we present the focused pattern here. Since archive entries are tagged with the originating purse's name, we can identify those C/D archive subsets corresponding to purses in *Dgood*?, and we assume for simplicity that all messages in the ether are tagged with originator's and addressee's names as the first two fields of the message.[3] **"** CDnamedConPurseIncreaseEquality *name* **"** is (not legal Z, for brevity, but) shorthand for equalities of named other purses' of-no-concern data in the following:

---

[2] Note that there is a somewhat philosophical question regarding the nature of the *Dgood* variable: should it be viewed as a genuine system variable or not? In this paper we do not go beyond saying that the viability of the precise promotion's using *Dgood*, attests to the ability of the concrete model's being able to keep track of the retrieving elements should it so choose.

[3] Note that this is a considerable simplification compared to [18]. In [18] it is the case that: (i) the models do not concern themeselves with details of physical message transmission, (ii) the relevant data can nevertheless be inferred indirectly from the contents of the message body.

R^{PP}_{CD}

$CConWorld;\ DConWorld^{PP}$

$\mathrm{dom}\, CconAuthPurse = \mathrm{dom}\, DconAuthPurse$

$\forall\, Dnm : Dgood \bullet$

$\quad (CconAuthPurse\ Dnm).CnextSeqNo = (DconAuthPurse\ Dnm).DnextSeqNo$

$\quad \wedge$ “ CDnamedConPurseIncreaseEquality Dnm ”

$Dgood \lhd Carchive = Dgood \lhd Darchive$

$(Dgood \times Dgood) \lhd Cether = (Dgood \times Dgood) \lhd Dether$

$W^{PP}_{CD,Increase}$

$CConWorld;\ DConWorld^{PP}$

$Cm? : CMESSAGE$

$Dm? : DMESSAGE$

$Cname?, Dname? : NAME$

$Cname? = Dname?$

$Cname? \in Dgood$

$O^{PP}_{CD,Increase}$

$\Delta DConWorld^{PP}$

$Cm! : CMESSAGE$

$Dm! : DMESSAGE$

$Dname? : NAME$

$Dgood' = Dgood$

$Cm! = Dm!$

$C^{PP}_{CD,Increase}$

$CConWorld';\ \Delta DConWorld^{PP}$

$CDConPurseIncreaseEquality'$

$Cm! : CMESSAGE$

$Dm! : DMESSAGE$

$Dname? : NAME$

$Dgood' = Dgood - \{Dname?\}$

“ CDnamedConPurseIncreaseEquality $Dname?$ ”

$(CconAuthPurse'\ Cname?).CnextSeqNo \geq (DconAuthPurse'\ Dname?).DnextSeqNo$

$Cm! = \bot$

$Dm! = (DpurseBlocked\ Dname?)$

$Dgood' \lhd Carchive' = Dgood' \lhd Darchive'$

$(Dgood' \times Dgood') \lhd Cether' = (Dgood' \times Dgood') \lhd Dether'$

5 Lifting the Retrenchment

The previous section described in fair detail how, despite its awkwardness, the real world finiteness of the sequence number can be taken account of, in a model that could be appended to the preexisting development. In this section we sketch rather briefly how this new D model can be related to the other models in the Mondex development, clarifying the relationship between sequence number finiteness and the concerns of these higher level models.

Essentially, the level of abstraction of the D model is first lifted to the level of the B model (this giving the E model) and then it is observed that there is a refinement

from the A model to the E model, due to the nonintrusiveness of the D model. So the construction of the F model becomes just a rebadging of the A model. See Fig. 1.

The lifting of the D model to the E model makes use of a generic construction [1] for lifting the concrete model of a retrenchment to the level of abstraction of the retrenchment's abstract system; the model generated, typically called U, then refines to the retrenchment's concrete system. This generic construction builds U out of the two original systems in the retrenchment. The required level of abstraction is defined indirectly via a collection of properties specific to the construction, and U captures this level by being refinable to any system that also enjoys these properties. Thus U is the most abstract such system. As far as the construction goes, any suitable system interrefinable with U is just as good as U, so we have the option of replacing U with something more convenient if we wish.

In the Mondex case we build the E model, which matches the level of abstraction of the B model. The retrenchment that we are lifting is the composition of the B to C forward refinement and the C to D retrenchment, such compositions themselves being a matter for careful definition; see [2] for details.

For clarity and simplicity let us examine how this works for the individual purse operation *IncreasePurseOkay*. Essentially, for the *IncreasePurseOkay* operation of the generated U system we have:

protoEIncreasePurseOkay

$$\begin{array}{l} BIncreasePurseOkay;\ \Delta DConPurse \\ Dm?, Dm! : DMESSAGE \\ \hline (R_{BD} \wedge R'_{BD} \wedge O_{BD,IncreasePurseOkay}) \vee (R_{BD} \wedge C_{BD,IncreasePurseOkay}) \end{array}$$

In the above, *BIncreasePurseOkay* 'is as' *CIncreasePurseOkay*, and R_{BD} 'is as' R_{CD}. Similarly $O_{BD,IncreasePurseOkay}$ 'is as' $O_{CD,IncreasePurseOkay}$, and $C_{BD,IncreasePurseOkay}$ 'is as' $C_{CD,IncreasePurseOkay}$. In *protoEIncreasePurseOkay*, *BIncreasePurseOkay* contributes the steps of the B model and $\Delta DConPurse$ contributes all legal D changes of state. The B-to-D retrenchment tells us that any *DIncreasePurseOkay* step satisfies the retrenchment correctness PO in terms of some witnessing B-step. In *protoEIncreasePurseOkay* it is clear that precisely the same witness establishes the E-to-D refinement correctness PO.

We note that there is considerable duplication of state and other information in *protoEIncreasePurseOkay*; the B and D parts of the state say practically the same thing via the *BDConPurseIncreaseEquality* in R_{BD} and R'_{BD}, and the I/O is similarly either irrelevant or discernable from the D element alone.

Since, as noted above, it is sufficient to fix on a system that is interrefinable with what the construction routinely generates, it is worth reflecting on the details of the U system, to see if the duplication can be avoided. Examining the details reveals that we can replace *protoEIncreasePurseOkay* with the simpler:

EIncreasePurseOkay 'is as' *DIncreasePurseOkay*

a welcome simplification, attributable to the nonintrusive nature of our D construction.

Of course our real focus of interest is on *DIncrease* and its lifting to *EIncrease*. The single purse operation just treated provides an indication of what to expect, in that the *IncreasePurseOkay* lifting should be discernable within the *Increase* one.

The B world *Increase* operation has the same shape as the D world one:

$$BIncrease == BIgnore \lor (\exists \Delta BConPurse \bullet \Phi BOp \land BIncreasePurseOkay)$$

The subtlety here is that in *BIgnore* and ΦBOp, instead of $\Delta BConWorld$ (as would be expected) we have $\Delta BetweenWorld$, where *BetweenWorld* features additional structure and constraints imposed on *BConWorld* in order to enable the A-to-B backward refinement to cary through. Aside from this, the constituents of *BIncrease* 'are as' their corresponding *CIncrease* ones.

For lack of space, the reader will have to take our word for it that the constraints in *BetweenWorld* do not materially affect our discussion; they express the consistency between the cryptographically protected messages in the ether and the purses' states; doubters can refer to [18]. We now retrace the earlier lifting construction and obtain:

protoEIncrease

$BIncrease;\ \Delta DConWorld^{PP}$

$Dm?, Dm! : DMESSAGE$

$(R^{PP}_{BD} \land R'^{PP}_{BD} \land O^{PP}_{BD,Increase}) \lor (R^{PP}_{BD} \land C^{PP}_{BD,Increase})$

It turns out that we can argue as before and replace *protoEIncrease* by *EIncrease* where:

$EIncrease$ 'is as' $DIncrease^{PP}$

except that *DetweenWorld* (which now 'is as' *BetweenWorld*) replaces occurences of *DConWorld* in *DIgnore* and ΦDOp in $DIncrease^{PP}$. Thus $DIncrease^{PP}$ is at the right level of abstraction after all, again due to the minimalist nature of the D construction.

Having dealt with the E model, the final step consists of observing that there is a (backward) refinement from the A model to the E model. The D level purse blocking behaviour when the sequence number overflows is simulated at A-level by the purse skipping; the A world has no sequence numbers. Aside from the fact that the details of this are beyond the scope of this paper, some points are worth making. Firstly, this is *not* a further instance of the lifting construction just used to build the E model. Secondly, the truth of it depends rather delicately on a suitable choice of retrieve and output relations, not to mention the precise notion of refinement employed and of course the minimalist nature of the D construction. Thus it is not a robust property, though it is a very pleasing one.

6 Validating the Retrenchment

In the preceding sections, we have designed the D model to do nothing useful once the limit on the sequence number has been reached. Since doing nothing is unlikely to satisfy users, it is incumbent on us to validate this design in the light of wider system requirements, which we do in this section. The argument now swings to showing that the

limit in fact never arises. This can be crystallised as saying the concession of the relevant retrenchment does not become true within the lifetime of the use of the product.[4]

The validation of the concession of the C to D retrenchment depends on the value of BIGNUM, a quantity we have hitherto left unspecified. Our analysis will generate a value for BIGNUM leading to acceptable overall system properties. Note that the lifted E model's dependence on BIGNUM is like that of the D model's so we can focus on just the C to D retrenchment. Roughly speaking, we want to know how long it will take before BIGNUM is reached, which we analyse as follows.

First of all, the increments of the sequence number are not deterministic, to prevent the values of the sequence number being exploited as a covert channel in any potential cryptographic attack. Thus the increments are random variables drawn from a probability distribution Θ. Let us say that Θ has a mean μ and variance σ both about 10. From here there are two approaches, the naive and the sophisticated.

In a naive approach, we expect the accumulated total sequence number after n trials to be approximately $n\mu$. Now consider the determined shopper, making the order of 100 transations per day using the purse, resulting in a daily sequence number increment of about 10^3. Taking a year to be about 10^3 days, leads to an approximate annual sequence number increment of about 10^6. On this basis, we can estimate how different choices of BIGNUM fare against the requirement that the BIGNUM limit is never in fact reached.

Suppose BIGNUM is about 2^{16} which is about 64×10^3. The limit is encountered within a couple of months, so this value of BIGNUM is clearly unsatisfactory. Similarly, choosing 2^{64} for BIGNUM gives a limit of about 16×10^{12} years, which is a *little* more conservative than necessary.

Suppose then that BIGNUM is about 2^{32} which is about 4×10^9. Dividing by 10^6 shows that the limit is reached in about 4000 years. Putting aside considerations of whether the purse will physically withstand that much use, it is certainly the case that the financial system underpinning the purse will have collapsed by that time. So a 32 bit BIGNUM provides plenty of room for even determined use, while safeguarding against overflow, and while still not being ridiculously overconservative.

Of course one can take a more cautious approach than the above, supposing that a determined attacker will go all out to breach the sequence number limit by subjecting the purse to as many transactions as it is possible to invoke, potentially leading to different estimates. Then again, it is hard to see what such an attacker stands to gain by disabling the purse in this way, locking in the value he has managed to put into it, since the system's security properties ensure that every purse operation leaves the whole system in a state that is at best equitable, at worst in the bank's favour.

Let us now turn to a more sophisticated treatment of the same situation. We note that the individual increments of the sequence number are the "arrivals" of a renewal process [11, 16, 13]. Thus if δSN_n is the n'th increment, then as n varies, we are interested in the behaviour of the random variables:

$$nextSeqNo_n = \delta SN_0 + \delta SN_1 + \ldots + \delta SN_n$$

[4] Note how the retrenchment framework has produced specific objects within the formal models, namely the concessions, that carry the information pertaining to the undesired state of affairs. A purely refinement based approach to the development could say nothing about such matters, disconnecting the formal world from the requirements level validation needed beyond.

In particular, we are interested in the random variable $N(t)$ given by:

$$N(t) = \max\{n \mid nextSeqNo_n \leq t\}$$

whose distribution describes how many increments of the sequence number are needed to reach the value t. Fortunately this is all standard material that can be found in loc. cit. The first order theory of $N(t)$ says that as t tends to infinity, $N(t)$ tends to the constant distribution t/μ almost surely. Furthermore the mean of $N(t)$ tends to the number t/μ. This agrees with the values obtained naively, and in particular, for a 32 bit BIGNUM, we again derive an overflow time of four thousand years.

To ensure the random characteristics of the situation do not lead to gambler's ruin type outcomes, we check out also the second order theory of renewals. This says that as t tends to infinity,

$$\frac{N(t) - t/\mu}{\sqrt{t\sigma^2/\mu^3}}$$

converges in distibution to $N(0,1)$, the standard normal distribution. This in turn means that the variance of $N(t)$ itself scales to $(t\sigma^2/\mu^3)^{\frac{1}{2}}$. When the numbers are substituted, this is of the order of a week or two. So in the end, the sophisticated story fully supports the naive one.

7 Conclusions, and the Tower Pattern

Above, we briefly reviewed the Mondex development and its "retrenchment opportunities." We then took the purse sequence number and showed how a more faithful treatment could be integrated with the existing refinement based development. The result was the collection of models related by refinements and retrenchments shown in Fig. 1.

One of the advantages of the retrenchment approach in dealing with model evolution situations, which the sequence number case study can be viewed as, is that it fits naturally with the idea that such evolutions often tend to be focused on judicious changes to one or more operations. In the limit, we can consider the change in each operation as a separate evolution step, expressed using a separate retrenchment, and compose them, e.g. as per [2]. For lack of space, we have not pursued this aspect here.

Note that Fig. 1 is a commutative diagram. Therefore it can be navigated in different ways with equivalent effect. For example, one can (as we did) start at the bottom of the tower, build the bottommost retrenchment, and build towards the top (it turned out that with a judicious choice of bottommost model, the top level blended seamlessly with the existing development). Alternatively one can start with the topmost retrenchment (an identity in our case), and proceed downwards, utilising different but compatible algebraic results on the combination of refinements and retrenchments [12]. This raises the general structure embodied in Fig. 1 to the level of a broadly applicable *pattern* for the deployment of retrenchment as a means of, (on the one hand) reconciling real world detail with an idealised but more transparent refinement development, or (on the other) propagating a top level requirements change down through a refinement stack, towards implementation. Of course, middle out deployments are also compatible with Fig. 1.

Note that the structure in Fig. 1 remains equally useful *regardless of the specific requirements issue(s) handled by the retrenchments that comprise it*, which lie buried in the details of the various retrenchment data. Its elevation to a methodological generality, the *Tower Pattern*, is therefore eminently justified.

References

[1] R. Banach. Maximally abstract retrenchments. In *Proc. IEEE ICFEM2000*, pages 133–142, York, August 2000. IEEE Computer Society Press.

[2] R. Banach, C. Jeske, and M. Poppleton. Composition mechanisms for retrenchment. 2004. submitted, http://www.cs.man.ac.uk/~banach/some.pubs/Retrench.Composition.pdf.

[3] R. Banach and M. Poppleton. Retrenchment: An engineering variation on refinement. In D. Bert, editor, *2nd International B Conference*, volume 1393 of *LNCS*, pages 129–147, Montpellier, France, April 1998. Springer.

[4] R. Banach and M. Poppleton. Sharp retrenchment, modulated refinement and simulation. *Formal Aspects of Computing*, 11:498–540, 1999.

[5] R. Banach and M. Poppleton. Retrenching partial requirements into system definitions: A simple feature interaction case study. *Requirements Engineering Journal*, 8(2), 2003. 22pp.

[6] R. Banach, M. Poppleton, and C. Jeske. Retrenchment and promotion in Z. submitted for publication, 2004.

[7] D. Bert, J.P. Bowen, S. King, and M. Waldén, editors. *Proc. ZB2003: Formal Specification and Development in Z and B*, volume 2651 of *LNCS*, Turku, Finland, June 2000. Springer.

[8] D. Cooper, S. Stepney, and J. Woodcock. Derivation of Z refinement proof rules. Technical Report YCS-2002-347, University of York, 2002.

[9] W.-P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge University Press, 1998.

[10] J. Derrick and E. Boiten. *Refinement in Z and Object-Z*. FACIT. Springer, 2001.

[11] G. Grimmett and Stirzaker D. *Probability and Random Processes*. O.U.P., 3 edition, 2001.

[12] C. Jeske. *Algebraic Integration of Retrenchment and Refinement*. PhD thesis, University of Manchester, 2005.

[13] S. Karlin and H.M. Taylor. *A First Course in Stochastic Processes*. Academic, 1975.

[14] Department of Trade and Industry. Information Technology Security Evaluation Criteria, 1991. http://www.cesg.gov.uk/site/iacs/itsec/media/formal-docs/Itsec.pdf.

[15] M. Poppleton and R. Banach. Controlling control systems: An application of evolving retrenchment. In D. Bert, J.P. Bowen, M.C. Henson, and K. Robinson, editors, *Second International Conference of B and Z Users*, volume 2272 of *LNCS*, pages 42–61, Grenoble, France, January 2002. Springer.

[16] S.L. Resnick. *Adventures in Stochastic Processes*. Birkhauser, 1992.

[17] S. Stepney, D. Cooper, and J. Woodcock. More powerful Z data refinement: Pushing the state of the art in industrial refinement. In J.P. Bowen, A. Fett, and M.G. Hinchey, editors, *11th International Conference of Z Users*, volume 1493 of *LNCS*, pages 284–307, Berlin, Germany, September 1998. Springer.

[18] S. Stepney, D. Cooper, and J. Woodcock. An electronic purse: Specification, refinement and proof. Technical Report PRG-126, Oxford University Computing Laboratory, 2000.

[19] S. Stepney, F. Polack, and I. Toyn. An outline pattern language for Z. In Bert et al. [7], pages 2–19.

[20] S. Stepney, F. Polack, and I. Toyn. Patterns to guide practical refactoring. In Bert et al. [7], pages 20–39.

[21] J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof*. Prentice-Hall, 1996.