A case-study in timed refinement: A mine pump

Brendan P Mahony
Ian J Hayes
Department of Computer Science
University of Queensland

June 1992

Abstract

A specification and top-level refinement of a simple mine pump control system, as well as a proof of correctness of the refinement, are presented as an example of the application of a formal method for the development of time-based systems. The overall approach makes use of a refinement calculus for timed systems, similar to the refinement calculi for sequential programs.

The specification makes use of topologically continuous functions of time to describe both analogue and discrete properties of both the system and its refinements. The basic building block of specifications is a specification statement, similar to Morgan's specification statement for sequential programs. It gives a clear separation between the specification of the assumptions that the system may make about the environment in which it is to be placed, and the effect the system is guaranteed to achieve if placed in such an environment.

The top-level refinement of the system is developed by application of refinement laws that allow design decisions to be made, local state to be introduced, and the decomposition of systems into pipe-lined and/or parallel processes.

Index Terms — Real-time specification, refinement calculus, Z, continuous functions, modal operators.

1 Introduction

Formal methods for the development of sequential programs have been available for some time, and more recently we have seen the development of refinement calculi for sequential programs [1, 14, 15]. An important feature of refinement calculi is the integration of specification and programming constructs into the same framework, so that

a smooth transition can be made from a top-level specification down to implementation detail.

The objective of this paper is to present a simple but realistic example of the use of a refinement calculus for time-based systems. We present a case study of the specification, analysis and top-level design of a mine monitoring and control system. In specifying such a system, we need to be able to discuss both continuous quantities, such as the water level in the mine, and discrete properties, such as whether or not the water pump is on. The approach taken models both continuous and discrete properties as functions of time within the same framework, and provides for the expression of relationships between these — usually separate — aspects of a system. The approach taken to specification is novel from a computer science viewpoint, but is, however, not completely new: it follows the approach to modelling systems that has been used by physicists and engineers for quite some time.

Another important aspect of the approach is that a clear distinction is made between the *assumptions* made about the environment of a system (or a component of a design) and the *effect* the system (or component) is expected to achieve. This separation allows for the use of a timed refinement methodology for the development and verification of successive design steps. This separation has the same advantages for timed specifications as the separation of pre-condition and post-condition in the specification of sequential programs.

Our example of a mine monitoring and control system is adapted from [2]. In that paper Burns and Lister elucidate an architectural framework for timely and reliable distributed information systems (TARDIS). The present paper is our attempt to formalise the specification and top-level design components of that architecture in a way which includes the timing properties of the system.

1.1 Specification

We have attempted to develop as general a specification of the mine monitoring and control system as possible, without biasing the system towards a particular form of implementation. Further we have developed a whole-system specification, rather than just a specification of the software components of the system. We consider the division of the implementation into hardware and software components to be part of the top-level design of the system, and that the top-level design process should be supported by our methodology.

For example, an implementation may use either a software monitor to regularly sample the water level in the mine, or it may use hardware to trigger an interrupt when the water level in the mine exceeds a critical level. A top-level specification of the whole system should allow both of these implementation strategies, and the design process should make it possible to calculate design parameters such as the maximum time between successive samplings of the water level.

In our specifications we make use of the Z mathematical and schema notations. A glossary of terminology used in this paper is provided in Appendix A. The reader is referred to [19, 5] for more detail on Z. We extend the Z notation with three (orthogonal) notational devices to facilitate the specification of real time systems:

- *units* (including dimensionality) of (physical) quantities,
- *(topologically) continuous functions* of both analogue and discrete quantities, and
- *specification statements* which specify the assumptions a process makes about its environment and the effect it is guaranteed to achieve.

We give an introduction to these techniques in Sections 1.2, 1.3, and 1.4.

1.2 Units

In specifying variables we follow the approach, commonly used in physics and engineering, of giving the (physical) units of the quantity. For example, the depth of the water in the mine is given in metres, so we introduce a type *DEPTH* which is a real number ($\mathbb{R}$ is the set of reals) with units of metres:

$$DEPTH \mathrel{\widehat{=}} \mathbb{R}[Metre].$$

Similarly, we introduce a type $TIME$ which is a real number with units of seconds:

$$TIME \mathrel{\widehat{=}} \mathbb{R}[Seconds].$$

A variable giving the depth at which the water level becomes dangerous can be declared using type *DEPTH*:

$$DangerH_2O : DEPTH.$$

As another example, the maximum rate of increase of water level could also be given by a variable:

$$MaxH_2Oin : DEPTH / TIME.$$

This variable is a real number with units of metres per second.

As our treatment of units is based on the familiar approach used in physics and engineering, we do not give all the details of our approach here. The interested reader is referred to [6, 12] for more details.

1.3 Continuous functions

To model time-varying quantities we take the simple approach of using a continuous function from $TIME$ to the type of the quantity. For example, the water level in the mine can be modelled by:

$$H_2O : TIME \twoheadrightarrow DEPTH.$$

We use the symbol '$\twoheadrightarrow$' to indicate a total continuous function. The water level at time *t* is given by $H_2O(t)$. In this case as the domain and range of the function are both real-valued, we have the normal continuous function of real analysis.

In mathematics, the notion of continuity has been generalised to *topological* continuity. For analogue functions, such as H_2O above, topological continuity corresponds to the usual real continuity, but one advantage of the topological generalisation is that we can discuss continuous functions with discrete ranges. For example, the water pump may be on or off. We can model the status of the pump by a continuous function from time to Boolean:

$$Pumping : TIME \twoheadrightarrow \mathbb{B}.$$

In this case the function is partial: it is not defined for all elements in its domain. In fact, the times at which it is not defined are the times at which the pump status is changing (or undefined).

The times when $Pumping$ is defined form into contiguous *intervals* of time, each of which comprises all the times between two instants of time, excluding the two endpoints. We call such intervals of time *open* intervals, since their endpoints are not contained within them. We write

$$(\!| \alpha \ldots \beta |\!) = \{\gamma : TIME \mid \alpha < \gamma < \beta\}$$

for the open interval with endpoints α and β and $\Im TIME$ for the collection of all open, time intervals. Sets of time, such as the domain of $Pumping$, which consist of a collection of open intervals are called *open* sets. The collection of all open sets of time forms a *topology* on the time domain, which we call $\mathcal{T}_{TIME}$.

The usefulness of topologies lies in the fact that continuity may be defined solely in terms of the topology of the domain and range of a function and that topologies may be defined on any set. For example, we give the booleans the discrete topology, in which every subset is an open set.

Continuous functions are useful because, for every open set in the range of a function, the set of times at which the value of the function lies in that open set is an open set of times; i.e. the inverse image of an open set is open. For example, $Pumping^{-1}(\!|\{\mathbf{true}\}|\!)$ is an open set. Recall that an open set of times comprises a collection of open intervals of time. Thus $Pumping$ is continuous in the sense that when it attains the value **true**

it *continues* to have the value **true** for some contiguous interval of time. In order to gain access to such time intervals we defined a function cov() which constructs the set of (maximal, disjoint) open intervals which comprise an open set of times. Thus in declaring a variable

$$\Delta : \mathrm{cov}(Pumping^{-1}(\!|\{\mathbf{true}\}|\!))$$

we may be sure that $Pumping$ is **true** over all of Δ, and that it is undefined at the endpoints of Δ. Δ forms the entirety of some interval of time during which $Pumping$ is **true**.

In using topologically, continuous functions we are able, for the most part, to consider the behaviour of any system at the level of intervals of time, rather than being forced to consider its behaviour in terms of individual instants of time. In this way the complexity involved in using the real numbers to model analogue quantities is held in check.

Timed history predicates When dealing with process state described as a function of time, it is convenient to introduce shorthand notations that allow commonly occurring forms of predicates to be expressed more succinctly. For example, to state that the water is below the danger level for the duration of *CriticalPeriod*, we can use the following timed history predicate:

$$H_2O < DangerH_2O \ \mathbf{on}\ CriticalPeriod.$$

It has the same meaning as the following:

$$\forall\, t : CriticalPeriod \bullet H_2O(t) < DangerH_2O.$$

1.4 The specification statement

To specify a system (or component process of a system) we need to define both the *assumptions* that the system makes of its environment and the *effect* it guarantees to achieve if placed in such an environment. As well, we enumerate the system variables that are constructed by the process. We use a *specification statement* to group these aspects of a specification together. It takes the form:

$$ConstructedVariables : [Assumption, Effect],$$

where *ConstructedVariables* is a list of variable declarations, and *Assumption* and *Effect* are (essentially) predicates.

The idea of a specification statement comes from [14], however, it has been adapted to the current context of timed histories rather than sequential programs.

For realistic specifications the predicates defining the assumptions and effects can become quite large. To help manage these predicates we make use of the Z schemas [19] to group together a list of variables with declared types and a predicate, and we use these schemas to define the assumption and effect predicates. This approach has the advantage that the assumption and effect schemas also make explicit the variables over which they are defined.

Z schemas can be used to group together variables defining the state of a process. For the mine monitoring system the state consists of the water level and the rates of flow of water into and out of the mine. These can be grouped together in a schema called *Water*:

$$
\begin{array}{|l}
\hline
\textit{Water} \\
H_2O : TIME \rightarrowtail DEPTH \\
H_2Oin, H_2Oout : TIME \rightarrowtail (DEPTH / TIME) \\
\hline
\forall\, t, t' : TIME \bullet t < t' \Rightarrow \\
\qquad H_2O(t') = H_2O(t) + \int_t^{t'} H_2Oin - H_2Oout \\
\hline
\end{array}
$$

The mine monitoring system assumes that the rate of flow of water into the mine is bounded by $MaxH_2Oin$. We can write a Z schema to capture this condition:

$$
\begin{array}{|l}
\hline
\textit{WaterRateLimited} \\
Water \\
\hline
H_2Oin < MaxH_2Oin \ \mathbf{on}\ TIME \\
\hline
\end{array}
$$

We have included the schema *Water* in the schema *WaterRateLimited* rather than rewriting the declarations and predicate of *Water*. The above definition of *WaterRateLimited* is equivalent to the following:

$$
\begin{array}{|l}
\hline
\textit{WaterRateLimited} \\
H_2O : TIME \rightarrowtail DEPTH \\
H_2Oin, H_2Oout : TIME \rightarrowtail (DEPTH / TIME) \\
\hline
\forall\, t, t' : TIME \bullet t < t' \Rightarrow \\
\qquad H_2O(t') = H_2O(t) + \int_t^{t'} H_2Oin - H_2Oout \\
H_2Oin < MaxH_2Oin \ \mathbf{on}\ TIME \\
\hline
\end{array}
$$

To specify that if the depth of water in the mine exceeds the danger level then the water alarm flag, H_2OFlag, must be raised, we can write:

$$
\begin{array}{|l}
\hline
\textit{WaterAlarm} \\
\textit{Water} \\
H_2OFlag : TIME \mathrel{+\!\!\!\rightarrow} \mathbb{B} \\
\hline
H_2O > DangerH_2O \Rightarrow H_2OFlag \ \mathbf{on}\ TIME \\
\hline
\end{array}
$$

These schemas define properties of systems or processes. To specify a process that has the *effect* of setting the flag if the water level exceeds the danger level, *assuming* that the rate of flow of water into the mine is below the given maximum rate, we can use the following specification statement:

$$WaterMonitor \mathrel{\widehat{=}} H_2OFlag : [WaterRateLimited, WaterAlarm].$$

In this case H_2OFlag is a variable completely constructed by the process.

2 Case study: specification

As an example of the use of our techniques for the specification of real-time processes, we consider the case of a mine-shaft pump system. This real-time case study has been treated by several authors [2, 3, 17, 18] and thus provides the opportunity to contrast our approach with that of others. In particular, we are able to attack the problem at a higher level of abstraction than is common, presenting a top-level specification rather than the design-oriented presentations in [2, 3, 17, 18].

The case study considered seeks to address the problem of water seepage in a mining shaft. The intention is to construct a process to monitor and control the water levels, so as to minimise the risk that the mine will need to be evacuated due to dangerously high water levels. Since activity within the mine is considered dangerous when high levels of methane gas are present, the process must also monitor the level of methane so that operations may be stopped during such danger periods.

Our purpose is to construct a formal description of the mine system and then use this system model to describe the process which will help guard against interruptions to mining operations caused by water seepage.

2.1 Monitoring the mine

Working conditions in the mine are monitored by various sensors which measure important safety parameters. We are primarily interested in those which affect the control of water levels in the mine, but others may exist (such as the air flow and carbon monoxide sensors discussed in [2]).

Water

Water tends to seep into the mine shafts from the surrounding water table. The depth of water in the mine is measured in metres.

$$DEPTH \mathrel{\widehat=} \mathbb{R}[Metre]$$

We model the water system in the mine as a sink into which and out of which water may flow.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Water}} \\
\hline
H_2O : TIME \mathrel{\ominus\!\!\!\rightarrow} DEPTH \\
H_2Oin, H_2Oout : TIME \mathrel{\ominus\!\!\!\rightarrow} (DEPTH \,/\, TIME) \\
\hline
\forall\, t, t' : TIME \bullet t < t' \Rightarrow \\
\qquad H_2O(t') = H_2O(t) + \int_t^{t'} H_2Oin - H_2Oout \\
\hline
\end{array}
$$

The mine is capable of continuing operation with a certain amount of water in the shaft, but if the water level exceeds

$$
\begin{array}{|l}
DangerH_2O : DEPTH,
\end{array}
$$

it will be necessary to take emergency action to ensure the safety of the workers in the mine. The monitor system is required to set a flag to indicate the existence of dangerous water levels. We represent this flag as a continuous, boolean history. Continuous histories with values in a discrete space, like the booleans, change value by becoming undefined for some period of time. In the case of flags we require that such state changes occur over a single instant of time.

$$Flag \mathrel{\widehat=} \left\{ b : TIME \mathrel{\oplus\!\!\!\rightarrow} \mathbb{B} \;\middle|\; \begin{array}{l} \forall\, t : TIME \bullet t \notin \operatorname{dom} b \Rightarrow \\ \qquad \exists\, \Delta : \Im\, TIME \bullet t \in \Delta \wedge \Delta \setminus \{t\} \subseteq \operatorname{dom} b \end{array} \right\}$$

The set $Flag$ is closed under the pointwise application of (finite numbers of) boolean operators, provided we handle the instants of undefinedness appropriately. Thus for $f, g : Flag$, if we define $\neg\, f$, $f \wedge g$, $f \vee g$, $f \Rightarrow g$, and $f \Leftrightarrow g$ by applying the operators pointwise, with the result being undefined whenever an operand is undefined, then each of these functions is also in $Flag$.

The water alarm flag must be set whenever the water is above the dangerous level.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{WaterAlarm}} \\
\hline
Water \\
H_2OFlag : Flag \\
\hline
H_2O > DangerH_2O \Rightarrow H_2OFlag \;\textbf{on}\; TIME \\
\hline
\end{array}
$$

Since any alarm system will take some time to react to the water level approaching its critical level, we require that there be some bound,

$$MaxH_2Oin : DEPTH / TIME,$$

on the rate at which the water level rises over all time.

$$\begin{array}{|l}
\hline
WaterRateLimited \\
Water \\
\hline
H_2Oin < MaxH_2Oin \ \mathbf{on} \ TIME \\
\hline
\end{array}$$

The water monitoring process is then required to raise the water danger flag whenever the water level is critical, provided that the rate at which the water level rises is sufficiently steady. We write this formally using a specification statement.

$$WaterMonitor \mathrel{\widehat{=}} H_2OFlag : [WaterRateLimited, WaterAlarm]$$

The water alarm flag is a construction of $WaterMonitor$, which must act to ensure that the effect $WaterAlarm$ is achieved, provided that the assumption $WaterRateLimited$ is satisfied.

Methane

Mining operations tend to cause the occasional release of trapped methane gas. High partial pressures of methane in the mine atmosphere can make the use of mining equipment hazardous so methane levels are also monitored. Pressures are measured in Pascals.

$$PRESSURE \mathrel{\widehat{=}} \mathbb{R}[Pascal]$$

As with the water level, we represent the methane level in the mine as a methane sink with inflows and outflows of methane.

$$\begin{array}{|l}
\hline
Methane \\
CH_4 : TIME \mathrel{\ominus\!\!\!\rightarrow} PRESSURE \\
CH_4in, CH_4out : TIME \mathrel{\ominus\!\!\!\rightarrow} (PRESSURE / TIME) \\
\hline
\forall\, t, t' : TIME \bullet t < t' \Rightarrow \\
\qquad CH_4(t') = CH_4(t) + \int_t^{t'} CH_4in - CH_4out \\
\hline
\end{array}$$

Atmospheric levels of methane above

$$DangerCH_4 : PRESSURE$$

are considered dangerous, so the monitor should raise a flag to warn of such levels.

$$
\begin{array}{|l}
\hline
\textit{MethaneAlarm} \\
\textit{Methane} \\
CH_4Flag : Flag \\
\hline
CH_4 > DangerCH_4 \Rightarrow CH_4Flag \ \textbf{on} \ TIME \\
\hline
\end{array}
$$

As in the case of the water monitor we require that methane buildups are bounded.

$$
\begin{array}{|l}
MaxCH_4in : PRESSURE/TIME
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{MethaneRateLimited} \\
\textit{Methane} \\
\hline
CH_4in < MaxCH_4in \ \textbf{on} \ TIME \\
\hline
\end{array}
$$

The formal specification of the methane monitor is then

$$MethaneMonitor \mathrel{\widehat{=}} CH_4Flag : [MethaneRateLimited, MethaneAlarm].$$

The final requirement of the mine monitoring system is that an alarm be raised when any of the mine parameters become safety critical.

$$
\begin{array}{|l}
\hline
\textit{RaiseAlarm} \\
Alarm, H_2OFlag, CH_4Flag : Flag \\
\hline
Alarm \Leftrightarrow H_2OFlag \vee CH_4Flag \ \textbf{on} \ TIME \\
\hline
\end{array}
$$

Note that the $RaiseAlarm$ schema may be modified to allow the introduction of other safety critical flags simply by factoring them into the above disjunction.

The action of the full monitoring system is then to detect safety critical levels of water or methane and when this occurs to sound an alarm.

The water and methane monitors act in parallel, which we write

$$Monitors \mathrel{\widehat{=}} WaterMonitor \parallel MethaneMonitor.$$

The parallel composition of two specifications forms a process which achieves the effects of both processes, provided the assumptions of both processes are satisfied. Note that extra monitoring functions may be added to the alarm system by simply redefining $Monitors$ so as to include them in parallel with the existing water and methane monitors.

The job of the alarm system is then to monitor the water and methane flags constructed by these monitors and to sound the alarm when they are set. Thus the alarm system operates in the environment created by the monitor systems. We express this using the piping operator, "$\_ \gg \_$".

$$MineMonitor \mathrel{\widehat{=}} Monitors \gg (Alarm : [\textbf{true}, RaiseAlarm])$$

2.2 Controlling water levels

In order to reduce the number of disruptions to the mine's operation due to high water levels it is considered desirable to implement a process for controlling the water level. The desired efficiency of the water level control process is that it should ensure that dangerous water levels occur less than once in every one thousand mining shifts. We represent the collection of all shifts as a time-ordered sequence of disjoint time-periods.

$$
\begin{array}{|l}
\hline
\textit{MineWork} \\
\textit{Shift} : \mathsf{seq}(\mathcal{T}_{TIME}) \\
\hline
\forall\, m, n : \mathsf{dom}\, \textit{Shift} \bullet \\
\qquad m < n \Rightarrow (\forall\, t : \textit{Shift}_m;\ t' : \textit{Shift}_n \bullet t < t') \\
\hline
\end{array}
$$

We formalise the efficiency requirement as: in any thousand consecutive shifts, at most one will be disrupted by dangerous water levels.

$$
\begin{array}{|l}
\hline
\textit{ControlWater} \\
\textit{WaterAlarm} \\
\textit{MineWork} \\
\hline
\forall\, m, n : \mathsf{dom}\, \textit{Shift} \bullet \#(m..n) \leq 1000 \Rightarrow \\
\qquad \#\{k : m..n \mid H_2O > \textit{Danger}H_2O \ \mathbf{in}\ \textit{Shift}_k\} \leq 1 \\
\hline
\end{array}
$$

We are presented with the question of under what conditions would it be feasible to achieve this effect.

As any equipment used to achieve this effect will take time to react to rising water levels, we must require that there be a bound on the rate at which the water flows into the mine. This is the assumption $\textit{WaterRateLimited}$ introduced above.

Further, we know that the presence of high levels of methane makes the use of equipment in the mine dangerous. It is reasonable to assume that long or frequent periods of dangerous methane levels will make the goal of controlling water levels difficult. Let

$$\textit{TooMuchCH}_4 : \textit{TIME}$$

represent the maximum duration of dangerous methane levels over a single shift and let

$$\textit{TooManyCH}_4 : \mathbb{N}$$

represent the maximum number of occurrences of periods of dangerous methane levels over a single shift, which would be compatible with the effective operation of the water level control process. That is, it is not possible to guarantee effective control of the water

level when these bounds are exceeded during a shift. We require that such sustained high levels of methane occur no more than once in every one thousand shifts.

$$
\begin{array}{|l}
\hline
\textit{SafeMethane} \\
\textit{MethaneAlarm} \\
\textit{MineWork} \\
\hline
\textbf{let}\ \textit{HighMethane} \mathrel{\widehat{=}} CH_4^{-1}(\!|\{\!-\textit{DangerCH}_4 \ldots \infty\}\!-\!|\!) \\
\quad \textit{BadMethane} \mathrel{\widehat{=}} \lambda\, P : \mathcal{T}_{TIME} \bullet \\
\qquad\qquad\qquad \|\textit{HighMethane} \cap P\| > \textit{TooMuchCH}_4 \vee \\
\qquad\qquad\qquad \#\mathrm{cov}(\textit{HighMethane} \cap P) > \textit{TooManyCH}_4 \\
\forall\, m, n : \mathrm{dom}\, \textit{Shift} \bullet \#(m..n) \leq 1000 \Rightarrow \\
\qquad \#\{k : m..n \mid \textit{BadMethane}(\textit{Shift}_k)\} \leq 1 \\
\hline
\end{array}
$$

The notation $\|\_\|$ measures the length of a period of time.

Our final requirement is more subtle. The water level control system will most likely act by removing water from the mine shaft. We must therefore require that there is no impediment to the removal of water from the mine shaft. This condition corresponds to the system observable H_2Oout being free to assume any value from its type domain.

$$
\begin{array}{|l}
\hline
\textit{WaterRemovable} \\
\textit{Water} \\
\hline
\textit{free}(H_2Oout, \textit{TIME} \mathrel{\ominus\!\!\!\rightarrow} (\textit{DEPTH} / \textit{TIME})) \\
\hline
\end{array}
$$

The predicate $\textit{free}$ is a modal logic operator which states that it is possible for its first argument to attain any value in the set determined by the second argument. Using the modal logic operator $\Diamond$ (read *possibly* [8]) the above condition may be expressed by

$$\forall\, f : \textit{TIME} \mathrel{\ominus\!\!\!\rightarrow} (\textit{DEPTH} / \textit{TIME}) \bullet (\Diamond\, H_2Oout = f).$$

The action of the water level control process is then specified by

$$\textit{WaterSystem} \mathrel{\widehat{=}} \left[\begin{array}{c} \textit{WaterRateLimited} \wedge \\ \textit{SafeMethane} \wedge \\ \textit{WaterRemovable} \end{array} , \textit{ControlWater} \right]$$

3 Refinement

Refinement calculi have been developed for sequential systems by Back [1], Morgan [14] and Morris [15]. These calculi all make use of a programming notation that allows program code and (sequential) specification statements to be intermixed in programs

during their development. The calculi make use of a well-defined notion of refinement between programs and, in particular, between an abstract specification and another, more concrete, specification.

To develop an implementation that satisfies the specification of the mine control system we make use of a refinement calculus for time-based processes [10, 11]. The basic philosophy behind the timed calculus is the same as for the sequential refinement calculi, however, because the basic model of the system is different, the timed refinement calculus differs.

As we are only concerned with the top-level design, we do not get down to the level of detailed program code, rather we produce a design consisting of a collection of processes which, in combination, satisfy the top-level specification. The design puts together the processes using operators such as piping and parallel combination, and gives a precise specification of each process in the form of a (timed) specification statement.

As examples of (timed) refinement laws we give two basic laws for refining one specification statement by another.

Weaken assumption A specification statement $z : [A1, E]$ is refined by $z : [A2, E]$, written

$$z : [A1, E] \sqsubseteq z : [A2, E],$$

provided

$$A1 \Rrightarrow A2.$$

Strengthen effect A specification statement $z : [A, E1]$ is refined by $z : [A, E2]$, provided

$$A \Rrightarrow (\forall z \bullet E2 \Rightarrow E1)$$

The timed refinement calculus provides laws to allow refinement to various combinations of processes including piping, parallelism, etc. We introduce these laws below, where they are required, and summarise them in Appendix B.

3.1 A water pump

We propose to implement the water level control process with the use of an electric pump to remove water from the mine shaft. The process will continuously monitor the water level in the mine shaft, so that it may commence pumping water from the mine before the water level becomes dangerous. Since methane is an explosive gas, the pump must not be operated in its presence.

The minimum ability of the pump to remove water from the mine shaft is represented by

$$\begin{array}{|l} PumpRating : DEPTH / TIME \\ \hline MaxH_2Oin < PumpRating. \end{array}$$

We model the pump by the times at which the pump is operating.

$$\begin{array}{|l} \hline Pump \\ Pumping : Flag \\ \hline \end{array}$$

When the pump is installed in the mine and pumping, it should remove water at a rate at least equal to its rating, provided there is any water present.

$$\begin{array}{|l} \hline PumpAction \\ Water \\ Pump \\ \hline Pumping \wedge H_2O > 0 \Rightarrow H_2Oout \geq PumpRating \ \mathbf{on} \ TIME \\ \hline \end{array}$$

Clearly the pump can be installed only if there is no impediment to water removal.

$$InstalledPump \mathrel{\widehat{=}} [WaterRemovable, PumpAction]$$

Once installed the pump must be controlled in such a way as to ensure the goals of $WaterSystem$. In order to ensure that water levels do not rise above $DangerH_2O$ we set a slightly lower water level,

$$\begin{array}{|l} HighH_2O : DEPTH \\ \hline HighH_2O < DangerH_2O, \end{array}$$

which the pump control system will react to by turning on the pump. Once the water level reaches $HighH_2O$ the minimum time that the water can conceivably reach $DangerH_2O$ is

$$\frac{DangerH_2O - HighH_2O}{MaxH_2OIn}.$$

We assume that the reaction time of the control system,

$$\begin{array}{|l} React : TIME \\ \hline React < \dfrac{DangerH_2O - HighH_2O}{MaxH_2OIn}, \end{array}$$

is fast enough to ensure that the pump is turned on before the water level becomes dangerous.

Methane

Remember that we have assumed that high levels of methane make the use of machinery in the mine hazardous. It may not be safe to have the pump running during periods of high methane levels, so the controller should ensure that this does not occur. This may lead to lost pumping time of up to $TooMuchCH_4$, as well as delays in turning the pump back on after each shutdown. Thus we must make even stronger assumptions about the reaction time. We require that the time lost due to methane interuptions is not sufficient to allow the water level to rise to dangerous levels

$$LostTime : TIME$$

$$LostTime = (TooManyCH_4 + 1) * React + TooMuchCH_4$$

$$LostTime < \frac{DangerH_2O - HighH_2O}{MaxH_2OIn} \qquad \text{EQ 3.1}$$

and that it be short enough that the pump can be turned off before any of the hazards of high methane levels are manifested.

Controlling the pump

The control mechanism for the pump turns the pump on when the water level gets high, except when the methane level is too high.

$PumpControl$

$Pump$
$Water$
$Methane$

$$\begin{aligned}
&\textbf{let}\ HighWater \mathrel{\widehat{=}} H_2O^{-1}(\!|\{HighH_2O \mathinner{..} \infty\}|\!) \\
&\quad HighMethane \mathrel{\widehat{=}} CH_4^{-1}(\!|\{DangerCH_4 \mathinner{..} \infty\}|\!) \\
&\quad LowMethane \mathrel{\widehat{=}} CH_4^{-1}(\!|\{-\infty \mathinner{..} DangerCH_4\}|\!) \\
&\forall \Delta : \mathrm{cov}(HighWater \cap LowMethane) \bullet \\
&\qquad Pumping\ \textbf{on}\ \{\inf \Delta + React \mathinner{..} \sup \Delta\} \\
&\forall \Delta : \mathrm{cov}(HighMethane) \bullet \\
&\qquad (\neg\ Pumping)\ \textbf{on}\ \{\inf \Delta + React \mathinner{..} \sup \Delta\}
\end{aligned}$$

The full specification of the pump system is then

$$\begin{aligned}
&PumpSystem \mathrel{\widehat{=}} \\
&\quad [\![(Pumping : [\textbf{true}, PumpControl]) \gg InstalledPump \setminus Pumping]\!].
\end{aligned}$$

The operator $[\![\_ \setminus Pumping]\!]$ hides the construction $Pumping$ from interaction with other processes, ensuring that only $PumpSystem$ has control over it.

Theorem 3.1 *The pump system forms an adequate implementation of the desired water control system.*

$$\mathit{WaterSystem} \sqsubseteq \mathit{PumpSystem}$$

Proof: We proceed by applying several small refinement transformations.

Any specification may be refined by adding a new constructed variable that is local to the specification.

$$WaterSystem \sqsubseteq |[\, Pumping : WaterSystem \setminus Pumping \,]| \qquad \text{[RL6]}$$

The piping operator may be introduced by splitting a specification statement at an appropriate intermediate condition. We decompose our specification into a piped composition by introducing the intermediate condition

$$WaterRateLimited \wedge SafeMethane \wedge WaterRemovable \wedge PumpControl.$$

For the sake of brevity we give a name to the assumptions made by the $WaterSystem$ process, $WSass \mathrel{\widehat{=}} WaterRateLimited \wedge SafeMethane \wedge WaterRemovable$.

$$\begin{aligned} & |[\, Pumping : WaterSystem \setminus Pumping \,]| \\ & \sqsubseteq |[\, (Pumping : [WSass, WSass \wedge PumpControl]) \\ & \qquad \gg [WSass \wedge PumpControl, ControlWater] \setminus Pumping \,]| \end{aligned} \qquad \text{[RL4]}$$

Timed specification statements preserve assumed invariants of a system.

$$\begin{aligned} & (Pumping : [WSass, WSass \wedge PumpControl]) \\ & \sqsubseteq (Pumping : [WSass, PumpControl]) \end{aligned} \qquad \text{[RL2]}$$

A specification statement may be refined by relaxing the assumptions it makes.

$$\begin{aligned} & (Pumping : [WSass, PumpControl]) \\ & \sqsubseteq (Pumping : [\mathbf{true}, PumpControl]) \end{aligned} \qquad \text{[RL1]}$$

Any specification statement may by refined by strengthening its effect in the context of its assumptions. Thus in order to deduce that

$$\begin{aligned} & [WSass \wedge PumpControl, ControlWater] \\ & \sqsubseteq [WSass \wedge PumpControl, PumpAction], \end{aligned} \qquad \text{[RL3]}$$

we need only show that for any system which satisfies

$$WaterRateLimited \wedge SafeMethane \wedge WaterRemovable \wedge PumpControl,$$

$PumpAction$ implies $ControlWater$. This follows from the assumptions we have made about the rate at which water may flow into the mine and about the duration and frequency of dangerous levels of methane. We postpone the formal details of this argument to the proof of Lemma 3.2.

A specification may be refined by relaxing the assumptions it makes.

$$[WSass \wedge PumpControl, PumpAction] \qquad \text{[RL3]}$$

$$\sqsubseteq [WaterRemovable, PumpAction] \qquad \text{[RL1]}$$

Finally since all of the specification operators are compositional with respect to refinement we may deduce that

$$\begin{aligned}
&|[\, Pumping : WaterSystem \setminus Pumping \,]| \\
&\sqsubseteq |[\, (Pumping : [\mathbf{true}, PumpControl]) \\
&\qquad \gg [WaterRemovable, ControlWater] \setminus Pumping \,]| \\
&= PumpSystem.
\end{aligned}$$

□

Lemma 3.2

$$WSass \wedge PumpControl \Rrightarrow (PumpAction \Rightarrow ControlWater)$$

Proof: Assume $WaterRateLimited$, $SafeMethane$, $PumpControl$, and $PumpAction$. Choose $m, n \in \mathrm{dom}\, Shift$ such that $\#(m..n) \leq 1000$ and $k \in m..n$.

Suppose that during $Shift_k$ the methane levels remain within acceptable operating parameters; that is, the intervals during which methane levels are dangerous are less than $TooMuchCH_4$ in duration and fewer than $TooManyCH_4$ in number and that Δ is an interval of time during which the water level is above $HighH_2O$ (that is $\Delta \in \mathrm{cov}(H_2O^{-1}(\!|\, HighH_2O \ldots \infty \,|\!))$).

From $PumpControl$ we know that the longest time during Δ for which the pump may not be operating is $LostTime$ (the maximum number of times the pump might have to be switched on times the reaction time, plus the maximum time it must be left off due to methane presence). Thus from $Water$ we know that for any $t : TIME$ in Δ,

$$H_2O(t) = H_2O(\inf \Delta) + \int_{\inf \Delta}^{t} H_2Oin - H_2Oout.$$

Since H_2O is continuous and total, $H_2O(\inf \Delta) = HighH_2O$. Since

$$PumpRating > MaxH_2Oin,$$

$WaterRateLimited$ and $PumpAction$ tell us that the water level is falling when the pump is on. The maximum depth of the water level over Δ is thus bounded by its initial depth ($HighH_2O$) and the amount it can rise whilst the pump is not operating; that is

$$H_2O < HighH_2O + LostTime * MaxH_2Oin \ \mathbf{on}\ \Delta \qquad [WaterRateLimited]$$

$$\Rrightarrow H_2O < DangerH_2O \ \mathbf{on}\ \Delta. \qquad \text{[By EQ 3.1]}$$

From this we may deduce that whenever the methane levels remain within safe tolerances over a shift the water level never rises above $DangerH_2O$.

Since, in $SafeMethane$, we have assumed that methane levels rise above safe tolerances only once in every one thousand shifts, we may deduce that water levels become dangerous no more frequently than this and $ControlWater$ is thus satisfied. □

4 Conclusions

A primary requirement for the specification of real-time systems is the ability to specify the timing aspects of such systems. In addition, real-time control systems typically have both discrete and analogue inputs and outputs. Hence any whole-system specification must be able to address both discrete and analogue properties of the system.

The approach taken in this paper is to make use of topological continuous functions of time, as they both capture the timing aspects of the behaviour of a system and allow discrete and analogue signals to be discussed within a single framework. This approach, while perhaps novel in a computing science context, has strong precedents within physics and engineering. An additional useful notational device, also adapted from the phyical sciences, is the use of (physical) units to aid in the understanding and type compatibility checking of specifications.

The use of continuous observables, as opposed to the simple adoption of analogue time models, helps direct the specifier away from detailed considerations of the method by which a system must evolve through time. Instead, the specifier is encouraged to consider the desired properties of a system throughout all time or during critical periods of time, with minimal attention to the activities required to achieve these properties. That is, to state what must be true of the system, rather than what the system must do. This may be contrasted with recent work based on timed transition systems and automata, such as Real-Time Logic [9], (Really) Temporal Logic [7], Timed CSP [16] and CCS [20], in which preoccupation with the actions which must occur inevitably leads to an algorithmic and evolutionary description of a system not always appropriate to the highest levels of the development process.

In a similar vein to the specification vocabulary presented in this paper is the Calculus of Durations [4] which also aims to describe the observable properties of real-time systems in terms of an interval based calculus of real-valued functions of time. The notation of the calculus of durations offers a higher level of abstraction, at the expense of some expressive power, but would benefit from using continuous functions as we do.

The sequential refinement calculus has been a significant breakthrough in providing a coherent framework for the systematic development of programs from specifications to code. The same overall philosophy has been adopted for the development of systems of processes that combine together to satisfy a specification. Central to the refine-

ment calculus is the concept of a specification statement that describes the allowable behaviours of a system (or component process), and an important aspect of the specification statement is a clear separation between the assumptions that the system makes about its environment and the effect that it guarantees to achieve provided that the environment satisfies the assumptions.

The objective of the present paper has been to assess these techniques on a small but realistic case study of a mine control system. By making use of timed history predicates we have been able to develop a detailed specification of the desired behaviours of the mine control system. In addition, we have been forced to formalise a number of concepts that were not clear in the informal specification on which our work is based [2]. Our specification is also at a higher level of abstraction: it makes fewer assumptions about the design of an implementation, and concentrates on specifying those properties of a mine control system that all implementations must satisfy.

The development of a top-level design has been performed and a proof of correctness of the refinement presented. The refinement technique allows a system to be decomposed into components that may be implemented in hardware or software, or a combination of both, and allows for further decomposition of components into subcomponents, etc. The end product is a system of processes combined together with piping and parallel operators, and making use of hidden variables.

The application of the timed refinement calculus techniques to the refinement of a mine control system has been successful at high-lighting the top-level design decisions, and verifying their validity before further development of the component processes is undertaken.

Acknowledgements

We would like to thank Andrew Lister for the motivation to apply these techniques to the mine pump example and his feedback on an early draft of this paper. The work reported in this paper has been supported by the Software Verification Research Centre, University of Queensland, and by Australian Research Council grant number A4913006: *Formal methods for the specification and refinement of time-based systems and processes.*

References

[1] R. J. R. Back. A calculus of program derivations. *Acta Informatica*, 25:593–624, 1988.

[2] A. Burns and A. M. Lister. A framework for building dependable systems. *The Computer Journal*, 34(2):173–182, April 1991.

[3] A. Burns and A. Wellings. *Real-time Systems and their Programming Languages*, chapter 16: A Case Study in Ada, pages 497–528. Addison-Wesley, 1990.

[4] Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. Technical Report ProCoS Rep. OU ZCC2, Programming Research Group, Oxford University, 1991.

[5] I. J. Hayes, editor. *Specification Case Studies*. Prentice Hall International, 1987.

[6] I. J. Hayes. A generalisation of bags in Z. In J. E. Nicholls, editor, *Proceedings of the Z User Meeting*, Workshops in Computing, pages 113–127. Springer-Verlag, December 1989.

[7] T. A. Henzinger, Z. Manna, and A. Pnueli. Temporal proof methodologies for real-time systems. In *Proceedings of the* 18th *ACM Symposium on the Principles of Programming Languages*, pages 353–363, 1991.

[8] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen and Co Ltd, 1968.

[9] F. Jahanian and A. K. Mok. Safety analyis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, 12(9):890–904, 1986.

[10] B. P. Mahony. *The Specification and Refinement of Timed Processes*. PhD thesis, University of Queensland, 1992.

[11] B. P. Mahony and I. J. Hayes. A case-study in timed refinement: A central heater. In J. M. Morris and R. C. Shaw, editors, *Proceedings of the* 4th *Refinement Workshop*, Workshops in Computing, pages 138–149. Springer-Verlag, 1991.

[12] B. P. Mahony and I. J. Hayes. Using continuous real functions to model timed histories. In *Proceedings of the 6th Australian Software Engineering Conference (ASWEC91)*, 1991.

[13] C. C. Morgan, K. A. Robinson, and P. Gardiner. On the refinement calculus. Technical Monograph PRG-70, Oxford University Programming Research Laboratory, 1988.

[14] C.C. Morgan. The specification statement. *ACM Trans. Prog. Lang. and Sys.*, 10(3), July 1988. Reprinted in [13, pp. 7–30].

[15] J. M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer Programming*, 9:287–306, 1987.

[16] G. M. Reed. *A Uniform Theory for Real-Time Distributed Computing*. PhD thesis, St. Edmund Hall, Oxford University, Hilary 1988.

[17] S. K. Shrivastava, L. V. Mancini, and B. Randell. On the duality of fault tolerant system structures. In *Experiences with Distributed Systems*, volume 309 of *Lecture Notes in Computer Science*, pages 19–37. Springer-Verlag, 1987.

[18] M. Sloman and J. Kramer. *Distributed Systems and Computer Networks*. Prentice-Hall International Series in Computer Science. Prentice-Hall Intenational, 1987.

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

[20] C. Stirling. An introduction to modal and temporal logics for CCS. In A. Yonezawa and T. Ito, editors, *Concurrency: Theory, Language, and Architecture*, LNCS 491, pages 2–20. Springer-Verlag, 1991.

A Glossary

$A \nrightarrow B$ – The continuous, total functions from A to B.

$A \nrightarrow B$ – The continuous, partial functions from A to B.

$(\alpha \ldots \beta)$ – The open set of times between α and β, exclusive.

$\Im$*TIME* – The collection of open intervals of time.

$\mathcal{T}_{TIME}$ – Periods of time which are comprised of a collection of disjoint, open intervals. The time topology.

cov(*PERIOD*) – The collection of disjoint, open intervals which comprise the open set of time $PERIOD$.

PRED **on** *PERIOD* – The statement that $PRED$ is true for all times in $PERIOD$.

PRED **in** *PERIOD* – The statement that $PRED$ is true for some times in $PERIOD$.

PRED **at** *t* – The statement that $PRED$ is true at the time t.

$\Diamond$ *PRED* – The statement that a system must be able to exhibit the behaviour described by $PRED$.

$\vec{z} : [A, E]$ – The process which adds the observables $\vec{z}$ to its environment and causes the composite system to satisfy E, provided that its environment satisfies A.

$S1 \gg S2$ – The process $S1$ acts to create the environment in which the process $S2$ acts.

$S1 \parallel S2$ – The processes $S1$ and $S2$ acting in parallel.

$|[\, S \setminus \vec{w} \,]|$ – The constructions $\vec{w}$ are local; protected from external influences.

B Refinement laws

RL1 Weaken assumption.

$$\frac{\vec{z} : [A_1, E]}{\vec{z} : [A_2, E]} \quad [\ A_1 \Rrightarrow A_2\ ]$$

RL2 Drop invariants.

$$\frac{\vec{z} : [A, E \wedge A]}{\vec{z} : [A, E]}$$

RL3 Strengthen effect.

$$\frac{\vec{z} : [A, E_1]}{\vec{z} : [A, E_2]} \quad [\ A \Rrightarrow \forall\, \vec{z} \bullet E_2 \Rightarrow E_1\ ]$$

RL4 Introduce piping.

$$\frac{\vec{z} : [A, E]}{(\vec{z}_1 : [A, Mid]) \gg (\vec{z}_2 : [Mid, E])} \quad [\ \vec{z}_1 \cap \vec{z}_2 = \{\ \}, \vec{z}_1 \cup \vec{z}_2 = \vec{z}\ ]$$

RL5 Introduce parallelism.

$$\frac{\vec{z} : [A_1 \wedge A_2, E_1 \wedge E_2]}{(\vec{z}_1 : [A_1, E_1]) \parallel (\vec{z}_2 : [A_2, E_2])} \quad [\ \vec{z}_1 \cap \vec{z}_2 = \{\ \}, \vec{z}_1 \cup \vec{z}_2 = \vec{z}\ ]$$

RL6 Introduce hiding.

$$\frac{\vec{z} : [A, E]}{|[\ \vec{z};\ \vec{w} : [A, E] \setminus \vec{w}\ ]|} \quad [\ \vec{w} \text{ new variables }]$$