A Formal Approach to Computer Systems Requirements Documentation

Marcin Engel, Marcin Kubica, Jan Madey[1], David Lorge Parnas[2], Anders P. Ravn[3], A. John van Schouwen[4]

[1] Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
E-mail: madey@mimuw.edu.pl
[2] Communications Research Laboratory
Department of Electrical and Computer Engineering
McMaster University
Hamilton, Ontario, Canada L8S 4L7
E-mail: parnas@triose.eng.mcmaster.ca
[3] Department of Computer Science
Technical University of Denmark, Bldg. 344
DK 2800 Lyngby, Denmark
E-mail: apr@id.dth.dk
[4] Bell-Northern Research Limited
P.O. Box 3511 Station C
Ottawa, Ontario, Canada K1Y 4H7 E-mail: schouwen@bnr.ca

Abstract. This paper demonstrates how the extended duration calculus [4] can be used to support the approach to documentation of computer systems presented by in [1]. This approach uses the general concept of mathematical relations to specify properties, while the calculus of durations provides the means to reason about such specifications, and in particular, prove formally that a design implies the requirements. The presentation is based on an example originally presented in [2], and later reformulated in [3] following the approach described on [1]. In the present paper we introduce all needed relations, express them in terms of duration calculus, and formally verify software design acceptability.

1 Introduction

Software design is carried out using methods quite unlike those used by engineers in the other disciplines. It often proceeds by building first and documenting afterwards. When software documentation is produced prior to implementation, it is typically inaccurate and imprecise. The documentation uses anthropomorphic analogies and intuitive language in a way that would be considered unprofessional in other disciplines. Although the long-term benefits of precise documentation are great, developers are reluctant to do better because good documentation requires substantial effort. Documenting a software system is seen as a diversion because the end-product is not being realized.

Precise documentation has many potential benefits:

1. designers know what they are building before proceeding with further design work;
2. reviewers can check that the specifiers' intentions meet their expectations;
3. those charged with quality assurance can verify an implementation against a statement of required behaviour and can formulate validation tests without looking at source code; and,
4. maintainers can become acquainted with the system more easily, and can maintain the original design intentions when making changes.

The logical starting point for any system design should be a description of the environment in which the system is to be used and the system's required behaviour within that environment. This information is recorded in what we call a *system requirements document.* The system is modelled as a state machine that monitors and controls certain aspects of its environment, i.e. as a dynamic system. The state of the system is a total function of time. Those aspects determine the *environmental quantities of interest.* There may be physical constraints, such as feedback relations and maximum derivatives, on the environmental quantities. The computer system will be required to further restrict these relations.

This paper focuses primarily on software systems. Usually before software design begins, certain decisions will have been made about the computer(s) and peripheral devices that will be used. Specifications for these should be documented so that the software designers can refer to details about them readily. These decisions are recorded in what we call the *system design document.* It describes the interfaces to the I/O devices by presenting the relations between the I/O registers and the environmental quantities. (New environmental quantities of interest might need to Be added to represent abstract device states for some of the devices or classes thereof.) Since the software designers will need to refer to the contents of both the system requirements and system design documents, both of them can be combined into a single one called the *software requirements document.* The software designer's job is to implement a mapping from computer inputs to computer outputs such that the required relation between the environmental quantities of interest is satisfied.

In this paper, we deal with a *functional* approach[5] to documenting the systems that we wish to build as well as the devices that will exist within the system. This approach is explained in Section 3 and illustrated on examples; a more detailed discussion of it can be found in [1]. In Section 2 we introduce briefly the Water Level Monitoring System from which the examples are derived. A complete specification and discussion of this system is presented in [2], while its reformulated, shorter version is to be found in [3]. In Section 4 we first informally introduce the extended calculus of durations [4,5], and then use it to describe the requirements and verify the software's acceptability. Conclusions are presented in Section 5. The Appendix provides a more detailed description

[5] More precisely, we use *binary relations* rather than functions. Since, however, a binary relation $R \subseteq A \times B$ can be treated as a function $R : A \rightarrow \mathcal{P}(B)$, where $\mathcal{P}(B)$ denotes the set of all subsets of B, the term "functional approach" is well justified.

of applying the extended calculus of durations to the main example [6].

2 The Water Level Monitoring System

The Water Level Monitoring System (WLMS) is a steam generator subsystem that monitors the water level in the steam generation vessel. The system controls, among other things: a display that informs the operator of the water level; a light that notifies the operator of the fact that the water level has exceeded an upper bound; and, a switch that, when open, prevents additional water from entering the vessel.

In this paper by "WLMS" we will understand an implementation of such a system, which is described in [2]. Among the monitored environmental quantities for the WLMS is *WaterLevel*, which denotes the level of water (measured in centimetres) in the horizontally-oriented vessel measured along the vertical axis on the left side (when looking from its front to its back), 5.0 cm from the front edge. Among the controlled environmental quantities are some display windows on a CRT screen, e.g. *LevelDisplay* (denoting the current level of water) and *HighWindow* (representing a visible alarm, a light).

Typically, an independent system would control the steam generation process. It would control The rate at which water is introduced into the system and the rate at which water is boiled off. The monitoring system's purpose is twofold: it must convey information about the system's state to an operator, and it must take certain safety actions in the face of control system failure.

Further discussion of WLMS is relegated to the examples.

3 A functional approach to system documentation

In this section we will discuss the mathematical relations that should be described in the three documents mentioned in the Introduction (i.e. the system requirements, system design, and software requirements documents). Let us first explain some important terms which will be used in the sequel.

Definition 1. Let $R \subseteq X \times Y$ be a (binary) relation, then:

– The set of pairs R could also be defined by its *characteristic predicate*, $R(x, y)$, i.e.:

$$R = \{(x, y) \in X \times Y \mid R(x, y)\}.$$

– The *domain* and the *range* of R can be expressed as follows:

$$\mathrm{domain}(R) = \{x \in X \mid R(x, y) \text{ for some } y \in Y\},$$

$$\mathrm{range}(R) = \{y \in Y \mid R(x, y) \text{ for some } x \in X\}.$$

[6] D. L. Parnas participated only in the first three sections of the paper.

– The *(relational) image* of an element x from the domain of $R, R(x)$, is the following subset of the range of R:

$$R(x) = \{y \in Y \mid R(x, y)\}.$$

– If for every x from the domain of R, $R(x)$ has exactly one element — R is thus a *function* — this element is called the *value* of R at x, and we can omit the set braces; in other words, $R(x)$ will denote the value of the function R at x, rather than a singleton set.
– The notion of image can also be extended to any subset A of X, as follows:

$$R(A) = \{y \in Y \mid R(a, y) \text{ for some } a \in A\},$$

hence:
  - $\text{range}(R) = R(X) = R(\text{domain}(R))$,
  - $(x \notin \text{domain}(R)) \Rightarrow (R(x) = \emptyset)$, i.e. the image is empty if R is not defined at x.

3.1 The system requirements document

A critical step in documenting the requirements of a computer system treated as a "black box" is the definition of the environmental quantities of interest (those to be measured and/or controlled). They include:

– physical properties (such as temperatures),
– the readings on user-visible displays,
– administrative information (such as the number of people assigned to a given task), and even
– the wishes of a human user.

The values of these environmental quantities change with time. Hence, each of them can be expressed as a *time function* (i.e. function of time) in the way that is usual in engineering (i.e. by modelling time as a set of real numbers). The association between the physical quantities of interest and their mathematical representations must be carefully defined. (One should remember that the environmental quantities of interest can include various aspects of hardware devices — typically introduced at the system design stage.) For the rest of this paper we will assume that the system S under consideration is known and so, for simplicity, we will omit the subscript S in some of the denotations related to this system.

Definition 2.

– The set of environmental quantities is divided into two subsets:
  - *controlled* quantities, whose values the system is intended to restrict, and
  - *monitored* quantities, those that are to affect the controlled ones.
– Let p be the number of quantities to be controlled in S, and F_C a selected set of time-functions[7]. Then:

[7] In this and following definitions each selected set of time-functions is determined by physical characteristic of the system S.

  - controlled quantities will be represented by a p-tuple called a *controlled state function*, $\boldsymbol{c} = (c_1, \dots, c_p)$, where each c_i is a member of F_C and corresponds to the i-th controlled quantity,
  - the set of controlled state functions will be denoted by C_S.
- Let q be the number of quantities to be monitored in S, and F_M a selected set of time-functions. Then:
  - monitored quantities will be represented by a q-tuple called a *monitored state function*, $\boldsymbol{m} = (m_1, \dots, m_q)$, where each m_i is a member of F_M and corresponds to the i-th monitored quantity,
  - the set of monitored state functions will be denoted by M_S.

Note, that if the same quantity is to be both monitored and controlled, the corresponding time-functions must be specified to be identical (by relation NAT, see below).

Example 1. For the WLMS we have:

1. Among the monitored quantities: the level of water measured as described in Section 2 and represented by the function *WaterLevel*.
2. Among the controlled quantities:
   (a) a visible annunciation signal whose window is labelled "WATER LEVEL HIGH". This aspect is represented by the function *HighWindow*, which is described as follows:
   $$\mathit{HighWindow}(t) = \begin{cases} \mathit{on}, \text{ if "light" is on at time } t \\ \mathit{off}, \text{ if "light" is off at time } t \end{cases}$$
   (b) the state of a switch. The function used to represent this is named *Pump-Switch* and is described as follows:
   $$\mathit{PumpSwitch}(t) = \begin{cases} \mathit{closed}, \text{ if switch contacts are closed at time } t \\ \mathit{open}, \quad \text{if switch contacts are open at time } t \end{cases}$$
   (c) the value presented in a display labelled "WATER LEVEL". The real-valued function used to represent this is named *LevelDisplay*.

Relation NAT. Nature and previously installed systems place constraints on the values of environmental quantities. These restrictions may be documented by means of a relation, which we call NAT.

Definition 3.

- NAT $\subseteq M_S \times C_S$,
- NAT$(\boldsymbol{m}, \boldsymbol{c})$ if and only if the environmental constraints allow the controlled quantities to take on the values described by $\boldsymbol{c}$ when the values of the monitored quantities are described by $\boldsymbol{m}$.

One should note the following:

1. domain(NAT) (respectively, range(NAT)) contains *exactly* those instances of $\boldsymbol{m}$ (respectively, $\boldsymbol{c}$) that are allowed by the environmental constraints.
2. If there are no constraints on $\boldsymbol{m}$ and $\boldsymbol{c}$, then NAT $= M_S \times C_S$.

3. If any values of $\boldsymbol{m}$ are not included in the domain of NAT, the system designer may assume that these values do not occur.
4. NAT is rarely a function; if it were the computer system would not be able to vary the values of the controlled quantities without affecting changes in the monitored quantities (the values of the controlled quantities would be uniquely determined by the values of the monitored ones).

Example 2. A subset of the constraints on the domain of NAT for the WLMS is:

1. $\mid \frac{d\,WaterLevel(t)}{dt} \mid \leq 0.375$ cm/s; and
2. 0.0 cm $\leq WaterLevel(t) \leq$ 30.0 cm.

Thus, there are no values of $\boldsymbol{m}$ in the domain of NAT when the water level rises or falls more rapidly than as stated in (1), nor when the water level exceeds 0.0 cm or 30.0 cm, as stated in (2).

Relation REQ. The computer system is required to impose further constraints on the environmental quantities. The permitted values may be documented by means of a relation, which we call REQ.

Definition 4.

- REQ $\subseteq M_S \times C_S$,
- REQ$(\boldsymbol{m}, \boldsymbol{c})$ if and only if the computer system may permit the controlled quantities to take on the values described by $\boldsymbol{c}$ when the values of the monitored quantities are described by $\boldsymbol{m}$.

One should note the following:

1. domain(REQ) contains those instances of $\boldsymbol{m}$ which are allowed by the environmental constraints.
2. range(REQ) contains only those instances of $\boldsymbol{c}$ that are considered permissible.
3. REQ is usually not a function because the application can tolerate some errors in the values of controlled quantities.

Example 3. We can illustrate REQ as follows:

In the WLMS, the value conveyed by the “WATER LEVEL” display is required to reflect water level in the vessel. In addition, the operator must be given some indication that the display device is operating properly. Let the system be powered up at time 0. Then the requirement could be stated as follows:

$$LevelDisplay(t) = \begin{cases} 0.0 & \text{if } (0 \leq t < 4) \\ \lfloor t - 4 \rfloor \times 11.1 & \text{if } (4 \leq t < 14) \\ WaterLevel(t) \pm 1\% & \text{if } (t \geq 14) \end{cases}$$

Requirements feasibility. The requirements should specify behaviour for all cases that can arise, but at the same time should not demand the impossible. Hence:

Definition 5.

- The relation REQ is called *feasible with respect to* NAT if the following two conditions hold:
  1. domain(REQ) $\supseteq$ domain(NAT),
  2. domain(REQ $\cap$ NAT) = (domain(REQ) $\cap$ domain(NAT)).
- Note that two above conditions can be reduced to:
  3. domain(REQ $\cap$ NAT) = domain(NAT)

The first condition is quite obvious; equation (2) states that for each element $\boldsymbol{m}$ that is in the domains of both REQ and NAT, both relations must share at least one common element $\boldsymbol{c}$ to which this $\boldsymbol{m}$ can be mapped. Feasibility, in the above sense, means only that nature (as described by NAT) allows at least the required behaviour (as described by REQ); it does not mean that the functions involved are computable or that an implementation is practical.

Example 4. Let us consider NAT and REQ as illustrated in Examples 2 and 3. Note that the constraints given on the domain of NAT do not suggest that NAT defines any dependence between *WaterLevel* and *LevelDisplay.* If NAT contains all pairs of values (x, y) such that x belongs to the set of values of *WaterLevel* and y belongs to the set values of *LevelDisplay*, then domain(NAT) equals domain(REQ) and REQ is feasible with respect to NAT.

3.2 The system design document

During the system design stage we use a "clear-box" approach, i.e. we identify the computers within the computer system and describe how they communicate, with emphasis on the peripheral devices. Hence, new sets of quantities are distinguished, namely input and output registers, which are also represented by time-functions. Assume a system S, and $\boldsymbol{c}, \boldsymbol{m}$, C_S, M_S, as in Section 3.1. The additional environmental quantities can be classified as follows:

Definition 6.

- The set of additional environmental quantities is divided into two subsets:
  - *inputs*, quantities that can be read by the computers in the system, associated with input registers on those computers, and
  - *outputs*, quantities whose values are set by the computers in the system, associated with output registers on those computers.
- Let u be the number of input registers on the computers in the system S, and F_I a selected set of time-functions. Then:
  - inputs will be represented by a u-tuple $\boldsymbol{i} = (i_1, \ldots, i_u)$, where each i_k is a member of F_I and corresponds to the k-th input register,

  - the set of all $\boldsymbol{i}$ will be denoted by I_S.
- Let v be the number of output registers on the computers in the system S, and F_O a selected set of time-functions. Then:
  - outputs will be represented by a v-tuple $\boldsymbol{o} = (o_1, \ldots, o_v)$, where each o_k is a member of F_O and corresponds to the k-th output register,
  - the set of all $\boldsymbol{o}$ will be denoted by O_S.

Relation IN. The physical interpretation of the inputs and their connection with monitored quantities can be specified by a relation, which we call IN.

Definition 7.

- $\mathrm{IN} \subseteq M_S \times I_S$,
- $\mathrm{IN}(\boldsymbol{m}, \boldsymbol{i})$ if and only if $\boldsymbol{i}$ describes the possible values of the inputs when $\boldsymbol{m}$ describes the values of the monitored quantities.

One should note the following:

1. domain(IN) contains all possible instances of $\boldsymbol{m}$.
2. range(IN) contains possible instances of $\boldsymbol{i}$.
3. IN describes the behaviour of the input devices; it is a relation rather than a function because it gives some freedom to implement a measuring device with some imprecision.
4. It must be the case that domain(IN) $\supseteq$ domain(NAT), because the input device must transmit some value for every condition that can occur in nature.

Example 5. WaterLevel can be calculated from the measurement of differential pressure between two vertically separated points in the water vessel. The differential pressure gauge is connected to an analog-to-digital converter whose input values (to the computer) are represented by the input quantity DiffPres. That quantity is described as follows:

$$\left[(\mathrm{DiffPres}(t) \in [1, 254]) \Rightarrow \left(\frac{\mathrm{DiffPres}(t)}{255} \times 14.53\mathrm{cm} + 12.73\mathrm{cm}\right) \in I\right] \vee$$

$$[((\mathrm{DiffPres}(t) \in \{0, 255\}) \Rightarrow (\mathit{LevelDevice}(t) = \mathit{failed}))]$$

where $I = \mathit{WaterLevel}(t) \pm 0.34\mathrm{cm}$.

The differential pressure gauge is calibrated so that while the water level is between 13.0 cm and 27.0 cm, the value of DiffPres should be in the closed interval $[1, 254]$. If its value is either 0 or 255, then either the water level has drifted outside of the pressure gauge's calibrated range or the device has failed. It is to be assumed that if the value of DiffPres is 0 or 255 that the pressure gauge is not operating properly.

Relation OUT. The effects of the outputs and their connection with the controlled quantities can be specified by a relation, which we call OUT.

Definition 8.

- OUT $\subseteq O_S \times C_S$,
- OUT$(\boldsymbol{o}, \boldsymbol{c})$ if and only if $\boldsymbol{c}$ describes the possible values of the controlled quantities when $\boldsymbol{o}$ describes the values of the outputs.

One should note the following:

1. domain(OUT) contains all possible instances of $\boldsymbol{o}$.
2. range(OUT) contains possible instances of $\boldsymbol{c}$.
3. OUT describes the behaviour of the output devices; it is a relation rather than a function because it gives some freedom to implement an actuator with some imprecision.

Example 6. The value of PumpSwitch is affected through one bit of a discrete output register represented by the output quantity Shutdown. The effects of the relation OUT are described as follows:

$$(\text{Shutdown}(t), \text{PumpSwitch}(t)) \in \text{OUT}$$

where

$$\text{OUT} = \{(0, \textit{open}), (1, \textit{closed})\}$$

3.3 The software requirements document

The software requirements are determined by the system requirements document and system design documents. As mentioned earlier, the *software requirements document* can be seen as a combination of the two documents. It would contain the relations NAT (if it does not include all combinations of $\boldsymbol{m}$ and $\boldsymbol{c}$), REQ, IN, and OUT.

In the sequel we assume that all definitions and notational conventions from the previous sections apply. We will also assume that REQ is feasible with respect to NAT.

Relation SOF. The software implementation will yield a system with input-output behaviour that can be described by relation, which we call SOF.

Definition 9.

- SOF $\subseteq I_S \times O_S$,
- SOF$(\boldsymbol{i}, \boldsymbol{o})$ if and only if the software could produce the values described by $\boldsymbol{o}$ when the inputs are described by $\boldsymbol{i}$.

One should note the following:

1. domain(SOF) contains all possible instances of $\boldsymbol{i}$, i.e.

   $$\text{domain(SOF)} \supseteq \text{range(IN)}$$

   .
2. range(SOF) contains all instances of $\boldsymbol{o}$, i.e. range(SOF) $\subseteq$ domain(OUT).
3. SOF will be a function if the software is deterministic.

Software acceptability. The resulting software must satisfy all the requirements expressed by the relations. Hence the following:

Definition 10. For the software to be *acceptable*, the relation SOF must be such, that:

$$(*)\ \text{NAT}(\boldsymbol{m}, \boldsymbol{c}) \wedge \text{IN}(\boldsymbol{m}, \boldsymbol{i}) \wedge \text{SOF}(\boldsymbol{i}, \boldsymbol{o}) \wedge \text{OUT}(\boldsymbol{o}, \boldsymbol{c}) \Rightarrow \text{REQ}(\boldsymbol{m}, \boldsymbol{c})$$

$$\text{for all } \boldsymbol{m} \in M_S, \boldsymbol{i} \in I_S, \boldsymbol{o} \in O_S, \boldsymbol{c} \in C_S$$

One should note the following:

1. If one or more of the predicates IN($\boldsymbol{m}, \boldsymbol{i}$), OUT($\boldsymbol{o}, \boldsymbol{c}$), or NAT($\boldsymbol{m}, \boldsymbol{c}$) are false, then any software behaviour will be considered acceptable. (For example, if a given value of $\boldsymbol{m}$ is not in the domain of IN, the behaviour of acceptable software in that case is not constrained by the formula ($*$) above.)
2. Using relational composition [8] the formula ($*$) can be expressed in a more concise way:

   $$\text{NAT} \cap (\text{IN} \bullet \text{SOF} \bullet \text{OUT}) \subseteq \text{REQ}$$

3. If we assume that relations REQ, IN, OUT, and SOF are functions, we can use functional notation to rewrite ($*$) as follows:

   $$\boldsymbol{m} \in \text{domain(NAT)} \Rightarrow \text{REQ}(\boldsymbol{m}) = \text{OUT}(\text{SOF}(\text{IN}(\boldsymbol{m}))) \text{ for all } \boldsymbol{m} \in M_S$$

The writers of the requirements document must describe NAT, REQ, IN, and OUT. (The implementors determine SOF and verify ($*$).) A document of this type will require natural language in the description of the environmental quantities, but can otherwise be precise and mathematical. The use of natural language in the definition of the physical interpretation of mathematical quantities is unavoidable and quite usual in engineering.

4 Duration calculus and its application to WLMS

Duration calculus [4], an extension of interval temporal logic, was invented to allow reasoning about design and requirements for real-time, embedded systems.

[8] Given two relations $R \subseteq A \times B$, and $S \subseteq B \times C$, *relational composition* $R \bullet S$ is defined by: $R \bullet S = \{(a, c) \in a \times C \mid R(a, b) \wedge S(b, c) \text{ for some } b \in B\}$

To capture properties of piecewise continuous states the extended duration calculus (in short: EDC) is proposed in [5]. We believe it can be used in a natural way to express the relations introduced in previous sections (NAT, REQ, IN, OUT, and SOF). In this section we will formulate some aspects of the WLMS in EDC and verify formally that the formula $(*)$ from Section 3.3 holds. In the sequel we will consider a more general case of WLMS then the one described in [2]. Hence there will be some minor terminological differences and we will not use the particular values from [2].

4.1 Introduction to the extended calculus of duration

The extended duration calculus is a first order logic with special duration and interval operators, and with corresponding axioms. It is based on a mathematical theory of functions. The following definition summarizes the most important aspects of EDC and introduces terms needed for this paper. A more complete description of EDC is to be found in [5].

Definition 11.

- *States* are names, which are interpreted as a functions from reals (representing time) to reals.
- *State expressions* can be built from states by applying arithmetic operators (e.g: $+$). The result is defined pointwise (e.g. $(f+g)(t) \mathrel{\hat{=}} f(t)+g(t)$).
- *State assertions* can be built from states and state expressions by applying property names (e.g. Continuous (.)) or relational symbols. These assertions are interpreted as operators with a value domain restricted to the subset $\{0,1\}$ of reals (with 1 denoting the value "true").
- We assume, that any expression and assertion is finite and sectionally continuous, i.e. there can be only a finite number of discontinuities in a finite interval.
- For a state assertion P and an arbitrary observation interval (b,e), a *duration* of P, denoted $\int P$, is interpreted as follows: $\int P \mathrel{\hat{=}} \int_b^e P(t)\,dt$. Note, that:
  - $\int P$ is the accumulated time when $P=1$ within the observation interval,
  - $\int 1 = e-b$, i.e. it is the length of the observation interval; we will denote it in the sequel by ℓ.
- *Initial* value of a state expression f, denoted $b.f$ is interpreted as $\lim_{t\to b^+} f(t)$ for the arbitrary, nonempty observation interval (b,e).
- *Final* value of a state expression f, denoted $e.f$ is interpreted as $\lim_{t\to e^-} f(t)$.
- *Duration terms* are built from global variables (which are interpreted as real numbers), durations, initial and final values and n-ary operators on duration terms.
- From duration terms we can build *duration formulae*. We introduce now the most important kinds of such formulae:
  - Simple predicates can be formed in the two following ways. Any property Q which holds for any proper (non empty) interval in the mathematical theory can be lifted to a simple predicate $\lceil Q \rceil$, which is true if and

only if $(\ell > 0) \wedge Q(x)$ for all $x \in (b, e)$. Also for a given valid relation from mathematical theory $R(r_1, \ldots, r_n)$ over non-negative real numbers $r_1, \ldots, r_n$, a simple predicate is formed by substituting duration terms for $r_1, \ldots, r_n$.
  - We use $\lceil\rceil$ to denote the empty interval, i.e. $\lceil\rceil \mathrel{\hat{=}} \ell = 0$.
  - We can combine duration formulae using the boolean logic operators which have here standard meaning, and the interval temporal logic operator "chop" (denoted ";"). The latter one is defined as follows. Let A and B be duration formulae. The formula $A; B$ is true for any interval which can be split into two subintervals, of which the first satisfies A and the second satisfies B.
  - The modal operators $\Box$ and $\Diamond$ are introduced as abbreviations of the following formulae:
    $\Diamond D \mathrel{\hat{=}} \text{true}; D; \text{true}$
    ($\Diamond D$ holds if there is a subinterval of a given interval where D holds)
    $\Box D \mathrel{\hat{=}} \neg(\Diamond \neg D)$
    (D holds in any subinterval of a given interval)
  - We say that D_1 is *τ-followed* by D_2 and write it $D_1 \xrightarrow{\tau} D_2$, if: $(D_1; \ell = \tau) \Rightarrow (D_1; D_2)$
- We use the following precedence rules (with the highest priority first):
  - $\int$, b., e.
  - arithmetic operators
  - relational operators
  - $\neg, \Box, \Diamond$
  - ;
  - $\vee, \wedge$
  - $\Rightarrow$

4.2 The WLMS requirements specification

We will limit our considerations only to that part of the WLMS that is responsible for switching the pump on and off depending on the level of water (cf. Section 3.1 Example 1). The monitored quantity is represented by the function *Water-Level*, which we will abbreviate in the sequel as WL. The controlled quantity is represented by the function *PumpSwitch*, which we will abbreviate in the sequel as PS. Hence $\boldsymbol{m}$ = (WL) and $\boldsymbol{c}$ = (PS). We will slightly redefine the function *PumpSwitch*: its range is a subset $\{0, 1\}$ of reals (*closed* being replaced by 1, *open* by 0), i.e. :

$$\mathrm{PS}(t) = \begin{cases} 1, \text{if the switch contacts are closed at time } t \\ 0, \text{if the switch contacts are open at time } t \end{cases}$$

According to the approach presented in the previous section, we should now formulate the relations NAT and REQ. Let us begin with the latter.

Relation REQ. To be able to specify the informal statement "switch the pump off if the level of water is too high", we will assume that we are given four real constants: $High$, Δ, T_1, T_2, such that $High > \Delta > 0$, $T_1 > 0, T_2 > 0$, and will formulate our requirements as follows:

1. If the level of water remains above the high mark $High$ for at least time T_1, then the pump switch should be opened after the period T_1, and
2. If the level of water remains below $(High - \Delta)$ for at least time T_2, then the pump switch should be closed after the period T_2.

The above requirements can be expressed in DC as follows:

$$\begin{aligned}\mathrm{REQ}_1 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \\ \Box(\lceil \mathrm{WL} > High \rceil \wedge (\ell > T_1) \Rightarrow \mathrm{true}; \lceil \mathrm{PS} = 0 \rceil)\}\end{aligned}$$

$$\begin{aligned}\mathrm{REQ}_2 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \\ \Box(\lceil \mathrm{WL} < (High - \Delta) \rceil \wedge (\ell > T_2) \Rightarrow \mathrm{true}; \lceil \mathrm{PS} = 1 \rceil)\}\end{aligned}$$

The relation REQ is then defined:

$$\mathrm{REQ} \mathrel{\hat{=}} \mathrm{REQ}_1 \cap \mathrm{REQ}_2$$

Relation NAT. NAT should describe constraints imposed by nature. Here we should have:

$$\mathrm{NAT}_1 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \lceil 0 < \mathrm{WL} < \mathrm{Max} \rceil\}$$

$$\mathrm{NAT}_2 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \Box(\mid e.\mathrm{WL} - b.\mathrm{WL} \mid \leq c \cdot \ell \vee \ell = 0)\}$$

$$\mathrm{NAT}_3 \mathrel{\hat{=}} \lceil Continuous\ (\mathrm{WL}) \rceil \vee \lceil\rceil$$

$$\mathrm{NAT} \mathrel{\hat{=}} \mathrm{NAT}_1 \cap \mathrm{NAT}_2 \cap \mathrm{NAT}_3$$

The first condition states that the water level is always greater than 0, and less then a certain constant Max.

The second one bounds the pump speed by a constant c.

The third condition states that the water level is continuous.

4.3 The WLMS design specification

The relation IN and OUT will be very simple in our case. We will assume that the values of the function WL are represented by the input [9] register $\mathrm{WL}_\circ$, and that the pump switch is actuated by the output register $\mathrm{PS}_\circ$, i.e. $\boldsymbol{i} = (\mathrm{WL}_\circ)$, and $\boldsymbol{o} = (\mathrm{PS}_\circ)$.

There will be only a tolerance requirement on $\mathrm{WL}_\circ$, which is expressed as follows:

$$\mathrm{IN} \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{WL}_\circ) \in M_S \times I_S \mid \Box(\lceil \mid \mathrm{WL} - \mathrm{WL}_\circ \mid < \varepsilon_1 \rceil \vee \lceil\rceil)\}$$

where ε_1 is a constant such, that $0 < \varepsilon_1 < \Delta/2$.

As far as output is concerned we will make an idealistic assumption that $\mathrm{PS}_\circ$ equals PS almost all the time:

$$\mathrm{OUT} \mathrel{\hat{=}} \{(\mathrm{PS}_\circ, \mathrm{PS}) \in O_S \times C_S \mid \Box(\lceil \mathrm{PS} = \mathrm{PS}_\circ \rceil \vee \lceil\rceil)\}$$

[9] An input/output register is usually a part of a digital computer and in such a case has discrete values.

4.4 The WLMS software specification

Software specification SOF is a little bit more complicated. To describe it in terms of EDC we will need a simple model for the program. The simplest one is an automaton. We will design a three phase [10] automaton with a set of phases $\{P_0, P_1, P_2\}$. We will use a state name Phase ranging over$\{P_0, P_1, P_2\}$ to denote the phase the automaton is in. We will also use an abbreviation $\lceil P_i \rceil$ to denote that $\lceil \mathrm{Phase} = P_i \rceil$.

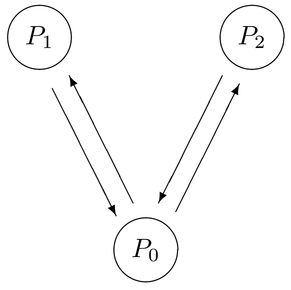

Figure 1: Three phase automaton model.

When the measured water level is above $High - \varepsilon_1$, the automaton changes its phase from P_0 to P_2. When the level is below $High - \Delta + \varepsilon_1$, the automaton changes its phase from P_0 to P_1. We know that $0 < \varepsilon_1 < \frac{\Delta}{2}$, hence $High - \Delta + \varepsilon_1 < High - \varepsilon_1$. Between these two values the automaton changes its phase to P_0. These transition properties are described by the following four conditions:

$$\mathrm{Sof}_1 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ > High - \varepsilon_1 \rceil \Rightarrow (\lceil P_0 \rceil \xrightarrow{\tau_1} \lceil P_2 \rceil))$$

$$\mathrm{Sof}_2 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ \le High - \varepsilon_1 \rceil \Rightarrow (\lceil P_2 \rceil \xrightarrow{\tau_2} \lceil P_0 \rceil; \mathrm{true}))$$

$$\mathrm{Sof}_3 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ < High - \Delta + \varepsilon_1 \rceil \Rightarrow (\lceil P_0 \rceil \xrightarrow{\tau_3} \lceil P_1 \rceil))$$

$$\mathrm{Sof}_4 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ \ge High - \Delta + \varepsilon_1 \rceil \Rightarrow (\lceil P_1 \rceil \xrightarrow{\tau_4} \lceil P_0 \rceil; \mathrm{true})).$$

The positive constants $\tau_1, \ldots, \tau_4$ describe transition delays. P_0 is the intermediate phase between P_1 and P_2 (cf. figure 1), what is expressed by "true" at the end of Sof_2 and Sof_4.

When the automaton is in phase P_2 the pump is turned off and when it is in phase P_1 the pump is turned on. Delays in turning the pump on and off are limited by two constants: $\delta_1 > 0$ and $\delta_2 > 0$. Behaviour of the pump is not determined when the automaton is in phase P_0. These requirements are expressed by the following two formulae:

$$\mathrm{Sof}_5 \mathrel{\hat{=}} \Box((\lceil P_1 \rceil \wedge (\ell > \delta_1)) \Rightarrow \mathrm{true}; \lceil \mathrm{PS}_\circ = 1 \rceil)$$

$$\mathrm{Sof}_6 \mathrel{\hat{=}} \Box((\lceil P_2 \rceil \wedge (\ell > \delta_2)) \Rightarrow \mathrm{true}; \lceil \mathrm{PS}_\circ = 0 \rceil).$$

[10] The word *state* is used already to denote a time function in EDC. To avoid ambiguity we will use the word *phase* to denote a state of an automaton.

Formulae $\mathrm{Sof}_1, \ldots, \mathrm{Sof}_4$ do not specify the behaviour of the automaton in all situations. For instance, when the system is started, the water level may be low (respectively high) and the automaton may already be in phase P_1 (respectively P_2). Hence, there are two conditions specifying that if the automaton is in state P_1 (respectively, P_2) and $\mathrm{WL}_\circ$ is less than $High - \Delta + \varepsilon_1$ (respectively, is greater than $High - \varepsilon_1$), then the automaton stays in phase P_1 (respectively, P_2):

$$\mathrm{Sof}_7 \mathrel{\hat{=}} \Box((\lceil P_1 \rceil ; \mathrm{true} \wedge \lceil \mathrm{WL}_\circ < High - \Delta + \varepsilon_1 \rceil) \Rightarrow \lceil P_1 \rceil)$$

$$\mathrm{Sof}_8 \mathrel{\hat{=}} \Box((\lceil P_2 \rceil ; \mathrm{true} \wedge \lceil \mathrm{WL}_\circ > High - \varepsilon_1 \rceil) \Rightarrow \lceil P_2 \rceil).$$

One can notice that formula Sof_7 (respectively Sof_8) forces the automaton to stay in phase P_1 (respectively P_2) as long as the water level is low (respectively high).

Due to the finite variability of states, each phase covers an interval, what is expressed by the formula:

$$\mathrm{Sof}_9 \mathrel{\hat{=}} \Box(\lceil P_0 \rceil ; \mathrm{true} \vee \lceil P_1 \rceil ; \mathrm{true} \vee \lceil P_2 \rceil ; \mathrm{true} \vee \lceil \rceil).$$

The whole software specification is now defined as follows:

$$\mathrm{SOF} \mathrel{\hat{=}} \{(\mathrm{WL}_\circ, \mathrm{PS}_\circ) \in I_S \times O_S \mid \bigwedge_{i=1,\ldots,9} \mathrm{Sof}_i, \text{ for some Phase}\}$$

We will have to make some additional assumptions on $\tau_1, \ldots, \tau_4$, δ_1 and δ_2 constants because there is no correlation between them, and the constants T_1 and T_2. These assumptions naturally appear in a software verification.

4.5 An example of software acceptability verification

Verification of software acceptability is a proof that the software requirements, input and output specifications and the knowledge of the nature imply the system requirements [11]. The complete proof can be found in the Appendix. Here we will show the main part of it only. The system requirements consist of two conditions REQ_1 and REQ_2. Let us consider REQ_1.

$$\mathrm{REQ}_1 = \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \Box(\lceil \mathrm{WL} > High \rceil \wedge (\ell > T_1) \Rightarrow$$

$$\mathrm{true}; \lceil \mathrm{PS} = 0 \rceil)\}$$

For arbitrarily chosen pair $(\mathrm{WL}, \mathrm{PS}) \in \mathrm{REQ}_1$ we have:

$$\Box(\lceil \mathrm{WL} > High \rceil) \wedge (\ell > T_1) \Rightarrow \mathrm{true}; \lceil \mathrm{PS} = 0 \rceil).$$

Assume that:

$$\lceil \mathrm{WL} > High \rceil \wedge (\ell > T_1).$$

[11] Since in our case the required actions (switching the pump on and off) do not depend on conditions expressed by NAT, the latter relation will not be used in the verification process.

For any $\mathrm{WL}_\circ$ satisfying $(\mathrm{WL}, \mathrm{WL}_\circ) \in \mathrm{IN}$ we obtain:

$$\lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil \wedge (\ell > T_1).$$

From Sof_9 we know that initially the automaton is in one of three phases: P_0, P_1, P_2. Assume that it is P_1.

$$\lceil P_1 \rceil ; \mathit{true}$$

From Sof_4 we have:

$$\lceil P_1 \rceil \xrightarrow{\tau_4} \lceil P_0 \rceil ; \mathrm{true}.$$

If $T_1 > \tau_4$ then

$$(\ell > \tau_4) ; \lceil P_0 \rceil ; \mathrm{true} ; (\ell > T_1 - \tau_4) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil .$$

From that we obtain:

$$(\ell > \tau_4) ; (\lceil P_0 \rceil ; \mathrm{true} \wedge (\ell > T_1 - \tau_4) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

If $T_1 - \tau_4 > \tau_1$, then from Sof_1 we know that

$$(\ell > \tau_4) ; ((\ell > \tau_1) ; (\lceil P_2 \rceil ; \mathrm{true} \wedge (\ell > T_1 - \tau_4 - \tau_1)) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil),$$

what can be weaken to:

$$(\ell > \tau_4 + \tau_1) ; (\lceil P_2 \rceil ; \mathrm{true} \wedge (\ell > T_1 - \tau_4 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From Sof_8 we obtain:

$$(\ell > \tau_4 + \tau_1) ; (\lceil P_2 \rceil \wedge (\ell > T_1 - \tau_4 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

If $T_1 - \tau_4 - \tau_1 > \delta_2$ then from Sof_6 we know that

$$(\ell > \tau_4 + \tau_1) ; \mathrm{true} ; \lceil \mathrm{PS}_\circ = 0 \rceil .$$

From OUT we have

$$\mathrm{true} ; \lceil \mathrm{PS} = 0 \rceil$$

The successor of the implication have been derived from the predecessor under the assumption that $T_1 > \tau_1 + \tau_4 + \delta_2$.

If the automaton is initially in phase P_0 then the proof is similar to the second half of that one above. If it starts in phase P_2 then from IN and Sof_8 we obtain

$$\lceil P_2 \rceil$$

and from Sof_6 and OUT

$$\mathrm{true} ; \lceil \mathrm{PS} = 0 \rceil .$$

The fact that WL, $\mathrm{WL}_\circ$ and PS had been chosen arbitrarily completes the proof of REQ_1.

We can similarly prove REQ_2. We will get the assumption that $T_2 > \tau_2 + \tau_3 + \delta_1$. The assumptions on constants $\tau_1, \ldots, \tau_4, \delta_1, \delta_2$ should form one more software requirement limiting these constants and leaving their concrete values up to software designers.

5 Conclusions

This work is a first step towards merging two distinct approaches to formal specification of real-time systems. We have shown how one can express the relations described in requirements documents in terms of extended duration calculus. This allows us to verify formally each design step, and in particular to check whether certain design decisions are correct from the requirements point of view.

We expect this first step to be followed by further investigations and by elaboration of more examples.

Acknowledgments

This work was partially supported by:

(a) In Canada:
- Telecommunications Research Institute of Ontario (TRIO),
- Atomic Energy Control Board (AECB),
- Natural Sciences and Engineering Research Council of Canada (NSERC);

(b) In Europe:
- Commission of the European Communities (CEC) under the ESPRIT programme in the field of Basic Research Action proj. no. 3104: "ProCoS: Provably Correct Systems".

References

1. Parnas, D.L., and Madey, J., "Functional Documentation for Computer Systems Engineering (Version 2)", *CRL Report 237*, Telecommunications Research Institute of Ontario (TRIO), McMaster University, Hamilton, ON, September 1991, 14pp.
2. van Schouwen, A.J., "The A-7 Requirements Model: Re-examination for Real-Time Systems and an Application to Monitoring Systems", *Technical Report 90–276*, Queen's University, C&IS, Telecommunications Research Institute of Ontario (TRIO), Kingston, ON, January 1991 (revision 3), 94pp.
3. van Schouwen, A.J., Parnas, D.L., and Madey, J., "Documentation of Requirements for Computer Systems", Proceeding of the IEEE International Symposium on Requirements Engineering, San Diego, California, January 4–6, 1993.
4. Zhou Chaochen, C.A.R. Hoare, Anders P. Ravn, "A Calculus of Durations", *Information Processing Letters 40, 5*, pp. 269 – 276, 1992.
5. Zhou Chaochen, Michael R. Hansen, Anders P. Ravn "An Extended Duration Calculus for Hybrid Real-Time Systems" in this volume.

Appendix: Software verification

We have introduced the following definitions of REQ, NAT, IN, OUT and SOF:

$$\mathrm{REQ}_1 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid$$

$$\Box(\lceil \mathrm{WL} > \mathit{High} \rceil \wedge (\ell > T_1) \Rightarrow \mathrm{true}; \lceil \mathrm{PS} = 0 \rceil)\}$$

$\mathrm{REQ}_2 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \Box(\lceil \mathrm{WL} < (\mathit{High} - \mathit{\Delta})\rceil \wedge (\ell > T_2) \Rightarrow$

$\mathrm{true}; \lceil \mathrm{PS} = 1 \rceil)\}$

$\mathrm{REQ} \mathrel{\hat{=}} \mathrm{REQ}_1 \cap \mathrm{REQ}_2$

$\mathrm{NAT}_1 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \lceil 0 < \mathrm{WL} < \mathrm{Max} \rceil\}$

$\mathrm{NAT}_2 \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{PS}) \in M_S \times C_S \mid \Box(\mid e.\mathrm{WL} - b.\mathrm{WL} \mid \leq c \cdot \ell \vee \ell = 0)\}$

$\mathrm{NAT}_3 \mathrel{\hat{=}} \lceil \mathit{Continuous}\ (\mathrm{WL}) \rceil \vee \lceil\rceil$

$\mathrm{NAT} \mathrel{\hat{=}} \mathrm{NAT}_1 \cap \mathrm{NAT}_2 \cap \mathrm{NAT}_3$

$\mathrm{IN} \mathrel{\hat{=}} \{(\mathrm{WL}, \mathrm{WL}_\circ) \in M_S \times I_S \mid \Box(\lceil \mid \mathrm{WL} - \mathrm{WL}_\circ \mid < \varepsilon_1 \rceil \vee \lceil\rceil)\}$

$\mathrm{OUT} \mathrel{\hat{=}} \{(\mathrm{PS}_\circ, \mathrm{PS}) \in O_S \times C_S \mid \Box(\lceil \mathrm{PS} = \mathrm{PS}_\circ \rceil \vee \lceil\rceil)\}$

$\mathrm{Sof}_1 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil \Rightarrow (\lceil P_0 \rceil \xrightarrow{\tau_1} \lceil P_2 \rceil))$

$\mathrm{Sof}_2 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ \leq \mathit{High} - \varepsilon_1 \rceil \Rightarrow (\lceil P_2 \rceil \xrightarrow{\tau_2} \lceil P_0 \rceil; \mathrm{true}))$

$\mathrm{Sof}_3 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil \Rightarrow (\lceil P_0 \rceil \xrightarrow{\tau_3} \lceil P_1 \rceil))$

$\mathrm{Sof}_4 \mathrel{\hat{=}} \Box(\lceil \mathrm{WL}_\circ \geq \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil \Rightarrow (\lceil P_1 \rceil \xrightarrow{\tau_4} \lceil P_0 \rceil; \mathrm{true}))$

$\mathrm{Sof}_5 \mathrel{\hat{=}} \Box((\lceil P_1 \rceil \wedge (\ell > \delta_1)) \Rightarrow \mathrm{true}; \lceil \mathrm{PS}_\circ = 1 \rceil)$

$\mathrm{Sof}_6 \mathrel{\hat{=}} \Box((\lceil P_2 \rceil \wedge (\ell > \delta_2)) \Rightarrow \mathrm{true}; \lceil \mathrm{PS}_\circ = 0 \rceil)$

$\mathrm{Sof}_7 \mathrel{\hat{=}} \Box((\lceil P_1 \rceil; \mathrm{true} \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil) \Rightarrow \lceil P_1 \rceil)$

$\mathrm{Sof}_8 \mathrel{\hat{=}} \Box((\lceil P_2 \rceil; \mathrm{true} \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil) \Rightarrow \lceil P_2 \rceil)$

$\mathrm{Sof}_9 \mathrel{\hat{=}} \Box(\lceil P_0 \rceil; \mathrm{true} \vee \lceil P_1 \rceil; \mathrm{true} \vee \lceil P_2 \rceil; \mathrm{true} \vee \lceil\rceil)$

$$\mathrm{SOF} \mathrel{\hat{=}} \{(\mathrm{WL}_\circ, \mathrm{PS}_\circ) \in I_S \times O_S \mid \bigwedge_{i=1,\ldots,9} \mathrm{Sof}_i, \text{ for some Phase}\}$$

We assumed that all constants in considered example are positive. We have also derived additional assumptions on constants:

$T_1 > \tau_1 + \tau_4 + \delta_2$

$T_2 > \tau_2 + \tau_3 + \delta_1$

which should be included into SOF. In particular design constants T_1, T_2 and ε_1 should be replaced by concrete numbers. Constants $\tau_1, \ldots\ \tau_4, \delta_1$ and δ_2 may be left up to software designers.

To prove REQ_1 we assume that for a chosen pair $(\mathrm{WL}, \mathrm{PS}) \in \mathrm{REQ}_1$ and for a chosen period of time, the following holds:

$$\lceil \mathrm{WL} > \mathit{High} \rceil \wedge (\ell > T_1).$$

For any $\mathrm{WL}_\circ$ such that $(\mathrm{WL}, \mathrm{WL}_\circ) \in \mathrm{IN}$ we have:

$$\lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil \wedge (\ell > T_1).$$

From Sof_9 we know that initially the automaton is in one of three phases: P_0, P_1, P_2. Hence, we have to consider the three cases.

1° Assume that:

$$\lceil P_1 \rceil ; \mathit{true}.$$

We have:

$$\lceil P_1 \rceil ; \mathit{true} \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil \wedge (\ell > T_1),$$

and from that:

$$\lceil P_1 \rceil ; (\ell = \tau_4) ; (\ell > T_1 - \tau_4) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil.$$

From Sof_4 we obtain:

$$(\lceil P_1 \rceil ; (\ell = \tau_4) \wedge \lceil P_1 \rceil \xrightarrow{\tau_4} \lceil P_0 \rceil ; \mathrm{true}) ; (\ell > T_1 - \tau_4) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil.$$

We can simplify it:

$$(\ell > \tau_4) ; \lceil P_0 \rceil ; (\ell > T_1 - \tau_4) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil.$$

We can weaken it:

$$(\ell > \tau_4) ; (\lceil P_0 \rceil ; (\ell > T_1 - \tau_4) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From that we obtain:

$$(\ell > \tau_4) ; (\lceil P_0 \rceil ; (\ell = \tau_1) ; (\ell > T_1 - \tau_4 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From Sof_1 we know that:

$$(\ell > \tau_4)((\lceil P_0 \rceil ; (\ell = \tau_1) \wedge \lceil P_0 \rceil \xrightarrow{\tau_1} \lceil P_2 \rceil) ; (\ell > T_1 - \tau_4 - T_1) \wedge$$
$$\lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil),$$

hence:

$$(\ell > \tau_4) ; ((\ell > \tau_1) ; \lceil P_2 \rceil ; (\ell > T_1 - \tau_4 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From that we obtain:

$$(\ell > \tau_4) ; (\ell > \tau_1) ; (\lceil P_2 \rceil ; \mathit{true} \wedge (\ell > T_1 - \tau_4 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

We can simplify it:

$$(\ell > \tau_4 + \tau_1) ; (\lceil P_2 \rceil ; \mathrm{true} \wedge (\ell > T_1 - \tau_4 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From Sof_8 we have:

$$(\ell > \tau_4 + \tau_1); (\lceil P_2 \rceil \wedge (\ell > T_1 - \tau_4 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From Sof_6 we know that:

$$(\ell > \tau_4 + \tau_1); \mathrm{true}; \lceil \mathrm{PS}_\circ = 0 \rceil.$$

From OUT we have:

$$(\ell > \tau_4 + \tau_1); \mathrm{true}; \lceil \mathrm{PS} = 0 \rceil,$$

hence:

$$\mathrm{true}; \lceil \mathrm{PS} = 0 \rceil.$$

Which is the desired consequence.

2° Assume that:

$$\lceil P_0 \rceil; \mathit{true}.$$

We have:

$$\lceil P_0 \rceil; \mathit{true} \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil \wedge (\ell > T_1),$$

and from that:

$$\lceil P_0 \rceil; (\ell = \tau_1); (\ell > T_1 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil.$$

From Sof_1 we know that:

$$(\lceil P_0 \rceil; (\ell = \tau_1) \wedge \lceil P_0 \rceil \xrightarrow{\tau_1} \lceil P_2 \rceil); (\ell > T_1 - T_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil.$$

We can simplify it:

$$(\ell > \tau_1); \lceil P_2 \rceil; (\ell > T_1 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil.$$

We can simplify it:

$$(\ell > \tau_1); (\lceil P_2 \rceil; \mathit{true} \wedge (\ell > T_1 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From Sof_8 we have:

$$(\ell > \tau_1); (\lceil P_2 \rceil \wedge (\ell > T_1 - \tau_1) \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil).$$

From Sof_6 we obtain:

$$(\ell > \tau_1); \mathrm{true}; \lceil \mathrm{PS}_\circ = 0 \rceil.$$

From OUT we have:

$$(\ell > \tau_1); \mathrm{true}; \lceil \mathrm{PS} = 0 \rceil,$$

hence:

$$\mathrm{true}; \lceil \mathrm{PS} = 0 \rceil.$$

Which is the desired consequence.

3° Assume that:

$$\lceil P_2 \rceil; \mathit{true}.$$

We have:

$$\lceil P_2 \rceil; \mathit{true} \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil \wedge (\ell > T_1).$$

From Sof_8 we obtain:

$$\lceil P_2 \rceil \wedge \lceil \mathrm{WL}_\circ > \mathit{High} - \varepsilon_1 \rceil \wedge (\ell > T_1).$$

From Sof_6 we know that:

$$\mathrm{true}; \lceil \mathrm{PS}_\circ = 0 \rceil.$$

From OUT we finally get:

$$\mathrm{true}; \lceil \mathrm{PS} = 0 \rceil$$

Which is the desired consequence.

We have considered three possible cases. The fact that we have arbitrarily chosen WL, $\mathrm{WL}_\circ$ and PS ends the proof of REQ_1.

A proof of REQ_2 is very similar to the proof of REQ_1. To prove REQ_2 we assume that for a chosen pair $(\mathrm{WL}, \mathrm{PS}) \in \mathrm{REQ}_2$ and for a chosen period of time the following holds:

$$\lceil \mathrm{WL} < \mathit{High} - \Delta \rceil \wedge (\ell > T_2).$$

For any $\mathrm{WL}_\circ$such that $(\mathrm{WL}, \mathrm{WL}_\circ) \in \mathrm{IN}$ we have:

$$\lceil \mathrm{WL}_\circ < \mathit{High} - \Delta + \varepsilon_1 \rceil \wedge (\ell > T_2).$$

From Sof_9 we know that initially the automaton is in one of three phases: P_0, P_1, P_2. Hence, we have to consider the three cases.

1° Assume that:

$$\lceil P_2 \rceil; \mathit{true}.$$

We have:

$$\lceil P_2 \rceil; \mathit{true} \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \Delta + \varepsilon_1 \rceil \wedge (\ell > T_2),$$

and from that:

$$\lceil P_2 \rceil; (\ell = \tau_2); (\ell > T_2 - \tau_2) \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \Delta + \varepsilon_1 \rceil.$$

From Sof_2 we obtain:

$$(\lceil P_2 \rceil; (\ell = \tau_2) \wedge \lceil P_2 \rceil \xrightarrow{\tau_2} \lceil P_0 \rceil; \mathrm{true}); (\ell > T_2 - \tau_2) \wedge$$

$$\lceil \mathrm{WL}_\circ < \mathit{High} - \Delta + \varepsilon_1 \rceil,$$

hence:

$$(\ell > \tau_2); \lceil P_0 \rceil; (\ell > T_2 - \tau_2) \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil.$$

We can weaken it:

$$(\ell > \tau_2); (\lceil P_0 \rceil; (\ell = \tau_3); (\ell > T_2 - \tau_2 - \tau_3) \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil).$$

From Sof_3 we know that:

$$(\ell > \tau_2); ((\lceil P_0 \rceil; (\ell = \tau_3) \wedge \lceil P_0 \rceil \xrightarrow{\tau_3} \lceil P_1 \rceil); (\ell > T_2 - \tau_2 - \tau_3) \wedge$$

$$\lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil),$$

hence:

$$(\ell > \tau_2); ((\ell > \tau_3); \lceil P_1 \rceil; (\ell > T_2 - \tau_2 - \tau_3) \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil).$$

From that we obtain:

$$(\ell > \tau_2); (\ell > \tau_3); (\lceil P_1 \rceil; \mathit{true} \wedge (\ell > T_2 - \tau_2 - \tau_3) \wedge$$

$$\lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil).$$

We can simplify it:

$$(\ell > \tau_2 + \tau_3); (\lceil P_1 \rceil; \mathrm{true} \wedge (\ell > T_2 - \tau_2 - \tau_3) \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil).$$

From Sof_7 we have:

$$(\ell > \tau_2 + \tau_3); (\lceil P_1 \rceil \wedge (\ell > T_2 - \tau_2 - \tau_3) \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil).$$

From Sof_5 we know that:

$$(\ell > \tau_2 + \tau_3); \mathrm{true}; \lceil \mathrm{PS}_\circ = 1 \rceil.$$

From OUT we obtain:

$$(\ell > \tau_2 + \tau_3); \mathrm{true}; \lceil \mathrm{PS} = 1 \rceil,$$

hence:

$$\mathrm{true}; \lceil \mathrm{PS} = 1 \rceil.$$

Which is the desired consequence.

2° Assume that:

$$\lceil P_0 \rceil; \mathit{true}.$$

We have:

$$\lceil P_0 \rceil; \mathit{true} \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil \wedge (\ell > T_2),$$

and from that:

$$\lceil P_0 \rceil; (\ell = \tau_3); (\ell > T_2 - \tau_3) \wedge \lceil \mathrm{WL}_\circ < \mathit{High} - \mathit{\Delta} + \varepsilon_1 \rceil.$$

From Sof_3 we know that:

$$(\lceil P_0\rceil;(\ell=\tau_3)\wedge\lceil P_0\rceil\xrightarrow{\tau_3}\lceil P_1\rceil);(\ell>T_2-\tau_3)\wedge\lceil \mathrm{WL}_\circ<\mathit{High}-\varDelta+\varepsilon_1\rceil,$$

hence:

$$(\ell>\tau_3);\lceil P_1\rceil;(\ell>T_2-\tau_3)\wedge\lceil \mathrm{WL}_\circ<\mathit{High}-\varDelta+\varepsilon_1\rceil.$$

We can weaken it:

$$(\ell>\tau_3);(\lceil P_1\rceil;(\ell>T_2-\tau_3)\wedge\lceil \mathrm{WL}_\circ<\mathit{High}-\varDelta+\varepsilon_1\rceil).$$

From that we have:

$$(\ell>\tau_3);(\lceil P_1\rceil;\mathit{true}\wedge(\ell>T_2-\tau_3)\wedge\lceil \mathrm{WL}_\circ<\mathit{High}-\varDelta+\varepsilon_1\rceil).$$

From Sof_7 we obtain:

$$(\ell>\tau_3);(\lceil P_1\rceil\wedge(\ell>T_2-\tau_3)\wedge\lceil \mathrm{WL}_\circ<\mathit{High}-\varDelta+\varepsilon_1\rceil).$$

From Sof_5 we know that:

$$(\ell>\tau_3);\mathrm{true};\lceil \mathrm{PS}_\circ=1\rceil.$$

From OUT we have:

$$(\ell>\tau_3);\mathrm{true};\lceil \mathrm{PS}=1\rceil,$$

hence:

$$\mathrm{true};\lceil \mathrm{PS}=1\rceil.$$

Which is the desired consequence.

3° Assume that:

$$\lceil P_1\rceil;\mathit{true}.$$

We have:

$$\lceil P_1\rceil;\mathit{true}\wedge\lceil \mathrm{WL}_\circ<\mathit{High}-\varDelta+\varepsilon_1\rceil\wedge(\ell>T_2).$$

From Sof_7 we obtain:

$$\lceil P_1\rceil\wedge\lceil \mathrm{WL}_\circ<\mathit{High}-\varDelta+\varepsilon_1\rceil\wedge(\ell>T_2).$$

From Sof_5 we know that:

$$\mathrm{true};\lceil \mathrm{PS}_\circ=1\rceil.$$

From OUT we finally get:

$$\mathrm{true};\lceil \mathrm{PS}=1\rceil.$$

Which is the desired consequence.

The fact that we have arbitrarily chosen WL, $\mathrm{WL}_\circ$ and PS ends the proof of REQ_2. These two proofs give the full software acceptability verification.