Modeling Aircraft Mission Computer Task Rates

Jin Song Dong[1]* Brendan P. Mahony[2]** Neale Fulton[3]***

[1] School of Computing, National University of Singapore
dongjs@comp.nus.edu.sg

[2] Defence Science and Technology Organisation (DSTO), Australia
Brendan.Mahony@dsto.defence.gov.au

[3] Commonwealth Science and Industrial Research Organisation (CSIRO), Australia
neale.fulton@cmis.csiro.au

Abstract. Safety critical systems, such as aviation systems controlled by software, often have hard real-time requirements. Producing the correct result at the right time is the fundamental goal of such systems. Formally specifying the system functions and the timing requirements is the crucial step towards achieving such a goal. Aviation systems often need to be modified or upgraded on a regular basis, i.e. functionality and timing constraints may be altered. Therefore, the formal specification of such systems needs to be easily reused, maintained and modified. This paper demonstrates how an aircraft mission computer task rate sequences with their interactions can be formalised in TCOZ (an integrated notation of Timed CSP and Object-Z).

Keywords: formal object modeling, timed concurrent specification, aviation system.

1 Introduction

The use of formal methods on critical systems has been growing in the last few years [1, 10]. In particular, there are number of successful applications of using formal specification techniques in the aviation industry. For example, Leveson's group produced a formal requirements specification for the Traffic Collision Avoidance System (TCAS) II by using RSML [15]. The software company Praxis used VDM and CCS to deliver to the UK Civil Aviation Authority the CCF Display Information System (CDIS) [11] (a central part of the new air traffic management system for London's airspace).

* This paper was written when Jin Song Dong was working as a research scientist at CSIRO Mathematical and Information Sciences,

** This paper was written during Brendan Mahony's visit to the CSIRO Mathematical and Information Sciences as part of the DSTO/CSIRO Fellowship Programme.

*** Neale Fulton wishes to acknowledge industry support and interest for this project from Royal Australian Air Force, RAAF Base Williamtown and from GEC Marconi Systems Pty Ltd.

Aviation systems often need to be modified or upgraded on a regular basis. For example, the maintenance of the Mission Computer (MC) Operational Flight Program (OFP) may involve the modification of the baseline requirements for the purpose of enhancing performance or adding new capabilities. Therefore maintenance requirements further compound the complexity of the requirements management problem. This potentially increases the benefit of adopting formal requirements modeling techniques.

Recently the Royal Australian Air Force (RAAF) has been considering an upgrade to the F/A-18 aircraft in Australia. This upgrade may well involve the modification of Mission Computer systems. Maintaining correct functionality for the upgraded F/A-18 is therefore a major concern for the RAAF. This particular problem received interest from CSIRO and DSTO to support a joint Research Fellowship to investigate specification of hard real-time characteristics by formal method approaches.

Neale Fulton, an avionics domain expert with 25 years aviation experience, identified that precise MC OFP task schedule requirements are crucial for this upgrade. In preparation for the contribution to the F/A-18 upgrade, we decided to investigate the use of formal specification techniques to model the MC system requirement. A literature search on this area found that the Canadian Forces had improved the rigour of the MC requirement documentation. Gagne [8], Campbell [2] and Falardeau [6] have used tables and some mathematics to document the CF-188 MC OFP task scheduling requirement. However, we are able to identify a number of improvements to this work which we believe can be addressed by the application of formal modeling techniques:

- Some mathematical terms and symbols have different meanings, even within different sections of the same document. For example, in [8] *Proc* were used to refer to a process instance in one section and to a process type (a set of processes) in the other section.
- The modeling techniques adopted were not adequate for describing concurrent interactions between OFP processes. For example, there were a number of ambiguities raised regarding the synchronisation relationships between the MC processes in the task sequence diagrams in [6].
- The description of the schedule is un-structured and repetitious. For example, the resources allocations and processes' interactions in one task rates are separately documented.

Our initial approach [3] used Object-Z [5] to model a static pre-run time scheduler based on that of Gagne [8]. However, the treatment of timing issues in this approach is cumbersome and it is not well suited for modeling the OFP process concurrent interactions and the task rate sequences that are documented by Falardeau [6]. From this experience, we realised that the state-based Object-Z notation lacks adequate mechanisms for treating real-time and concurrency. Therefore we have developed a notation called Timed Communicating Object Z (TCOZ) [18,16] which integrates Object-Z with Timed CSP [21]. In this paper, we demonstrate how MC process definitions, concurrent interactions, and task rate sequences can be effectively formalised in TCOZ.

The remainder of the paper is organised as follows. Section 2 briefly introduce the TCOZ notation. Section 3 outlines the multi-parallel task rates requirements for an MC OFP. Section 4 presents the TCOZ model of the task rates. Section 5 discusses the benefits of our approach and a comparison to CDIS project approach. Section 6 concludes the paper.

2 TCOZ features

TCOZ is essentially a blending of Object-Z and Timed CSP, for the most part preserving them as proper sub-languages of the blended notation. The essence of this blending is the identification of Object-Z operation specification schemas with terminating CSP processes. Thus operation schemas and CSP processes occupy the same syntactic category, operation schema expressions may appear wherever processes may appear in CSP and CSP process definitions may appear wherever operation definitions may appear in Object-Z. The primary specification structuring device in TCOZ is the Object-Z class mechanism.

In this section we briefly consider the aspects of TCOZ which help to bring the two notations together. A detailed introduction to TCOZ and its Timed CSP and Object-Z features may be found elsewhere [18]. The semantics of TCOZ is documented in [19]. A glossary of TCOZ notation is presented in Appendix A.

2.1 A model of time

In TCOZ, all timing information is represented as real valued measurements in *seconds*, the SI standard unit of time [14]. We believe that a mature approach to measurement and measurement standards is essential to the application of formal techniques to systems engineering problems. In order to support the use of standard units of measurement, extensions to the Z typing system suggested by Hayes and Mahony [12] are adopted. Under this convention, time quantities are represented by the type

$$\mathbb{T} == \mathbb{R} \odot \mathsf{T},$$

where $\mathbb{R}$ represents the real numbers and T is the SI symbol for dimensions of time. Time literals consist of a real number literal annotated with a symbol representing a unit of time. For example, $3\,\mathsf{ms}$ is a literal representing a period of three milliseconds, the basic time unit for the Mission Computer systems. All the arithmetic operators are extended in the obvious way to allow calculations involving units of measurement.

2.2 Deadlines and delays

In order to describe the timing requirements of operations and sequences of operations, a deadline command inspired by Hayes and Utting [13] is introduced. If OP is an operation specification (defined through any combination of CSP process primitives and Object-Z operation schemas) then OP; $\textsc{Deadline}\, t$ describes the process which has the same effect as OP, but is constrained to terminate no later than t.

The $\textsc{WaitUntil}$ operator is a dual to the deadline operator.
The process OP; $\textsc{WaitUntil}\, t$ performs OP, but will not terminate until at least time t.

2.3 Declaring channels

CSP channels are given an independent, first class role in TCOZ. This allows the communications and control topology of a network of objects to be designed orthogonally to their class structure.

In order to support the role of CSP channels, the state schema convention is extended to allow the declaration of communication channels. If c is to be used as a communication channel by any of the operations of a class, then it must be declared in the state schema to be of type **chan**. Channels are type heterogeneous and may carry communications of any type. Contrary to the conventions adopted for internal state attributes, channels are viewed as shared (global) rather than as encapsulated entities. This is an essential consequence of their role as communications interfaces *between* objects. The introduction of channels to TCOZ reduces the need to reference other classes in class definitions, thereby enhancing the modularity of system specifications. Channels play an important role in synchronising the end of a high rate process and with the start of a lower rate process in the MC model.

2.4 Sensor and Actuators

Complementary to the synchronising CSP channel mechanism, TCOZ also adopts a non-synchronising shared variable mechanism. A declaration of the form s : X **sensor** provides a channel-like interface for using the shared variable s as an input. A declaration of the form s : X **actuator** provides a local-variable-like interface for using the shared variable s as an output. Sensors and activators may appear either at the system boundary (usually describing how global analog quantities are sampled from, or generated by the digital subsystem) or else within the system (providing a convenient mechanism for describing local communications which do not require synchronisations). For a detailed discussion on TCOZ sensor and actuators see [17]. Sensors and actuators are used in the MC model to specify the real-time task clock and signals from MC1 to MC2.

2.5 Active and passive objects

Active objects have their own thread of control, while passive objects are controlled by other objects in a system. In TCOZ, an identifier MAIN (non-terminating process) is used to determine the behaviour of active objects of a given class [4]. The MAIN operation is optional in a class definition. It only appears in a class definition when the objects of that class are active objects. Classes for defining passive objects will not have the MAIN definition, but may contain CSP process constructors. If ob_1 and ob_2 are active objects of the class C, then the independent parallel composition behaviour of the two objects can be represented as $ob_1 \mid\mid\mid ob_2$, which means $ob_1.\mathrm{MAIN} \mid\mid\mid ob_2.\mathrm{MAIN}$

2.6 Complex network topologies

The syntactic structure of the CSP synchronisation operator is convenient only in the case of pipe-line like communication topologies. Expressing more complex communication topologies generally results in unacceptably complicated expressions. For example, consider that processes A and B communicate privately through the channel ab, processes A and C communicate privately through the channel ac, and processes B and C communicate privately through the channel bc. One CSP expression for such a network communication system is

$$(A[bc'/bc] \,|[\, ab, ac \,]|\, (B[ac'/ac] \,|[\, bc \,]|\, C[ab'/ab]) \setminus ab, ac, bc)[ab, ac, bc/ab', ac', bc'].$$

The hiding and renaming is necessary in order to cover cases such as C being able to communicate on channel ab.

The above expression not only suffers from syntactic clutter, but also serves to obscure the inherently simple network topology. In TCOZ, a graph-based approach is adopted to represent the network topology. For example, the network topology of A, B and C may be described by

$$\|(A \xleftrightarrow{ab} B;\; B \xleftrightarrow{bc} C;\; C \xleftrightarrow{ca} A).$$

Other forms of lax usage allow network connections with common nodes to be run together, for example

$$\|(A \xleftrightarrow{ab} B \xleftrightarrow{bc} C \xleftrightarrow{ca} A),$$

and multiple channels above the arrow, for example

$$\|(A \xleftrightarrow{ab_1, ab_2} B).$$

For a detailed discussion on TCOZ network topology see [16]. The network topology operator is used in the MC system model to describe the synchronisations between processes.

2.7 Semantics of TCOZ

A separate paper details the blended state/event process model which forms the basis for the TCOZ semantics [19]. In brief, the semantic approach is to identify the notions of operation and process by providing a process interpretation of the Z operation schema construct. TCOZ differs from many other approaches to blending Object-Z with a process algebra in that it does not identify operations with events. Instead an unspecified, fine-grained, collection of state-update events is hypothesised. Operation schemas are modelled by the collection of those sequences of update events that achieve the state change described by the schema. This means that there is no semantic difference between a Z operation schema and a CSP process. It therefore makes sense to also identify their syntactic classes.

The process model used by TCOZ consists of sets of tuples consisting of: an *initial* state; a *trace* (a sequence of time stamped events, including update-events), a *refusal* (a record what and when events are refused by the process), and a *divergence* (a record of if and when the process diverged). The trace/refusal pair is called a *failure* and the overall model the state/failures/divergences model. The state of the process at any given time is the initial state updated by all of the updates that have occurred up to that time. If an event trace terminates (that is if a $\checkmark$ event occurs), then the state at the time of termination is called the *final* state.

The process model of an operation schema consists of all initial states and update traces (terminated with a $\checkmark$) such that the initial state and the final state satisfy the relation described by the schema. If no legal final state exists for a given initial state, the operation diverges immediately. An advantage of this semantics is that it allows CSP process refinement to agree with Z operation refinement.

The process model of an active class encapsulates the object state by hiding all of the update-events. There can be no shared-variable contentions between active objects in TCOZ.

3 MC system process scheduling requirements

In this section we informally present the Mission Computer task rates scheduling requirements as described by Falardeau [6, Chapter 4]. The aircraft avionics systems comprise on-board processors, sensors and displays operating under the control of two general purpose computers, MC1 (for navigation) and MC2 (for weapon delivery), which are interconnected with each other and with all peripheral avionics equipment by means of dual-redundant serial multiplex bus channels.

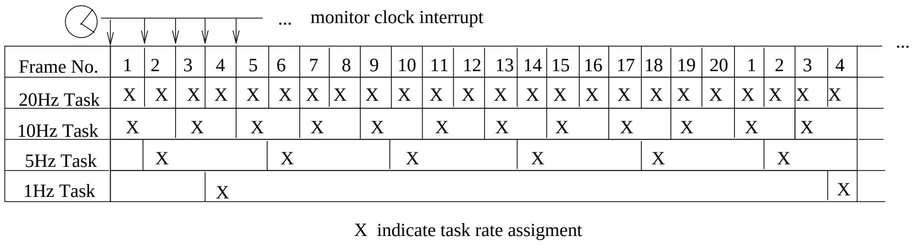

Fig. 1. Task Rate Sequences

3.1 Scheduling task rates

The various Mission Computer processes or *tasks*[1] are classified according to their periodicity. All processes have a periodicity of either 50, 100, 200, or 1000 milli-seconds, corresponding to *task rates* of 20, 10, 5, and 1 hz respectively.[2] The resources of each MC consist of a Central Processing Unit (CPU), an Input/Output Processor (IOP) and a disk. In the OFP, each process in a schedule can be allocated to one specific resource only. The scheduler uses a rate monotonic priority assignment for the task rates (assigning top priority to the 20 hz routines and lowest to the 1 hz routines). The activation of the MC1 task rates is based on a single periodic 50 ms clock interrupt (each 50 ms interval is called a *frame*) as depicted in Figure 1.

Each frame is assigned a number that is used to determine the assigned task rates. The task rate allocation pattern repeats itself at the end of 20 frames. The task rates allocation in MC2 are similar, but are activated by the "20 hz Data Available" mode command sent during the execution of the 20 hz processing task in MC1. The sequencing of each task rate frame is depicted in Figure 2. First the inputs required by the routines are read from the peripherals (via channel 1 and/or channel 2). Once the inputs are completed, the data processing routines are then executed. Once the processing phase is completed, the outputs are sent to peripherals (via channel 1 and/or channel 2).

3.2 Synchronising task rates

Since, in most frames, there are two active task rates sharing resources, there is a need to synchronise their behaviour so as to prevent resource conflicts. Only one periodic event sequence, MC1-20 hz-Task-Rate, is initiated by a fixed periodic event, generated by a 50 ms clock interrupt. All other MC1 and MC2 task

[1] The term 'task' and 'process' are used interchangingly in this paper.

[2] There are also two aperiodic tasks, data-link and bomb-release. Because the number of aperiodic processes is small and their execution time is very short, it is possible to treat aperiodic processes as pseudo-periodic processes. The data-link and bomb-release processes are therefore incorporated into the 20 hz task rate.

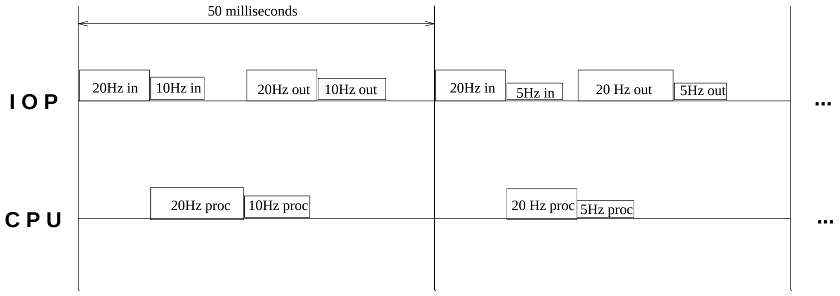

Fig. 2. Two execution frames

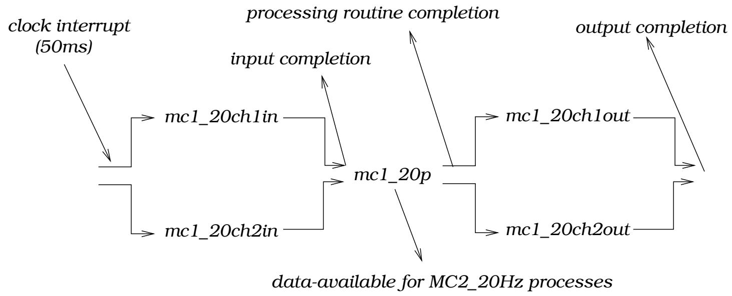

Fig. 3. Relative ordering sequence for MC1-20Hz-Task-Rate

rates are released by interprocess events and thus display jitter characteristics.[3] Figure 3 depicts the ordering of activities for the 20 hz task rate of MC1. For example, 'mc1_20ch1in' represents the utilisation of channel 1 for the collection of 20 hz input data and 'mc1_20p' represents the utilisation of CPU1 for 20 hz processing routines. Also depicted is the interprocess events which are used to release resources for the use of other processes. For example, once the 20 hz input routines are completed, an input completion event must be generated if there are other task rates allocated to the current frame.

The sequencing diagram of the 10 hz task rate is the dual of the 20 hz, using the 20 hz completion events of odd numbered frames to initiate its activities. The sequencing of the 5 hz task rate is identical to that of the 10 hz task rate, which the exception that it is active for different frame numbers. However, the 1 hz task rate is also similar, except that it requires additional input from the disk before processing routines start. The 10 hz sequence is illustrated in Figure 4 (the 1 hz sequence diagram is omitted).

[3] Periodic with jitter indicates a periodic event sequence whose initiation is not triggered at fixed intervals.

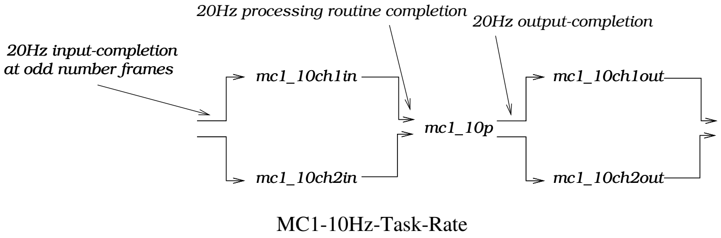

Fig. 4. Relative ordering sequence for 10Hz

The MC2 tasks schedule is similar to the MC1 except that its 20 hz task rate is triggered by the data-available signal from the MC1 20 hz task rate.

3.3 Task attributes

The definition of each task (process) includes a set of attributes ('system', 'interval', 'resource', 'worst case computation time' and 'routines'[4] which is captured in the following table.

Process	**System**	**Interval period** t	**Resource** r	**Worse case comp-time** c	**Routines**
$mc_1 20ch_1 in$	MC1	50	Chan1	3	...
$mc_1 20ch_2 in$	MC1	50	Chan2	6	...
$mc_1 20p_1$	MC1	50	CPU1	17	...
$mc_1 20ch_1 out$	MC1	50	Chan1	4	...
$mc_1 20ch_2 out$	MC1	50	Chan2	7	...
...	...	...	...	...	...
$mc_2 01 disk\_in$	MC2	1000	Disk2	1	...
...	...	...	...	...	...

Other OFP requirements are omitted in this paper because we aim to demonstrate the approach rather than give a complete detailed formal documentation. We also abstractly simplify some requirements to make the presentation of this approach more effective. For example, there are two or three routines associated with each task rate (executing sequentially), in this paper, we abstractly view those routines as a single routines.

4 Modeling the MC scheduling system

The MC system specification is developed in a bottom-up manner, beginning with models of basic (passive) objects, such as task routines, which are then

[4] The description of routines for each task is not documented in [6] as they are in very low level detail.

used to develop more complex (active) objects, such as task rate sequences. Then the individual mission computer schedulers and the overall scheduler for the two MCs are modeled as composites of their component active objects which interact through their channel/sensor/actuator interfaces.

Before beginning the model, we introduce some generic definitions.

4.1 Generic definitions

Firstly, there are two Mission Computers.

$$MC \mathrel{\widehat{=}} MC_1 \mid MC_2$$

The resources of the Mission Computers are specified as

$$Resource \mathrel{\widehat{=}} CPU_1 \mid CPU_2 \mid Chan_1 \mid Chan_2 \mid Disk_1 \mid Disk_2.$$

Different task rate routines start at different frame numbers (20 frames). Let

$$FN == 1 \mathinner{\ldotp\ldotp} 20$$

represent these frames. The 20 hz routines execute at every frame, while the 10 hz routines execute at every odd frame (see Figure 1 in Section 3). Let

$$10FN == \{n : FN \mid n \bmod 2 = 1\}$$

represent the set of odd number frames. The 5 hz routines execute at the 2nd, 6th, 10th, 14th and 18th frame. This set is modelled by:

$$5FN == \{n : FN \mid (n + 2) \bmod 4 = 0\}$$

The 1 hz routines only execute at the 4th frame.

$$1FN == \{4\}$$

In all other frames, the 20 hz routines execute alone.

$$Alone == FN \setminus (10FN \cup 5FN \cup 1FN)$$

4.2 Basic process definitions and classifications

Every MC process (task) has the following attributes.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Proc}} \\
\hline
\begin{array}{|ll}
\hline
sys : MC & \\
t : \mathbb{T} & \text{[period]} \\
c : \mathbb{T} & \text{[(worst case) computation time]} \\
r : Resource & \\
\ldots & \text{[local states omitted]} \\
\hline
0 < c \leq t & \\
\hline
\end{array} \\
\begin{array}{|ll}
\multicolumn{2}{l}{\textit{Operation}} \\
\hline
\ldots & \text{[details omitted]} \\
\hline
\end{array} \\
Act \mathrel{\widehat{=}} Operation;\ \textsc{Deadline}\ c \\
\hline
\end{array}
$$

The two major process classifications are based on the associated MC system and the associated task rate respectively.

$$
\begin{array}{|l}
\multicolumn{1}{l}{MC1_p} \\
\hline
Proc \\
\hline
sys = MC_1 \\
\hline
\end{array}
\qquad
\begin{array}{|l}
\multicolumn{1}{l}{MC2_p} \\
\hline
Proc \\
\hline
sys = MC_2 \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{20Hz_p} \\
\hline
Proc \\
\hline
t = 50 \\
\hline
\end{array}
\quad
\begin{array}{|l}
\multicolumn{1}{l}{10Hz_p} \\
\hline
Proc \\
\hline
t = 100 \\
\hline
\end{array}
\quad
\begin{array}{|l}
\multicolumn{1}{l}{5Hz_p} \\
\hline
Proc \\
\hline
t = 200 \\
\hline
\end{array}
\quad
\begin{array}{|l}
\multicolumn{1}{l}{1Hz_p} \\
\hline
Proc \\
\hline
t = 1000 \\
\hline
\end{array}
$$

These general (abstract) process definitions are used as a class library to define the individual process of the MC system.

4.3 20Hz processes and 20Hz-Task-Rate sequence

We make use of (multiple) inheritance, to define the MC1-20 hz input/output/processing routines.

$$
\begin{array}{|l}
\multicolumn{1}{l}{MC1\_20ch1in} \\
\hline
MC1_p, 20Hz_p \\
\hline
r = Chan_1 \land c = 3 \\
\hline
\end{array}
\qquad
\begin{array}{|l}
\multicolumn{1}{l}{MC1\_20ch2in} \\
\hline
MC1_p, 20Hz_p \\
\hline
r = Chan_2 \land c = 6 \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{MC1\_20p} \\
\hline
MC1_p, 20Hz_p \\
\hline
r = CPU_1 \\
c = 12 \\
\hline
\end{array}
\quad
\begin{array}{|l}
\multicolumn{1}{l}{MC1\_20ch1out} \\
\hline
MC1_p, 20Hz_p \\
\hline
r = Chan_1 \\
c = 4 \\
\hline
\end{array}
\quad
\begin{array}{|l}
\multicolumn{1}{l}{MC1\_20ch2out} \\
\hline
MC1_p, 20Hz_p \\
\hline
r = Chan_2 \\
c = 7 \\
\hline
\end{array}
$$

Inheritance ('is-a' relationship) promotes incremental understandability and highlights the differences. For example, based on the understanding of $MC1_p$ and $20Hz_p$, $MC1\_20ch1in$ 'is-a' MC1 process and also 'is-a' 20 hz process. The differences between these MC1-20 hz processes are captured by their individual class definitions. Inheritance also promotes reusability, e.g. the general classes $MC1_p$ and $20Hz_p$ are reused multiple times and will be further reused in the rest of the model.

Another TCOZ reuse mechanism is object composition. We describe the MC1-20 hz task rate sequence as a composition of the above MC1-20 hz process definitions.

$MC1\_20TR$

$fn : FN$	[50 ms frame number]
$ci : \mathbb{B}\,\mathbf{sensor}$	[MC1 clock interrupt]
$data\_avail : \mathbb{N}\,\mathbf{actuator}$	[to trigger the MC2 processes]
$20ic : \mathbf{chan}$	[20 hz Input completion event]
$20pc : \mathbf{chan}$	[20 hz processing completion event]
$20oc : \mathbf{chan}$	[20 hz Output completion event]
$fc : \mathbf{chan}$	[frame completion event]
$ch1in : MC1\_20ch1in$	[channel 1 input process]
$ch2in : MC1\_20ch2in$	[channel 2 input process]
$p : MC1\_20p$	[task rate processing routines]
$ch1out : MC1\_20ch1out$	[channel 1 output process]
$ch2out : MC1\_20ch2out$	[channel 2 output process]

INIT

$$fn = 1$$
$$data\_avail = 0$$

NextFrame

$$\Delta(fn)$$
$$fn < 20 \Rightarrow fn' = fn + 1$$
$$fn = 20 \Rightarrow fn' = 1$$

$$Input \mathrel{\widehat=} ch1out.Act \mathbin{|||} ch2out.Act$$
$$Output \mathrel{\widehat=} ch1out.Act \mathbin{|||} ch2out.Act$$
$$Process \mathrel{\widehat=} (p.Act \mathbin{|||} data\_avail := fn);\ \text{DEADLINE}\ p.c$$
$$JustRun \mathrel{\widehat=} Input;\ Process;\ Output;\ NextFrame$$
$$RunInitOther \mathrel{\widehat=} Input;\ 20ic!fn \to Process;\ 20pc \to Output;\ 20oc \to fc \to NextFrame$$
$$\text{MAIN} \mathrel{\widehat=} \mu\, S \bullet ci?true \to (([fn \in Alone] \bullet JustRun) \Box ([fn \notin Alone] \bullet RunInitOther));\ \text{DEADLINE}\ 50\,\mathsf{ms};\ S$$

Note that the expression $data\_avail := fn$ (in *Process*) is a short form of the schema

$$[\Delta(data\_avail) \mid data\_avail' = fn]$$

The active class $MC1\_20TR$ formalises the task model presented in Figure 3 in straightforward manner, with process sequencing representing the corresponding process precedence arrows and CSP events representing the corresponding process synchronisations. An interesting aspect of the definition is the mixed use of channels (for inter-process communications) and sensor/actuators (for inter-computer communications) in representing these synchronisations. Channels are used to represent MC1 inter-process synchronisations because these events resolve potential resource utilisation conflicts. For example, the start of the 20 hz-output and the start of the 10 hz-processing must wait for the completion of both the 20 hz-processing and the 10 hz-input (see Figure 2). Since CSP channel events represent precise synchronisations between events, the $20pc$ is able to fulfill a dual function. It delays the 10 hz-processing routine until the 20 hz-processing routine is completed and it also delays the 20 hz-output routine until the 10 hz-input routine is completed. On the other hand, the $data\_avail$ signal between MC1-20 hz task rate and the MC2-20 hz task rate doesn't need such symmetric synchronisation because they do not share any resources. After MC1-20 task rate signals 'data-available' with the current frame number, there is no reason for MC1-20 hz to wait for MC2-20 hz to continue. This behaviour is precisely captured by the TCOZ sensor/actuator (non-synchronising) mechanism.

4.4 MC1 other task rates

MC1-10 hz input/processing/output tasks, $MC1\_10ch1in$, $MC1\_10ch2in$, $MC1\_10p$, $MC1\_10ch1out$ and $MC1\_10ch2out$ can be similarly modelled (therefore omitted).

At every odd numbered frame, the MC1-10 hz task rate sequence is triggered by the 20 hz input completion signal. It is modelled as:

$MC1\_10TR$

$$
\begin{array}{ll}
20ic, 20pc, 20oc, fc : \mathbf{chan} & \text{[20 hz completion events]} \\
ch1in : MC1\_10ch1in & \\
ch2in : MC1\_10ch2in & \\
p : MC1\_10p & \\
ch1out : MC1\_10ch1out & \\
ch2out : MC1\_10ch2out &
\end{array}
$$

$$
\begin{array}{l}
\textsc{Main} \mathrel{\widehat=} \mu\, S \bullet [fn? : 10FN] \bullet 20ic?fn \rightarrow \\
\quad (ch1in.Act \mid\mid\mid ch2in.Act); \\
\quad 20pc \rightarrow p.Act; \\
\quad 20oc \rightarrow (ch1out.Act \mid\mid\mid ch2out.Act);\ fc \rightarrow S
\end{array}
$$

At every 2nd, 6th, 10th, 14th, and 18th frame, the MC1-5 hz task rate sequence is triggered by the 20 hz input completion signal.

$MC1\_05TR$

$$\begin{array}{|l}
\boxed{\ldots \qquad\qquad\qquad\qquad\qquad\qquad \text{[details omitted]}} \\
\textsc{Main} \mathrel{\widehat=} \mu\, S \bullet [fn? : 5FN] \bullet 20ic?fn \rightarrow \ldots \qquad \text{[details omitted]}
\end{array}$$

In just the 4th frame, the MC1-1 hz task rate sequence is triggered by the 20 hz input completion signal.

$MC1\_01TR$

$$\begin{array}{|l}
\boxed{\ldots \qquad\qquad\qquad\qquad\qquad\qquad \text{[details omitted]}} \\
\textsc{Main} \mathrel{\widehat=} \mu\, S \bullet [fn? : 1FN] \bullet 20ic?fn \rightarrow \ldots \qquad \text{[details omitted]}
\end{array}$$

4.5 MC1 scheduling model

The 20 hz real-time clock interrupt is modelled as a process that sets the interrupt high for a set brief period, every 50 ms.

$ClockInterrupt$

$$\begin{array}{|l}
\boxed{ci : \mathbb{B}\,\mathbf{actuator}} \\
\textsc{Main} \mathrel{\widehat=} \mu\, C \bullet \\
\qquad ((ci := true;\ \textsc{Deadline}\ 0.5\,\mathsf{ms};\ \textsc{WaitUntil}\ 1\,\mathsf{ms}); \\
\qquad (ci := false;\ \textsc{Deadline}\ 0.5\,\mathsf{ms});\ \textsc{WaitUntil}\ 50\,\mathsf{ms});\ C
\end{array}$$

MC1 task scheduling model consists of a clock and the four task rate sequences. The interaction between the clock and the four task rate sequences can be concisely captured by the TCOZ network topology operator.

$MC1\_Sch$

$$\begin{array}{|l}
\boxed{\begin{array}{l} clock : ClockInterrupt \\ 20tr : MC1_20TR \\ 10tr : MC1_10TR \\ 05tr : MC1_05TR \\ 01tr : MC1_01TR \end{array}} \\
\textsc{Main} \mathrel{\widehat=} \|(clock \xleftrightarrow{ci} 20tr \xleftrightarrow{20ic,20pc,20oc,fc} 10tr, 05tr, 01tr)
\end{array}$$

4.6 MC2 scheduling model

The MC2-20 hz task rate sequence is similar to the MC1-20 hz task rate. The difference is that it is triggered by a 'data-available' signal from MC1 rather than the clock interrupt and there is no other MC for it to trigger.

$MC2\_20TR$

$$
\begin{array}{|l}
\hline
\begin{array}{|ll}
fn : FN & \text{[50 ms frame number]} \\
data\_avail : \mathbb{N}\ \mathbf{sensor} & \text{[data available signal from MC1]} \\
20ic, 20pc, 20oc, fc : \mathbf{chan} & \\
\ldots & \text{[details omitted]} \\
\hline
\end{array} \\
\ldots \hfill \text{[operations omitted]} \\
\textsc{Main} \mathrel{\widehat=} \mu\, S \bullet data\_avail?fn \rightarrow \\
\quad ([fn \in Alone] \bullet JustRun \mathbin{\Box} [fn \notin Alone] \bullet RunInitOther); \\
\quad \textsc{Deadline}\ 50\,\text{ms};\ S \\
\hline
\end{array}
$$

The other MC2 task rate sequences $MC2\_10TR$, $MC2\_05TR$ and $MC2\_01TR$ are essentially identical to their MC1 counterparts (therefore omitted).

The MC2 scheduling model consists solely of its four task rate sequences.

$MC2\_Sch$

$$
\begin{array}{|l}
\hline
\begin{array}{|l}
20tr : MC2\_20TR;\ 10tr : MC2\_10TR \\
05tr : MC2\_05TR;\ 01tr : MC2\_01TR \\
\hline
\end{array} \\
\textsc{Main} \mathrel{\widehat=} \|(20tr \xleftrightarrow{20ic, 20pc, 20oc, fc} 10tr, 05tr, 01tr) \\
\hline
\end{array}
$$

4.7 The overall task rates scheduling model

The overall scheduling model for the MC system is simply and elegantly specified by composing the individual MC scheduling models.

$TwoMC\_Sch$

$$
\begin{array}{|l}
\hline
\begin{array}{|l}
mc1\_sch : MC1\_Sch \\
mc2\_sch : MC2\_Sch \\
\hline
\end{array} \\
\textsc{Main} \mathrel{\widehat=} \|(mc1\_sch \xleftrightarrow{data\_avail} mc2\_sch) \\
\hline
\end{array}
$$

5 Discussion

In this section, we briefly discuss the benefits of our approach and compare our specification approach to the approach used in CDIS Project [11].

5.1 Benefits

We believe that our adoption of a formal object-oriented specification language has enabled us to present a description of the MC OFP Task Schedule which is both clearer and more succinct than the existing descriptions [8, 2, 6]. The benefits have included consistent use of terminology, a well-defined collection of synchronisation and concurrency primitives, and the ability to apply object-oriented abstraction techniques to structure and simplify the description. The effect of adopting a small but powerful model of time and concurrency in the abstract should not be underestimated. It was not a superior understanding of the MC OFP itself which has allowed our improved presentation, but rather the powerful intellectual modeling techniques provided by the Timed CSP primitives used in TCOZ.

A common form of error in the original documents was the inconsistent use of terminology. For example, Gagne [8] uses *Proc* to refer to a process (task) instance in one section and to a process type (a set of processes) in the other section. Those errors can be removed perhaps by using different terms.

More importantly, during the formalisation process, we also identified a number of ambiguities in the Falardeau model [6] regarding the synchronisation relationships between the MC processes. For example, the differences between the simple data-available signal between the MCs and the inter-process synchronisations required to avoid resource contention are not clearly documented by Falardeau's task rate sequence diagram. Many task priority interruptions were missing from the diagram. Understanding those critical inter-process synchronisations required painstaking reading of different text sections with many clarifications from our local aviation domain expert (Neale Fulton). Whence we precisely understand those critical inter-process synchronisations differences, in the TCOZ model these differences are clearly captured by using differing communication mechanisms, sensor/actuators or channels as appropriate. We also believe our approach to be complementary to the work of Falardeau, in as much as the tables and diagrams provide a valuable visualisation and comprehension aid to the formal TCOZ model. In fact, after our formalisation process, we redrew most of the task rate sequence diagrams some of which are illustrated in Section 3 of this paper.

5.2 Comparison to CDIS approach

The applications described in both our project and the CDIS project reported by Hall [11] can be classified in the aviation systems domain. We are dealing with the mission computer system embedded within the aircraft, while CDIS is concerned with air traffic management. CDIS project is a much larger commercial application of formal methods which covers the complete development cycle (although only formal specification was used, not the formal verification process). Our project is a government/industry motivated/supported project which only covers the specification of the system. In the context of specification, there are some similarities between our approach and the approach adopted in the CDIS project. For example, we both used the combination of state-based and event-based notations (we used Object-Z with Timed CSP, CDIS used VDM and CSP/CCS). In addition, we both obtained significant benefits from the use of formal specification, even in the absence of formal analysis and verification techniques.

Some drawbacks of the CDIS approach which were reported by Hall [11, page 76] are:

- "It was difficult to get an overview from the formal specification, because it was neither top-down nor hierarchical"
- "The specification were not clear about certain timing issues ..."

Our approach clearly mitigates those drawbacks. This is because not only that our approach is based on a more structured formalism (Object-Z) and a timed formalism Timed-CSP, but also that we have spent a large research effort on the integration of the two notations, TCOZ. It is fair to point out that although the CDIS project was reported in 1996 [11], the actual project started in 1989 and delivered in 1992. At that time, perhaps formal object-oriented specification was not mature enough to be considered. Even though the development of TCOZ is primarily motivated by this mission computer system application, TCOZ was also shaped by a recent research focus on the intergration of Z/Object-Z and CSP/CCS [7, 9, 23, 24, 22].

6 Conclusion

In this paper, we have presented a formal specification of the CF-188 Mission Computer task rates requirements documented by Falardeau [6]. The contributions of our approach are as follows.

- A uniform, integrated formal notation was used to specify the MC task rates model.
- The TCOZ process communications primitives precisely described concurrent interactions between MC processes.
- The model of the task rates requirements is highly structured.

The model represents a course-grained approach to the task-scheduling problem. Fine-grained issues such as task context-shifting and interrupt priorities have been abstracted, thus the main aim has been to provide an expository model to aid in the understand the main issues involved. A finer-grained model more suited to detailed formal analysis is currently under construction. Nevertheless, it is possible to do some forms of analysis using the current model. For example, the model is constructed such that the question of the feasibility of satisfying the scheduling is equivalent to the TCOZ model being free from deadlocks. Absence of deadlocks can be checked by CSP analysis tools such as FDR [20].

This work not only formalises the scheduling requirements for a particular aircraft, but more importantly, demonstrates an incremental and extendible modeling approach. Thus, this formal model can be readily reused to specify other aircraft OFP task rates scheduling requirements. We believe that, given the necessary F/A-18 specific data, the model presented in this paper can be quickly modified to provide a scheduling model for the F/A-18 MC system. We are also planning to produce a model for the proposed upgraded version of F/A-18 MC system, so that we can precisely distinguished the difference between the existing F/A-18 and the proposed-upgraded version.

Another area in need of development is the provision of support tools for the TCOZ language. TCOZ preserves in large part both the syntax and semantics of the individual notations (Object-Z and Timed CSP) and hence can potentially benefit from the large body of experience developed in the use of, and tool support for the individual notations and their parents. For example, we have long-term plans to develop a type-checker for TCOZ based on the Object-Z's type-checker and to develop model checking techniques based on existing process algebra model-checkers [20].

References

1. J. Bowen and V. Stavridou. The Industrial Take-Up of Formal Methods in Safety-Critical and Other Areas: A Perspective. In J. C. P. Woodcock and P. G. Larsen, editors, *FME'93: , LNCS.* Springer-Verlag, April 1993.
2. D.W. Campbell. Timing Analysis in Hard-Real-Time Systems with Application to the CF-188 . Master's Thesis, Royal Military College of Canada, 1991.
3. J.S. Dong, N. Fulton, L. Zucconi, and J. Colton. Formalising Process Scheduling Requirements for an Aircraft Operational Flight Program. In M. Hinchey and S. Liu, editors, *the IEEE International Conference on Formal Engineering Methods (ICFEM'97)*, pages 161–168, November 1997. IEEE Computer Society Press.
4. J.S. Dong and B. Mahony. Active Objects in TCOZ. In J. Staples, M. Hinchey, and S. Liu, editors, *the 2nd IEEE International Conference on Formal Engineering Methods (ICFEM'98)*, pages 16–25. IEEE Computer Society Press, December 1998.
5. R. Duke, G. Rose, and G. Smith. Object-Z: a Specification Language Advocated for the Description of Standards. *Computer Standards and Interfaces*, 17:511–533, 1995.
6. J.D.G. Falardeau. Schedulability Analysis in Rate Monotonic Based Systems with Application to CF-188 . Master's Thesis, Royal Military College of Canada, 1994.
7. C. Fischer. CSP-OZ: A combination of Object-Z and CSP. In H. Bowmann and J. Derrick, editors, *Formal Methods for Open Object-Based Distributed Systems (FMOODS '97)*, volume 2, pages 423–438. Chapman & Hall, 1997.
8. J.A.M. Gagne. A Pre-Run-Time Scheduling Algorithm with Application to the CF-188 Aircraft. Master's Thesis, Royal Military College of Canada, 1989.
9. A. J. Galloway and W. J. Stoddart. An operational semantics for ZCCS. In M. Hinchey and S. Liu, editors, *the IEEE International Conference on Formal Engineering Methods (ICFEM'97)*, 1997. IEEE Computer Society Press.
10. S. Gerhart, D. Craigen, and T. Ralston. Experience with Formal Methods in Critical Systems. *IEEE Software*, January 1994.
11. A. Hall. Using formal methods to develop an ATC information systems. *IEEE Software*, 12(6), March 1996.
12. I. J. Hayes and B. P. Mahony. Using units of measurement in formal specifications. *Formal Aspects of Computing*, 7(3), 1995.

13. I. J. Hayes and M. Utting. Coercing real-time refinement: A transmitter. In D. J. Duke and A. S. Evans, editors, *BCS-FACS Northern Formal Methods Workshop*, Electronic Workshops in Computing. Springer Verlag, 1997.
14. International Organization for Standardization, Geneva. *Units of measurement: handbook on international standards for units of measurement*, 1979.
15. N. Leveson, M. Heimdahl, H. Hildreth, and J. Reese. Requirements specification for process-control systems. *IEEE Trans. Software Eng.*, 20(9), September 1994.
16. B. Mahony and J.S. Dong. Network Topology and a Case Study in TCOZ. In J. Bowen, A. Fett, and M. Hinchey, editors, *The 11th International Conference of Z Users, LNCS*, pages 308–327, Berlin, September 1998. Springer-Verlag.
17. B. Mahony and J.S. Dong. Sensors and Actuators: Process Control in TCOZ. Technical Report 98-38, Mathematical and Information Sciences, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia, 1998.
18. B. P. Mahony and J.S. Dong. Blending Object-Z and Timed CSP: An introduction to TCOZ. In K. Futatsugi, R. Kemmerer, and K. Torii, editors, *The 20th International Conference on Software Engineering (ICSE'98)*, pages 95–104, Kyoto, Japan, April 1998. IEEE Computer Society Press.
19. B. P. Mahony and J.S. Dong. Overview of the semantics of TCOZ. In K. Araki, A. Galloway, and K. Taguchi, editors, *Proc. Integrated Formal Methods (IFM'99)*, York, UK, June 1999. Springer-Verlag. (accepted).
20. A. W. Roscoe. Model-checking CSP. In *A Classical Mind: Essays in Honour of C.A.R. Hoare*. Prentice-Hall, 1994.
21. S. Schneider and J. Davies. A brief history of Timed CSP. *Theoretical Computer Science*, 138, 1995.
22. A. Simpson, J. Davies, and J. Woodcock. Security management via Z and CSP. In J. Grundy, M. Schwenke, and T. Vickers, editors, *IRW/FMP'98*. Springer-Verlag, 1998.
23. G. Smith. A semantic integration of Object-Z and CSP for the specification of concurrent systems. In J. Fitzgerald, C. Jones, and P. Lucas, editors, *Proceedings of FME'97*, Graz, Austria, September 1997. Springer-Verlag.
24. K. Taguchi and K. Araki. The State-Based CCS Semantics for Concurrent Z Specification. In M. Hinchey and S. Liu, editors, *the IEEE International Conference on Formal Engineering Methods (ICFEM'97)*, pages 283–292, Hiroshima, Japan, November 1997. IEEE Computer Society Press.

A TCOZ glossary

Notation	Explanation
$c : \mathbf{chan}$	declare c to be a channel
$\perp$	divergent process
$\textsc{Stop}$	deadlocked process
$\textsc{Skip}$	terminate immediately
$\textsc{Wait}\ t$	delay termination by t
$a \rightarrow P$	communicate a then do P

continued on next page

Notation	Explanation
$a@t \rightarrow P$	communicate a at time t then do P
$[t : \mathbb{T}] \bullet a@t \rightarrow P$	record time of a event in variable t
$c.a$	communicate a on channel c
$c?a$	input a on channel c
$c!a$	output a from channel c
$[b] \bullet P$	enable P only if b
$P;\ Q$	perform P until termination, then perform Q
$P \ \Box\ Q$	perform the first enabled of P and Q
$[i : I] \bullet P$	perform P with first enabled value of i (indexed external choice)
$P \sqcap Q$	perform either of P and Q
$[i! : I];\ P$	perform P with any value of i (indexed internal choice)
$v := e$	syntactic sugar for $[\Delta v \mid v' = e]$
$P \setminus A$	hide the events A from the environment of P
$P \mathbin{\vert[} A \mathbin{]\vert} Q$	synchronise P and Q on events from A
$(\big\Vert\, p_1, \ldots, p_n \bullet \ldots;\ p_i \xleftrightarrow{A} p_j;\ \ldots)$	network topology abstraction with parameters $p_1, \ldots, p_n$ and network connections including p_i communicating with p_j on private channels from A
$P \mathbin{\vert\vert\vert} Q$	P and Q running without sychronisations
$P \rhd\{t\}\ Q$	if P does not begin by time t, perform Q instead
$P \swarrow\{t\}\ Q$	perform P until time t, then transfer control to Q
$P \bigtriangledown e \rightarrow Q$	perform P until exception e, then transfer control to Q
$P \bullet \textsc{Deadline}\ t$	behaviours of P which terminate before time t
$P \bullet \textsc{WaitUntil}\ t$	after P idle until time t