

# Capturing Periodic Concurrent Interactions of Mission Computer Tasks

Jin Song Dong  
National University of Singapore  
dongjs@comp.nus.edu.sg

Brendan P. Mahony  
DSTO, Australia  
Brendan.Mahony@dsto.defence.gov.au

Neale Fulton  
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 the task scheduling aspects of an aircraft mission computer can be formalised in TCOZ.*

## 1. Introduction

The use of formal methods on critical systems development has been growing in the last few years. In particular, there are number of successful applications of using formal specification techniques in the aviation industry. 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.

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 [7], 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 weaknesses to this work which are addressed by the application of formal methods:

- Some mathematical terms and symbols have different meanings, even within different sections of the same document. For example, Gagne [7] uses *Proc* 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. 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 resource allocations and process interactions of one task rate are documented in separate sections.

Our initial approach [3] used Object-Z [5] to model a static pre-run time scheduler based on that of Gagne [7]. However, the treatment of timing issues in this approach is cumbersome. It is not well suited for modeling the OFP's 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) [13, 10] which integrates Object-Z with Timed CSP [14]. 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 introduces the TCOZ notation. Section 3 out-

lines 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 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 [13]. The semantics of TCOZ is documented in [11].

### 2.1. Time aspects

In TCOZ, all timing information is represented as real valued measurements in *seconds*, the SI standard unit of time. In order to support the use of standard units of measurement, extensions to the Z typing system suggested by Hayes and Mahony [8] are adopted. Time literals consist of a real number literal annotated with a symbol representing a unit of time. For example, 3 ms is a literal representing a period of three milliseconds, the typical time unit for the Mission Computer systems.

In order to describe the timing requirements of operations and sequences of operations, a deadline command inspired by Hayes and Utting [9] is introduced. If *OP* is an operation specification (defined through any combination of CSP process primitives and Object-Z operation schemas) then *OP*; DEADLINE *t* describes the process which has the same effect as *OP*, but is constrained to terminate no later than *t*. The WAITUNTIL operator is a dual to the deadline operator. The process *OP*; WAITUNTIL *t* performs *OP*, but will not terminate until at least time *t*.

### 2.2. Channels, Sensors and Actuators

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.

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 [12]. Sensors and actuators are used in the MC model to specify the real-time task clock and signals from MC1 to MC2.

### 2.3. Active objects and network topologies

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*<sub>1</sub> and *ob*<sub>2</sub> are active objects of the class *C*, then the independent parallel composition behaviour of the two objects can be represented as *ob*<sub>1</sub> ||| *ob*<sub>2</sub>, which means *ob*<sub>1</sub>.MAIN ||| *ob*<sub>2</sub>.MAIN

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. In TCOZ, a graph-based approach is adopted to represent the network topology [10]. For example, consider that processes *A* and *B* communicate privately through the interface *ab*, processes *A* and *C* communicate privately through the interface *ac*, and processes *B* and *C* communicate privately through the interface *bc*. This net-

work topology of  $A$ ,  $B$  and  $C$  may be described by

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

For a detailed discussion on TCOZ network topology see [10]. The network topology operator is used in the MC system model to describe the synchronisations between processes.

### 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.

#### 3.1. Scheduling task rates

The various Mission Computer processes or *tasks*<sup>1</sup> are classified according to their periodicity. All processes have a periodicity of either 50, 100, 200, or 1000 milliseconds, corresponding to *task rates* of 20, 10, 5, and 1 hz respectively<sup>2</sup>. 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).

<sup>1</sup> The term ‘task’ and ‘process’ are used interchangeably in this paper.

<sup>2</sup> 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.

#### 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 rates are released by interprocess events and thus display jitter characteristics.<sup>3</sup> 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).

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’)<sup>4</sup> which is captured in the following table.

Process	Sys	Intv <i>t</i>	Res <i>r</i>	Ctime <i>c</i>	Rt
mc1_20ch1in	MC1	50	Chan1	3	...
mc1_20ch2in	MC1	50	Chan2	6	...
mc1_20p1	MC1	50	CPU1	17	...
mc1_20ch1out	MC1	50	Chan1	4	...
mc1_20ch2out	MC1	50	Chan2	7	...
...	...	...	...	...	...
mc2_01disk	MC2	1000	Disk2	1	...
...	...	...	...	...	...

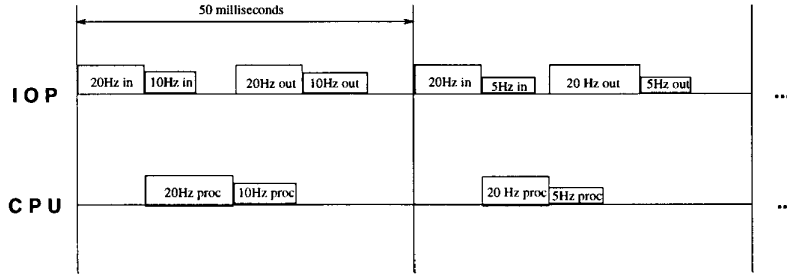
<sup>3</sup> Periodic with jitter indicates a periodic event sequence whose initiation is not triggered at fixed intervals.

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

	 ... monitor clock interrupt																							
Frame No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4
20Hz Task	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
10Hz Task	X		X		X		X		X		X		X		X		X		X		X		X	
5Hz Task		X				X				X				X			X				X			
1Hz Task				X																				X

X indicate task rate assignment

**Figure 1. Task Rate Sequences**



**Figure 2. Two execution frames**

Other OFP requirements are omitted in this paper because we aim to demonstrate the approach rather than give 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 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.

$$MC \triangleq MC_1 \mid MC_2$$

The resources of the Mission Computers are specified as

$$Resource \triangleq CPU_1 \mid CPU_2 \mid Chan_1 \mid Chan_2 \mid Disk_1 \mid Disk_2.$$

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

$$FN == 1 \dots 20$$

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

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

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

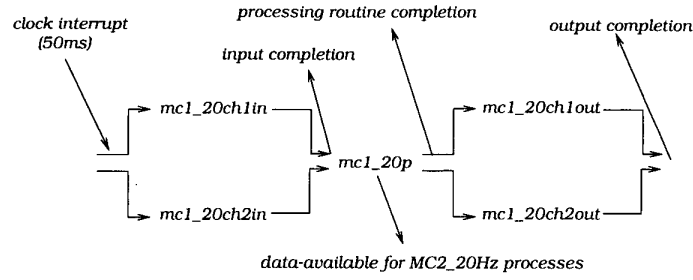
$$5FN == \{n : FN \mid (n + 2) \bmod 4 = 0\}$$

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

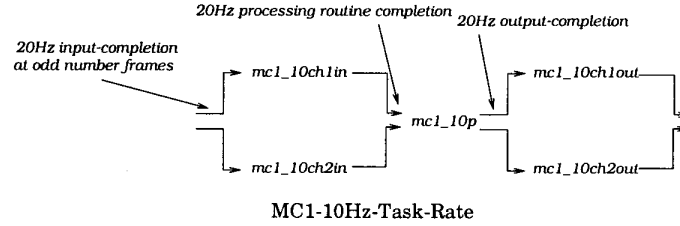
$$1FN == \{4\}$$

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

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



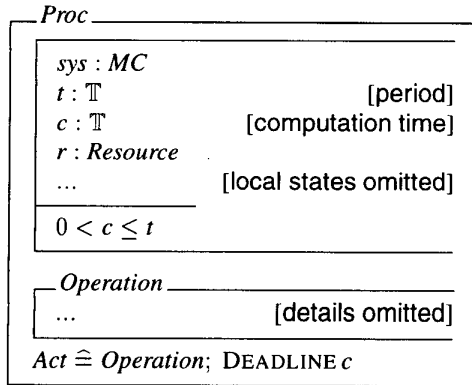
**Figure 3. Relative ordering sequence for MC1-20Hz-Task-Rate**



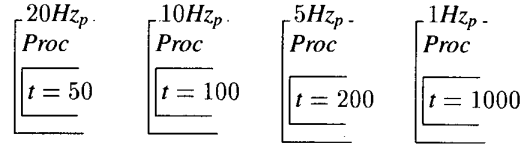
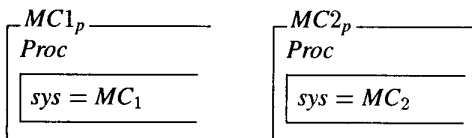
**Figure 4. Relative ordering sequence for 10Hz**

#### 4.2. Basic process definitions and classifications

Every MC process (task) has the following attributes.



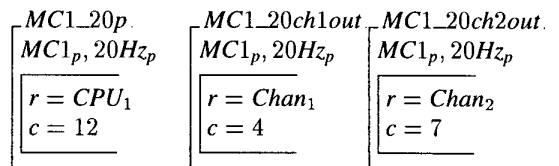
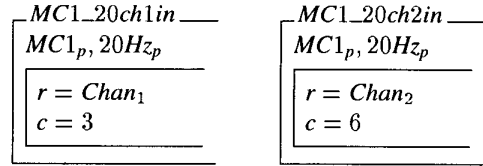
The two major process classifications are based on the associated MC system and the associated task rate respectively.



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

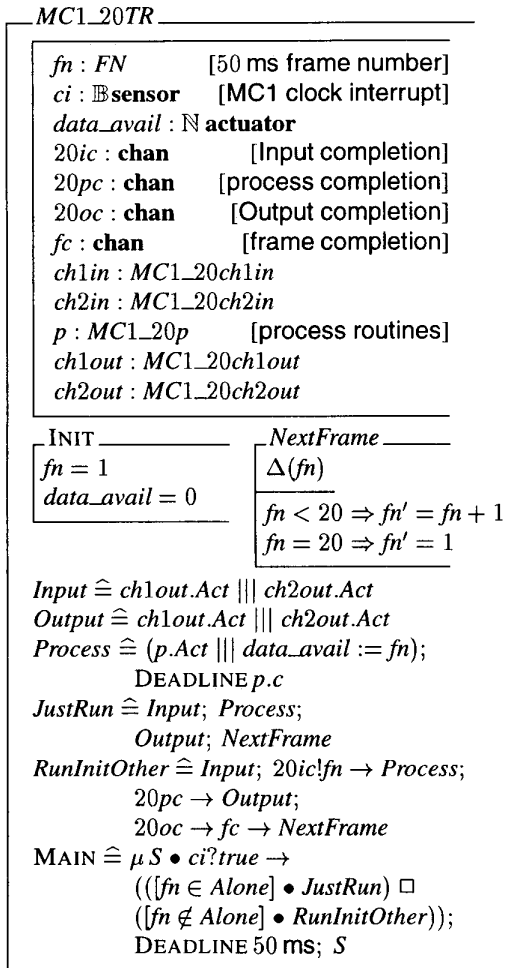
#### 4.3. 20Hz processes and 20Hz-Task-Rate sequence

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



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'  $20\text{ hz}$  process. The differences between these  $MC1$ - $20\text{ hz}$  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  $MC1$ - $20\text{ hz}$  task rate sequence as a composition of the above  $MC1$ - $20\text{ hz}$  process definitions.



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 sen-

sor/actuators (for inter-computer communications) in capturing these synchronisations.

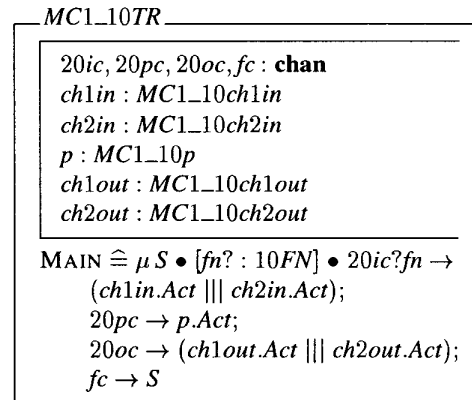
Channels are used to capture  $MC1$  inter-process synchronisations because these events resolve potential resource utilisation conflicts. For example, the start of the  $20\text{ hz}$ -output and the start of the  $10\text{ hz}$ -processing must wait for the completion of both the  $20\text{ hz}$ -processing and the  $10\text{ hz}$ -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  $10\text{ hz}$ -processing routine until the  $20\text{ hz}$ -processing routine is completed and it also delays the  $20\text{ hz}$ -output routine until the  $10\text{ hz}$ -input routine is completed.

On the other hand, the  $data\_avail$  signal between  $MC1$ - $20\text{ hz}$  task rate and the  $MC2$ - $20\text{ hz}$  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  $MC1$ - $20\text{ hz}$  to wait for  $MC2$ - $20\text{ hz}$  to continue. This behaviour is precisely captured by the TCOZ sensor/actuator (non-synchronising) mechanism.

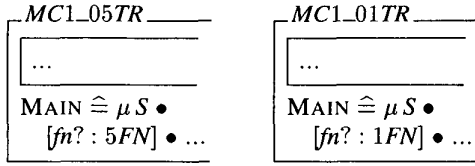
#### 4.4. $MC1$ other task rates

$MC1$ - $10\text{ hz}$  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  $MC1$ - $10\text{ hz}$  task rate sequence is triggered by the  $20\text{ hz}$  input completion signal. It is modelled as:

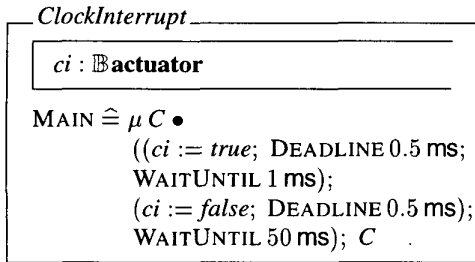


At every 2nd, 6th, 10th, 14th, and 18th frame, the  $MC1$ - $5\text{ hz}$  task rate sequence is triggered by the  $20\text{ hz}$  input completion signal. In just the 4th frame, the  $MC1$ - $1\text{ hz}$  task rate sequence is triggered by the  $20\text{ hz}$  input completion signal.

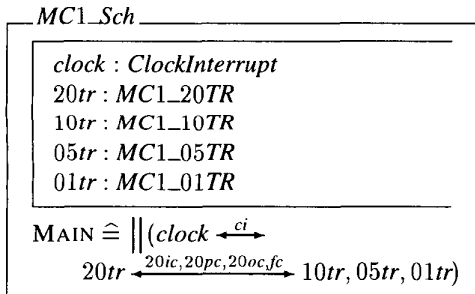


#### 4.5. MC1 scheduling model

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.

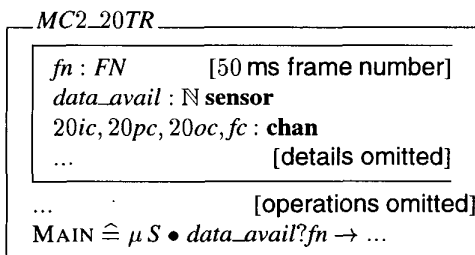


MC1 task scheduling model consists of a clock and the four task rate sequences. The interaction between the clock and the four task rate sequences can be concisely captured by the TCOZ network topology operator.



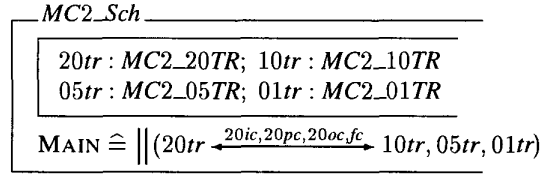
#### 4.6. MC2 scheduling model

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.



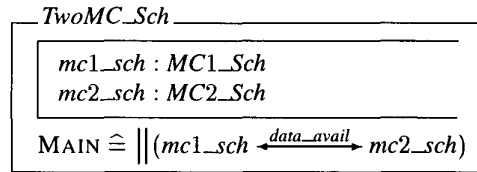
The other MC2 task rate sequences *MC2\_10TR*, *MC2\_05TR* and *MC2\_01TR* are essentially identical to their MC1 counterparts (therefore omitted).

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



#### 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.



### 5. Discussion and Conclusion

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 [7, 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 [7] 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 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.

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 [?].

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.

## References

- [1] K. Araki, A. Galloway, and K. Taguchi, editors. *IFM'99: Integrated Formal Methods, York, UK*. Springer-Verlag, June 1999.
- [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. *ICFEM'97*, pages 161–168, Hiroshima, November 1997. IEEE Press.
- [4] J.S. Dong and B. Mahony. Active Objects in TCOZ. *ICFEM'98*, pages 16–25. IEEE 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] 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.
- [8] I. J. Hayes and B. P. Mahony. Using units of measurement in formal specifications. *Formal Aspects of Computing*, 7(3), 1995.
- [9] I. J. Hayes and M. Utting. Coercing real-time refinement: A transmitter. *BCS-FACS Northern Formal Methods Workshop*, Springer Verlag, 1997.
- [10] B. Mahony and J.S. Dong. Network Topology and a Case Study in TCOZ. *The 11th International Conference of Z Users, LNCS*, pages 308–327, Berlin, Germany, September 1998. Springer-Verlag.
- [11] B. Mahony and J.S. Dong. Overview of the semantics of TCOZ. In Araki et al. [1].
- [12] B. Mahony and J.S. Dong. Sensors and Actuators in TCOZ. *FM'99: World Congress on Formal Methods*, Toulouse, France, September 1999. Springer-Verlag.
- [13] B. P. Mahony and J.S. Dong. Blending Object-Z and Timed CSP: An introduction to TCOZ. *ICSE'98*, pages 95–104, Kyoto, Japan, April 1998. IEEE Press.
- [14] S. Schneider and J. Davies. A brief history of Timed CSP. *Theoretical Computer Science*, 138, 1995.