ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

**D Duke, University of York, UK and A S Evans, University of Bradford, UK
(Eds)**

BCS-FACS Northern Formal Methods Workshop

Proceedings of the BCS-FACS Northern Formal Methods Workshop,
Ilkley, UK. 23-24 September 1996

Formal Analysis of Concurrent Real-Time Requirements Models

K. Chan and C. Fencott

Springer

Published in collaboration with the
British Computer Society

Formal Analysis of Concurrent Real-Time Requirements Models

Ken Chan Clive Fencott
Centre for Modelling and Simulation,
University of Teesside, Middlesbrough,
Cleveland, TS1 3BA, UK

Abstract

This report demonstrates the use of modal and temporal logic to analysis the functional and safety requirements of concurrent real-time systems. Our research is based on the integrated method: Hazard and Operability Studies; Ward and Mellor Essential Models; and the Temporal Calculus of Communicating Systems to model and analyse real-time control systems. In particular, we discuss the interplay between traditional hazard analysis techniques and formal methods and their associated analyses in the context of an integrated model. The approach is illustrated by a small but realistic industrial case study.

1 Introduction

At present, no single method or technique alone can be used to design safety related software for real-time applications. A possible approach is by using integrated methods which enhance the assurance of the software development process. Integrated methods increase both the comprehension of the systems under review and diversity of its construction and verification all of which in turn leads to greater levels of assurance [18].

An important area of current research is the integration of safety analysis, structured and formal methods to yield a single methodology that will be capable of capturing a holistic view of a real-time system, for the production of high integrity real-time software systems. This approach aims to make use of the strengths of one method to cover the weaknesses of the other methods. The Methods Integration Research Unit, within the Centre for Modelling and Simulation at Teesside, is concerned with research based on integrating three methodologies:

Figure 1: The Integrated Method.

- Ward and Mellor Structured Analysis for Real-time systems (WM) [28];
- Hazard and Operability Studies (HAZOPS) [5, 13, 12];
- Temporal Calculus of Communicating Systems process algebra (TCCS) [20, 21].

This report introduces Formal Analysis of the requirements models to the established integrations of WM with TCCS [15] and WM with HAZOPS [7]. This is achieved by identifying properties invariant that systems are required to possess, such as functional, safety and liveness properties, and then to prove whether or not a system does indeed possess or satisfy such properties. The safety and functional properties are consider in this paper. Safety properties

can be derived from the HAZOPS deviations and functional properties are derived from WM. These properties can be expressed in terms of modal mu-calculus logic [25, 26, 1]. This is a logic where formal properties about the nature of a system can be reasoned and proved.

The WM-TCCS integration provides translations between both models and facilitates a greater understanding of the Ward & Mellor syntax, semantics, heuristics and its limitations. A complete WM essential model provides enough information to generate a complete TCCS formal model. This integration is based on previous integration between WM and the Synchronised version of CCS (SCCS) [8, 10, 11, 4]. The change to TCCS was to resolute software development process problems these problems and solutions are documented in [15]. The WM-TCCS integration constitutes a gateway towards developing formal arguments to support the HAZOP study on WM models thus the WM-HAZOPS integration. Once the WM environmental and behavioural models are sufficiently matured appropriate HAZOP Studies are undertaken to identify hazards within the models.

Formal analysis can be automated in two ways firstly by verifying the system under review have the required functional properties and secondly by examine all the deviations, causes and consequences. Using Proof Tree Analysis we can identify the error states that give rise to the hazardous deviations that have been shown to be properties of the model.

There is existing tool supports for the integrated method. The software engineering group is currently developing a CASE tool, called ASCENT [17], to aid the design of WM models and to assist the documentation and procedures of a HAZOP study. We also have developed translator tools [15] to generate the TCCS view of the WM models from data held by ASCENT from which The Edinburgh Concurrency Workbench tool (CWB) [22, 23] is used to facility's specification, simulations and property checking of TCCS specifications. The CWB tool has been continually modified and adapted to support HAZOP study and proof tree analysis on WM-TCCS models.

The paper is organised as follows. In the rest of this section we briefly describe the three methods in context of our integrated method. In section 2 the formal framework of the integrated method and the relevant tool support is explained. Section 3 focuses on the generation of the formal functional and safety properties. The next section takes a look at proof tree analysis to identify error states. A Mixer System case study is presented in section 5 and finally conclusions and future works are discussed in section 6.

1.1 Preliminaries

1.1.1 Ward and Mellor Structured Analysis for Real-Time

Ward and Mellor Structured Analysis for Real-Time (WM) [28] is a popular software development method for real-time systems derived from the Yourdon Structured Method [29]. WM includes notations (e.g. Event-Response List, Data Flow Diagrams, State Transition Diagrams, etc.) to describe the behaviour of a system.

The original WM method [28] contains many ambiguities and inconsistencies. This paper is based on a particular formal syntactic and semantic interpretation of WM models documented in [15] thus allowing automatic translation from WM specification to an equivalent TCCS specification. The WM to TCCS translation is based on functional semantic specified in Z notation [24] and we believe our interpretation of Ward & Mellor is very close to the method intentions.

1.1.2 Hazard and Operability Studies

Hazard and Operability Studies (HAZOPS) [5] is a hazard analysis technique for identifying hazards that may occur in a system. HAZOPS is a formalised team based brainstorming activity that systematically reviews a system representation and its operating procedures to identify potential safety related hazards. The hazards are identified by examining the deviations from the intention of the system as represented by the model under review. A formal procedure is employed to search for the hazards in design models, element by element for every imaginable deviation from its normal operation using a list of guidewords and property words. These words are to invoke the designers to examine all possible abnormalities thus prompt discussion about potential causes and consequences and recommend appropriate solution to avoid the hazardous situation. The property words (i.e. Response and Event) are related to the classification of the intent under interrogation and directly influenced by the interpretation. Guide words (i.e. No, Part of, Other than, As well as, early, late, and other than) are associated with the failure mode (such as Service Omission, Service Commission, etc.) of the item under investigation and property words used to reflect its intent.

The integration between WM and HAZOPS is established and is an ongoing research to devise efficient HAZOP studies' [7, 13, 12] on WM specifications. WM HAZOP studies are performed for both the WM Environmental and Behavioural models. HAZOPing can be carried out in two views or both, a formal and informal perspectives, of the

WM Environmental and Behavioural models. A formal perspective is done in the accordance with formal semantics of WM essential model and the informal perspective of essential model is to view the components of model as unreliable. The environmental model is put to use for the informal perspective HAZOP study and the behavioural model is used for the formal perspective HAZOP study.

1.1.3 Temporal Calculus of Communicating Systems

The Temporal Calculus of Communicating Systems (TCCS) [20, 21] is an extension of Milner's CCS [19] process algebra that allows processes communicate and synchronise in a temporal concurrent system. The extension to the basic CCS language allows time to be modelled explicitly to express atomic actions (an event with its associated response) within the context of WM essential modelling semantics.

In TCCS, actions and time delays are disjointed that is actions describes the functional aspect of the process and the other describes its temporal aspect. Actions take zero duration so attaching a duration directly onto actions, requiring a process to take some amount of time in stabilising into a new state. The passing of one unit of time indicates one execution step (atomic action) in terms of WM essential modelling semantics.

2 The Integrated Process Model

In this section we briefly outline our approach with a proposed formal framework that incorporates all three methods HAZOP Study, WM and TCCS. Figure 2 describes the overall process model.

Figure 2: Overview of the process model.

This process has the following activities:

- Translating WM into TCCS based on the formal semantics
- Formalising functional requirements from WM Event-Response list
- Formalising safety requirements from
 1. either translating identified HAZOPS hazardous deviations
 2. or mechanically generating deviations for each event-response thread
- Verifying the TCCS model
 1. Functional requirements
 2. Safety requirements

When commissioned the development of the WM Environmental model will be initiated, this activity is described in [14]. Once the Environmental model is thought to be sufficiently mature a HAZOP study is undertaken. The resulting of the hazard study then provides emerging requirements that will subsequently influence modification of the model under construction. Normally, once these modifications have been adopted, satisfactory state of the Environmental model and development of Behavioural model will begin. Exceptionally, if the modifications are judged to be substantial, we may initiate a subsequent Environment HAZOP and running parallel the activity of Formal Analysis of Behaviour model can be conducted.

The formal analysis contains several sub-tasks: the translation of WM Behavioural model to TCCS; the formulation of the functional and safety properties; and the verification of the formal model. The complete process model is described in [3].

The process of translating a WM Behavioural model into TCCS is based on semantic function specified in Z notation [24] this is documented in [15]. This process is well understood and can be automatically translated using the ASCENT and WM-TCCS translator support tools.

The functional requirements are formalised manually using information from the WM Event-Response list and the functional temporal/modal logic templates described in section 3. Automating this formulation process is left for future tool development. The safety requirements are formalised by either: manually transforming the hazardous deviation, identified by the HAZOPS team, using temporal/modal logic templates (similar to the functional requirement's formalisation); or automatically generating all the formalised safety properties exhaustively based on the templates explained in the next section. The CWB tool has been modified to mechanically generate these safety requirements.

The WM-TCCS translation and the formulation of functional and safety requirements activities may run in parallel. Once the TCCS model is developed the functionality of the system can be validated by using the CWB tool to animate and verify the model has the required functional requirements. Eventually the animation of the TCCS model will be deemed to have achieved all it can under constraints of time, etc. and the formal verification of the safety requirements can be conducted. Again CWB is used to show whether or not the model does indeed satisfy the safety requirements. If the model does not satisfy the safety requirements further investigation into the design fault is conducted using Proof Tree analysis, this is explained in section 4.

Proof trees are generated while verifying requirement properties and by examining them the faulty state(s) and the action's traces leading to these faulty state(s) can be identified. Ultimately, with future research, the faulty state(s) can be traced back to the WM models identifying data and/or control transforms at fault.

There are tool supports for this process model. The WM models can be specified using the ASCENT CASE tool [17]. This tool allows the software engineer to design a software system graphically based on the WM method. There are also consistency and completeness checks on the models. An additional WM-TCCS translator tool [15] can be adapted to ASCENT to read data on the WM specification and translate it into a TCCS specification. An HAZOPS support tool is also embedded within the ASCENT CASE tool to assist the documentation and procedures of a WM HAZOP study. ASCENT is used for: structuring the HAZOPS team to examine each deviation from the intention of the system under review; allowing the documentation of potential causes and consequences and recommend solutions; and does consistency and completeness checks between the WM specification and the HAZOPS documentation. The CWB tool [22, 23] is used to facility's specification, simulations and property checking of TCCS specifications. The tool is employed to conduct formal analysis on the TCCS specification, generated from the WM-TCCS translator, to verify the functional and safety requirements. The tool has been continually modified and adapted to include support for WM HAZOP study and proof tree analysis. The tool current state can conduct proof tree analysis on functional and safety properties to determine faulty states if they exist; determine action's traces to the existing faulty states; and generate safety requirements. These safety requirements are automatically generated by providing information such as the input and output data or event flows of the WM context diagram, the temporal ordering event-response actions, and each event-response action from the WM Event-Response List. The current state of tool supports for HAZOPS is limited and is an ongoing development to improve the efficiency of a WM HAZOP Study.

3 Formulation of Properties

Properties of systems can be produced during the system analysis phase of system development. There are desirable or necessary properties of interest to a system designer such as safety, functionality, liveness, security, etc. The behaviour patterns of such system must be correct to prevent loss of life, damage to the environment, substantial financial loss, etc.

Using modal and temporal logic which can express properties of TCCS specifications - we can express functional and safety properties that may become false "always" and "sometime" in the future of the system. In the circumstances where these properties do become false, which indicates an error state may be reached, the faults in the design can be identified by using the proof tree analysis [2].

Modal mu-calculus logic [25] is used for expressing properties for the analysing of CCS specifications. These logic's are the propositional modal logic to express the concepts such as the possibility and necessity of performing particular actions.

The behaviour of TCCS can be expressed with modal mu-calculus logic properties, a slight extension of Henessey-Milner logic (HML) [26]. This is a logic where formal properties about the nature of a system can be reasoned and proved. The notion of 'necessity' and 'possibility' can be defined in terms of modal logic and the notion of 'always' and 'sometimes' can be defined in terms of mu-calculus. The combination of these two notions a complex property may be constructed providing a powerful tool for proving safety critical systems. These functional properties can be generated from the WM Event-Response list and the safety properties can deduce from the HAZOPS findings and/or mechanically generated as described in the following sections. Note the logic defined next is to be applied to TCCS specification interpretation of the WM Essential models.

3.1 Modal and Mu-calculus Logic

Propositional logic by itself can not express statements that just happen to be true now, and always will be true, and those which just happened to be the true at present. With the aid of modal logic, which seek to capture the different notions of truth, such as truth by necessity and truth by possibility, by way of additional operators. Modal logics are interpreted on labelled transition systems (LTS), $(S, T, \{\xrightarrow{\alpha} \mid \alpha \in T\})$ where S is a set of states, T is a set of actions, and $\{\xrightarrow{\alpha} \mid \alpha \in T\}$ is the set of relations for the transitions between states. Let K range over subset of an action set T , $K \in T$, then the following syntax definition represents HML, a particular modal logic:

$$A ::= tt \mid \neg A \mid A \wedge B \mid [K]A$$

where the formula may be either tt is a true constant, $\neg A$ a negated formula, $A \wedge B$ a conjunction, or $[K]A$ a modalised formula, necessity, which means A holds after every action in K . Assuming a fixed LTS we can show whether or not an agent E satisfies a particular property A we write $E \models A$ or $E \not\models A$ respectively. Refer to [25] for the definition of the satisfaction relation $\models$.

An another LTS can be defined so internal actions (silence actions symbolised as τ) are not observed in TCCS process. This is useful for expressing WM-TCCS functional and safety properties. This is done by replacing the transition relation $\rightarrow$ with $\Rightarrow$ in the LTS to represent observable action transitions, that is $(S, T, \{\xRightarrow{\alpha} \mid \alpha \in \mathcal{O}\})$ where $\mathcal{O} = (T - \{\tau\}) \cup \{\mathcal{E}\}$. The $\mathcal{O}$ is a universal set of observable actions and $\mathcal{E}$ meaning zero or more τ . So the relation $\xRightarrow{\alpha}$ permits zero or more silent actions and the relation $\xRightarrow{a}$ meaning silent activity before and after a happens which is equivalent to $\xRightarrow{\tau} \xrightarrow{a} \xRightarrow{\tau}$. The observable action set $\mathcal{O}$ is made up of observable input actions set $\mathcal{O}_i$ and observable output actions set $\mathcal{O}_o$, that is $\mathcal{O} = \mathcal{O}_i \cup \mathcal{O}_o$. The observable modalised formula $\llbracket \mathcal{K} \rrbracket A$ which means A holds after every action in $\mathcal{K}$ when $\mathcal{K} \subseteq \mathcal{O}$.

An useful modal operator, $\langle K \rangle^*$, for expressing inputs and outputs of WM flows to a transform, that is a notation to express linear modal combination. The semantics as follows:

$$\begin{aligned} \langle K \rangle^* P &\stackrel{\text{def}}{=} \bigvee_{a \in K} \langle a \rangle \langle K \setminus \{a\} \rangle^* P && \text{where } \#K > 1 \\ \langle a \rangle^* P &\stackrel{\text{def}}{=} \langle a \rangle P \end{aligned}$$

Similarly, the modal operator $[K]^*$ as the dual of $\langle K \rangle^*$, that is

$$\begin{aligned}
 [K]^* P &\stackrel{\text{def}}{=} \bigwedge_{a \in K} [a][K \setminus \{a\}]^* P && \text{where } \#K > 1 \\
 [a]^* P &\stackrel{\text{def}}{=} [a] P
 \end{aligned}$$

The operator $\langle\langle K \rangle\rangle^*$ and $\llbracket K \rrbracket^*$ is defined similarly. Modal logic alone is not rich enough to express properties with a temporal nature. So modal logic is extended by adding an extra operator, a fixed point operator, to cope with these temporal properties.

Within temporal logic, one can express logical operators corresponding to time-dependent concepts such as "always" and "sometimes". To express these temporal operators the fixed point operator, mu-calculus, is adapted into HML. The syntax of the HML with mu-calculus was introduced by D. Kozen [16] is as follows:-

$$A ::= Z \mid \neg A \mid A \wedge B \mid [K]A \mid \nu Z.A$$

where K ranges over sets of actions and Z ranges over variables. The only difference to the original HML syntax is the introduction of the variable set Z and the fixed point operator $\nu Z.A$. The $\nu Z.A$ expresses the property given by the maximal solution to $Z = A$: the operator νZ binding free occurrences of Z in A .

Now temporal properties can be expressed using the extended HML. To express properties more easily high order properties operators can be defined from these primitive logic operators. Other useful primitive operators and constants may be derived from the extended HML defined above, as follows:

$$\begin{aligned}
 \mu Z.A &\stackrel{\text{def}}{=} \neg \nu Z. \neg A \\
 \llbracket K \rrbracket A &\stackrel{\text{def}}{=} \nu Z. ([\tau]Z \wedge [K]\nu Y. (A \wedge [\tau]Y)) \\
 \langle\langle K \rangle\rangle A &\stackrel{\text{def}}{=} \mu Z. (\langle\tau\rangle Z \wedge \langle K \rangle \mu Y. (A \vee \langle\tau\rangle Y)) \\
 tt &\stackrel{\text{def}}{=} \nu Z. Z
 \end{aligned}$$

where $K \subseteq T$ in LTS. For a more detailed description of these modal and mu-calculus logic refer to [25, 26].

3.2 Property Requirements

Formal properties embody design decisions about the nature of the system's behaviour under development. It must be the designer's intention that the properties will be satisfied for any state that the system may evolve to. Some of the properties which are of interest to a system designer are liveness, deadlock, security, safety, etc. The safety property amounts to nothing bad ever happens while a liveness property expresses something good does eventually happen.

The notion of 'necessity' and 'possibility' can be defined with modal logic and the notion of 'always' and 'sometimes' can be defined with a branching time temporal logic [6]. Using both of these notions interesting properties can be formally expressed. Shown below are some useful temporal operators used in the formulation of functional and safety properties. For the full definitions of the operators refer to [3].

$$\begin{aligned}
 \mathbf{EF} P &\stackrel{\text{def}}{=} \mu Z. P \vee \langle - \rangle Z && \text{:weak eventually: at least one trace in which } P \text{ become possible;} \\
 P \mathbf{U}_{ss} Q &\stackrel{\text{def}}{=} \mu Z. Q \vee (P \wedge [-]Z \wedge \langle - \rangle tt) && \text{:strong strong until - in all traces } Q \text{ becomes true and } P \text{ is always true every trace until } Q \text{ becomes true} \\
 P \mathbf{U}_{sw} Q &\stackrel{\text{def}}{=} \mu Z. Q \vee (P \wedge \langle - \rangle Z) && \text{:strong weak until - there is one trace in which } Q \text{ becomes true and } P \text{ is always true in that until } Q \text{ becomes true}
 \end{aligned}$$

The formulation of WM-TCCS functional and safety properties are explained in the following sections.

3.2.1 Functional properties

Functional properties are used to verify that the system under development has specified functional requirements. These properties are derived from the WM Event-Response list (note WM event-response list reflects the functionality of the system). The construction of a formal functional property is of the generalised form:

$$EF \langle\langle \text{Event} \rangle\rangle^* \langle\langle \text{Response} \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$$

where *Event* is a set of input event stimulus(i) and *Response* is a set of output response stimulus(i). Event and response stimulus(i) are observable inputs and outputs WM flows (i.e. event, data and continuous flows) respectively. The WM semantics states there may be none or one input event flow (discrete event stimulus) in an event-response thread (i.e. an execution step) thus set *Event* must contain zero or one event flow. Notices the property uses the temporal operator *EF*, this is because we want to say that there exists this function requirement in the system under development and we are not interested if the system can deviate from this functional requirement, i.e. safety issues are not considered. The modal expression $\langle\langle 1 \rangle\rangle tt$ expresses it is possible one unit of time to passed and in terms of WM-TCCS semantics it indicates an execution step (atomic action). The expression above informally says there exists the possible actions of event's stimulus(i) followed by the possible actions of response stimulus(i) within an atomic execution step.

For example, a functional property with an event stimuli $\{e_1\}$ and two response stimulus $\{\bar{r}_1, \bar{r}_2\}$, is of form:

$$EF \langle\langle e_1 \rangle\rangle * \langle\langle \bar{r}_1, \bar{r}_2 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$$

and expanding the modal expression, it is equivalent to:

$$EF \langle\langle e_1 \rangle\rangle (\langle\langle \bar{r}_1 \rangle\rangle \langle\langle \bar{r}_2 \rangle\rangle \langle\langle 1 \rangle\rangle tt \vee \langle\langle \bar{r}_2 \rangle\rangle \langle\langle \bar{r}_1 \rangle\rangle \langle\langle 1 \rangle\rangle tt)$$

The semantics of WM Essential models says there are no ordering in the event and response stimuli thus possible ordering of stimuli must be considered, hence the modal notation $\langle\langle \rangle\rangle^*$.

This formal template allows us to mechanically the generation of formal properties and opens way for future tool development to support the automation of producing these formal functional requirements.

3.2.2 Safety Properties

Safety properties may be derived from either: formalising each deviation identified from a HAZOP Study using formulation templates in the tables 1, 2 and 3; or mechanically exhaustively generating safety requirements using formulation templates. Having identified the hazards within a system a safety property would express that the hazardous state can not be reached.

Formalising safety requirements from identified deviation from a HAZOP Study is similar to the formulation of functional requirements except formulation templates are different as explained in the following tables.

Also the formulation of safety requirements can be mechanically generated by examining all possible WM event-response threads, and the event and response stimulus(i) (i.e. external inputs and outputs flows within WM specification). For each WM event-response thread the deviations can be deduced by examining each property word (i.e. Response and Signal) with each associated guideword (i.e. No, Part Of, Other Than, As Well As, Early, Late) using the formalised templates below.

The deviation properties are expressed using temporal and modal logic to check if the deviation may become true sometimes in the future of the system. The deviations that become true indicate the system may have a potential hazardous consequence and requires further investigation. So the most ideal HAZOP study outcome is when all the deviation properties, say *D*, does not satisfies the system under review, where **SystemX** is a TCCS design specification and *D* is a formalised deviation property, that is

$$SystemX \models \neg D$$

which means that **SystemX** satisfies the property saying " the deviation cannot be performed". When the outcome is:

$$SystemX \not\models \neg D$$

which says **SystemX** does not satisfy the safety property and needs further investigation into fault(s) of the design specifications with proof tree analysis. The severity of the fault and the recommendations are left for the HAZOP team to decide.

The formalisation of deviations varies depending on the combination of property and guide words and the number of event stimulus (observable WM inputs flows to the system) and response stimulus (observable WM outputs flows to the system). Similarly to the formulation of functional properties the event and response stimuli have no ordering.

Let *Event* be a subset of observable inputs $\mathcal{O}_i$ with zero or one discrete event (WM event flows) and *Response* is a subset of observable outputs $\mathcal{O}_o$. Now the generalised formalisation can be illustrated for the property word RESPONSE in table 1. Table 2 and 3 demonstrate generalised formulation of the property word EVENT.

Guide words	Formalised Deviation	Meaning
No	$EF \langle\langle Event \rangle\rangle^* (\llbracket Response \rrbracket^* ff U_{ss} \langle\langle 1 \rangle\rangle tt)$	there exists event stimuli of set <i>Event</i> and after these events the response stimuli of set <i>Response</i> is incapable until end of the execution step
Part Of	$\bigvee EF \langle\langle Event \rangle\rangle^* \langle\langle R \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$ where $R \in (\mathcal{P} Response \setminus \{Response, \emptyset\})$	there exists event stimuli of set <i>Event</i> and after these events it is possible to have part of set <i>Response</i> response stimuli within an execution step
Other Than	$\bigvee EF \langle\langle Event \rangle\rangle^* \langle\langle R \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$ where $R \in (\mathcal{P}(\mathcal{O}_o \setminus Response) \setminus \emptyset)$	there exists event stimuli in <i>Event</i> and after it is possible to have one or more spurious responses stimuli other than in set <i>Response</i> within an execution step
As Well As	$\bigvee EF \langle\langle Event \rangle\rangle^* \langle\langle R \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$ where $R \in \{r: \mathcal{P} \mathcal{O}_o \mid r \notin (\mathcal{P} Response \cup \mathcal{P}(\mathcal{O}_o \setminus Response))\}$	there exists after event stimuli in <i>Event</i> there may be response stimuli in <i>Response</i> as well as one or more spurious response(s) within an execution step

Table 1: Deviation Formalisation for property word Response.

Guide words	Formalised Deviation	Meaning
No	$\llbracket Event \rrbracket^* ff U_{ss} \langle\langle Response \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$	there exists stimuli in <i>Event</i> incapable to act until the response stimuli of set <i>Response</i> within an execution step
Part Of	$\bigvee EF \langle\langle E \rangle\rangle^* \langle\langle Response \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$ where $E \in (\mathcal{P} Event \setminus \{Event, \emptyset\})$	there exists part of event stimuli of set <i>Event</i> and after these events it is possible to have response stimuli of set <i>Response</i> within an execution step
Other Than	$\bigvee EF \langle\langle E \rangle\rangle^* \langle\langle Response \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$ where $E \in \{e: \mathcal{P}(\mathcal{O}_i \setminus Event) \mid \#(e \setminus ContinuousEvent) \leq 1 \wedge e \neq \emptyset\}$	there exists event stimuli other than of set <i>Event</i> and after these events it is possible to have response stimulus of set <i>Response</i> within an execution step
As Well As	$\bigvee EF \langle\langle E \rangle\rangle^* \langle\langle Response \rangle\rangle^* \langle\langle 1 \rangle\rangle tt$ where $E \in \{e: \mathcal{P} \mathcal{O}_i \mid e \notin (\mathcal{P} Event \cup \mathcal{P}(\mathcal{O}_i \setminus Event)) \wedge \#(e \setminus ContinuousEvent) \leq 1\}$	there exists one or more spurious event stimuli as well as stimuli in <i>Event</i> and the response stimuli in <i>Response</i> within an execution step

Table 2: Deviation Formalisation for property word Event (event stimuli view).

The deviations for the property word EVENT is more difficult to formalise than the property word RESPONSE. There are two perspectives of the word **event**, the first view regards the event as an atomic action (i.e. a WM execution step) and second views it as an event action stimulus(i) (i.e. WM observable input flows). The two views are equality valid so both of them must be considered.

First lets consider an event as an event action stimulus(i) and assuming *Event* to be a subset of observable inputs flows $\mathcal{O}_i$ and *Response* be a subset of observable outputs flows $\mathcal{O}_o$. The deviation properties in table 2 are similar to

the formulation of the property word RESPONSE except in this table the event stimulus(i) are examined whereas table 1 considers the response stimulus(i). Another difference from response deviations is the set *Event* must have either zero or one discrete event stimulus, thus the need for a predicate logic expression to determine this condition

$$\#(e \setminus ContinuousEvent) \leq 1$$

where *ContinuousEvent* is a set of continuous input events (WM continuous data flow) that is $ContinuousEvent \subseteq \mathcal{O}_i$. Note 'Early' and 'Late' have not been specified as there are specialisations of the 'As Well As' formal deviation. The formalised deviations are in table 2.

Lets consider the event as atomic actions. This is illustrated by defining three atomic actions following in sequence (temporal ordering), the first atomic action has the functional property

$$\langle\langle Event_1 \rangle\rangle * \langle\langle Response_1 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$$

the next atomic action has the property

$$\langle\langle Event_2 \rangle\rangle * \langle\langle Response_2 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$$

and last actions has

$$\langle\langle Event_3 \rangle\rangle * \langle\langle Response_3 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$$

where $Event_{1,2,3}$ are subsets of observable inputs $\mathcal{O}_i$ with zero or one discrete event and $Response_{1,2,3}$ are observable outputs (i.e.. subset of $\mathcal{O}_o$). Assuming $Event_2$ and $Response_2$ to be the middle atomic action we can illustrating the general formulation of deviations in table 3. The guide words 'Part Of' and 'As Well As' can not be expressed because of the semantics of WM essential model but are considered when modelling individual event stimulus as described in table 1.

Guide words	Formalised Deviation	Meaning
No	$EF \langle\langle Event_1 \rangle\rangle * \langle\langle Response_1 \rangle\rangle * \langle\langle 1 \rangle\rangle ([Event_2] * \llbracket Response_2 \rrbracket * \llbracket 1 \rrbracket ff) U_{sw} \langle\langle Event_3 \rangle\rangle * \langle\langle Response_3 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$	there exists the atomic action with event stimuli of $Event_2$ and response stimuli of $Response_2$ never occurring between the first and third atomic actions
Other Than	$\bigvee EF \langle\langle Event_1 \rangle\rangle * \langle\langle Response_1 \rangle\rangle * \langle\langle 1 \rangle\rangle \langle\langle E \rangle\rangle * \langle\langle R \rangle\rangle * \langle\langle 1 \rangle\rangle \langle\langle Event_3 \rangle\rangle * \langle\langle Response_3 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$ where $R \in (\mathcal{P}(\mathcal{O}_o \setminus Response) \setminus \emptyset)$ and $E \in \{e: \mathcal{P}(\mathcal{O}_i \setminus Event_1) \mid \#(e \setminus ContinuousEvent) \leq 1\}$	there exists any atomic action(s) other than the atomic action event stimuli $event_2$ and response stimuli $Response_2$ occurring between the first and third atomic actions
Early	$EF \langle\langle Event_2 \rangle\rangle * \langle\langle Response_2 \rangle\rangle * \langle\langle 1 \rangle\rangle \langle\langle Event_1 \rangle\rangle * \langle\langle Response_1 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$	there exists an atomic action with event stimuli $Event_2$ and response stimuli $Response_2$ before the atomic action with the event $Event_1$ and response $Response_1$ stimuli
Late	$EF \langle\langle Event_3 \rangle\rangle * \langle\langle Response_3 \rangle\rangle * \langle\langle 1 \rangle\rangle \langle\langle Event_2 \rangle\rangle * \langle\langle Response_2 \rangle\rangle * \langle\langle 1 \rangle\rangle tt$	there exists an atomic action with event $Event_3$ and response $Response_3$ stimuli before the atomic action with event $Event_1$ and response $Response_1$ stimuli

Table 3: Deviation Formalisation for property word Event (atomic action view).

In conducting a HAZOP study some of the tedious work involved in identifying deviation for each guide word and for each property word and deriving safety requirements may be done automatically with the CWB tool. Only the creative parts of the HAZOP study will be required to be done by people this is because computer cannot do this effectively.

3.2.3 Other Properties

Properties such as deadlock free and livelock free (convergence) must be satisfied for the system under development to be fully safe. A WM-TCCS specification should be able to perform an action and a useful action at any time that is the system should never deadlock and never livelock respectively.

To show that a system is deadlock free the property $\mathbf{AG}(\llbracket \varepsilon \rrbracket \langle -\varepsilon \rangle tt)$ must be satisfied and to show the systems is livelock free (convergent or divergent free) the property $\mu \mathbf{Z}. [\tau] \mathbf{Z}$ is used to verify the WM-TCCS specification is safe.

4 Formal Analysis

Formal Methods have several important roles to play in the requirement's models. First, the use of formal language TCCS to provide a semantics for the WM model, means that the intent of the model under consideration will be clear and unambiguous. A further benefit is the ability to use the underlying formal model to prove whether or not a particular property of the model under inspection. Proof trees are generated while checking the required properties, that are the formalised functional and safety properties, against the WM-TCCS design specification. The proof tree can be examined, when a requirement property is not satisfied, to identify faulty states.

4.1 Proof Tree Analysis

Proof trees are based on a set of tableau rules [27]. When verifying the properties within a system there can be two types of outcomes, the property is either satisfied or not satisfied, that is $E \models P$ or $E \not\models P$ where E is the TCCS agent expression and P is the formal property. The former outcome is not so important because it just means the system satisfies the requirement specification, but the latter is of interest because it means the system has a fault in the design and needs further investigation to find the cause of the fault. This tableau proof system can be automated for finite transition systems.

A tree like structure is produced, meaning that proofs are conducted in top-down fashion, when checking whether or not an TCCS agent/state satisfies a particular property. Each step of the proof a state is checked against each part of the property until all the states is examined and if the outcome does not satisfy the property the concerning failed state(s) can be deduced.

We wish to demonstrate the correctness of the assertion that the simple Vending Machine System, $V \stackrel{\text{def}}{=} c.(l.\overline{ol}.V + b.\overline{ob}).V$, satisfies the property that after inserting a coin c both a little l and a big b chocolate bar can be selected and appropriate tray, the little tray ol or big tray ob , would open for collection. We wish to demonstrate the correctness of the assertion that $V \models \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ol} \rangle tt \wedge \langle b \rangle \langle \overline{ob} \rangle tt)$:

$$\begin{array}{c}
 \frac{V \vdash \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ol} \rangle tt \wedge \langle b \rangle \langle \overline{ob} \rangle tt)}{\frac{V \vdash \langle c \rangle tt}{l.\overline{ol}.V + b.\overline{ob}.V \vdash tt} \quad \frac{V \vdash [c](\langle l \rangle \langle \overline{ol} \rangle tt \wedge \langle b \rangle \langle \overline{ob} \rangle tt)}{l.\overline{ol}.V + b.\overline{ob}.V \vdash \langle l \rangle \langle \overline{ol} \rangle tt \wedge \langle b \rangle \langle \overline{ob} \rangle tt}} \\
 \frac{l.\overline{ol}.V + b.\overline{ob}.V \vdash \langle l \rangle \langle \overline{ol} \rangle tt}{\overline{ol}.V \vdash \langle \overline{ol} \rangle tt} \quad \frac{l.\overline{ol}.V + b.\overline{ob}.V \vdash \langle b \rangle \langle \overline{ob} \rangle tt}{\overline{ob}.V \vdash \langle \overline{ob} \rangle tt} \\
 \frac{\overline{ol}.V \vdash \langle \overline{ol} \rangle tt}{V \vdash tt} \quad \frac{\overline{ob}.V \vdash \langle \overline{ob} \rangle tt}{V \vdash tt}
 \end{array}$$

All leaves of the proof tree are of the form $F \vdash tt$ and so we have proved the correctness of the sequent $V \vdash \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ol} \rangle tt \wedge \langle b \rangle \langle \overline{ob} \rangle tt)$ at the root of the tree, that is $V \models \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ol} \rangle tt \wedge \langle b \rangle \langle \overline{ob} \rangle tt)$. We will now show that the vending machine does not possess the property, that when a coin is entered and a little bar is selected the big tray opens, $\langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ob} \rangle tt)$, as follows:

$$\begin{array}{c}
 \frac{V \vdash \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ob} \rangle tt)}{\frac{V \vdash \langle c \rangle tt}{l.\overline{ol}.V + b.\overline{ob}.V \vdash tt} \quad \frac{V \vdash [c](\langle l \rangle \langle \overline{ob} \rangle tt)}{l.\overline{ol}.V + b.\overline{ob}.V \vdash \langle l \rangle \langle \overline{ob} \rangle tt}} \\
 \frac{l.\overline{ol}.V + b.\overline{ob}.V \vdash \langle l \rangle \langle \overline{ob} \rangle tt}{\overline{ol}.V \vdash \langle \overline{ob} \rangle tt} \\
 \frac{\overline{ol}.V \vdash \langle \overline{ob} \rangle tt}{\overline{ol}.V \vdash ff}
 \end{array}$$

The unsuccessful leaf is $\overline{ol}.V \vdash ff$ which does not meets the requirements laid down for a successful leaf node and we conclude that $V \not\models \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ob} \rangle tt)$.

4.2 Fault Modes

The analysis of proof trees can be base-on a chain of causality fault to catastrophe [9]. It is possible to distinguish the interesting stages, in an unsuccessful proof tree, leading to a hazardous system that can be roughly characterised as follows:

- Fault:** A fault occur at the beginning of the error TCCS state, that is at the proof tree stage when the system commit itself from a norm state to an error state.
- Failure:** At this stage it indicates the state(s) where in TCCS specification is disfunctional. This is the end result of the proof tree, that is the leaf nodes of the proof tree

The fault and failure are the events are error states. An accident event cannot be defined by the proof tree because it is to do with the effects of the system's environment, e.g. software indirectly cause deaths.

A proof tree shows the stages leading to a hazard can be identified and hence proof tree analysis can be performed to locate the faults, i.e. the causes of the fault, within the TCCS design specifications. Consider for example the failed proof tree above, $V \not\models \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ob} \rangle tt)$, the fault modes be:

- **Fault** $V \vdash \langle c \rangle tt \wedge [c](\langle l \rangle \langle \overline{ob} \rangle tt)$
- **Failure** $\overline{ol}.V \vdash ff$

5 The Mixer System Case Study

The Mixer System has been created to provide a small yet effective illustration of the integrated method developed during this research [13, 12]. This case study has been applied on a previous paper [4] using the SCCS instead of TCCS in the integrated method. The case study is currently being investigated for the integrated method with TCCS the partial results are illustrated in this section. The requirements for the Mixer are as follows:

"A liquid mixing system mixes three liquids, A, B and Bulking agent. When the operator presses a start button, liquid A are added to a mixing vat, once 10 litres have been input, the heater is switched on and 2 litres of liquid B are added. At this point the mixer motor is turned on, then a further 3 litres of liquid B are added. When all of liquid B has been mixed in, the bulking agent is added half a litre at a time until the required 1.0 specific gravity value is reached. The heating system is then switch off and a message is sent to the operator. The operator then requests the system to turn off the mixer motor and empty the vat.

Environment release of liquid A, B and the bulking agent in isolation do not constitute a serious hazard. However, the intermediate product, an A & B Mixture does pose a serious threat. If released, it is to be considered a major hazard. Once the bulking agent is added the resulting A/B/Bulking Agent Mixture is safe."

The partial environment model used is composed of an Event-Response list and Context Diagram, the event/response is as follows:

No.	Event Name	Response (signals to be sent)
1.	operator start manufacturing	close vat and open liquid A
2.	10 litres of liquid A added	close liquid A, open liquid B and heat on
3.	2 litres of liquid B added	mixer on
4.	a further 3 litres of liquid B added	close liquid B and add bulking agent
5.	$\frac{1}{2}$ litres of agent added and specific gravity not reached yet	add $\frac{1}{2}$ litres of bulking agent
6.	$\frac{1}{2}$ litres of agent added and specific gravity reached	inform operator batch is complete and heater off
7.	operator empty vat	mixer off and open vat

Table 4: Event Response List.

Figure 3: The Mixer system Behavioural Model (top level).

Figure 3 represent a partial WM Behavioural Model of these requirements. This WM model is translated to an equivalent TCCS model (labelled *Mixer*) using ASCENT and WM-TCCS translator tools. The TCCS specification is too large to be included in this paper, refer to [3] for the full Mixer System TCCS specification.

The HAZOP study seeks to identify the hazardous deviations within the Behavioural model. There are seven event-response threads in all, let's contemplate the partial results of study on event number 4 that is the event "further 3 litres of liquid B added" with the response "close liquid B and add bulking agent". The HAZOP study has identified the deviations for the property word EVENT is shown in table 5.

Response Guideword	Deviation
No	further 3 litres of liquid B never sent
Other Than	further 3 litres of liquid B not sent, however other signals are sent
Early	Final liquid B level perceived before the actual level is reached
Late	Final liquid level perceived after the actual level is reached

Table 5: HAZOP study on the behavioural model for property word event.

Once the functional and safety properties are formulated, as explain in section 3, the verification of the system under development can be conducted using the CWB tool. The results are shown in tables 6, 7 & 8.

No.	Functional Expression
1.	$Mixer \models EF \langle\langle start \rangle\rangle * \langle\langle \overline{closeVatSump}, \overline{openLiquidA} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$
2.	$Mixer \models EF \langle\langle VolumeOfContent_{10} \rangle\rangle * \langle\langle \overline{closeLiquidA}, \overline{OpenLiquidB}, \overline{heaterOn} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$
3.	$Mixer \models EF \langle\langle VolumeOfContent_{12} \rangle\rangle * \langle\langle \overline{mixerOn} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$
4.	$Mixer \models EF \langle\langle VolumeOfContent_{15} \rangle\rangle * \langle\langle \overline{closeLiquidB}, \overline{addAgent} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$
5.	$Mixer \models EF \bigwedge_{x \in \{r:\mathbf{R} \mid r \geq 0 \wedge r \leq 1.5 \wedge r \neq 1.0\}} \langle\langle agentAdded, currentSGLevel_x \rangle\rangle * \langle\langle \overline{addAgent} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$ where
6.	$Mixer \models EF \langle\langle agentAdded, currentSGLevel_{1,0} \rangle\rangle * \langle\langle \overline{SGReached}, \overline{heaterOff} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$
7.	$Mixer \models EF \langle\langle emptyVat \rangle\rangle * \langle\langle \overline{openVatSump} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$

Table 6: Functional Requirements.

Table 6 indicates all of the Mixer's functional requirements are satisfied. Now we can verify the safety requirements, relating to identified deviation in table 5, using the templates described in section 3 are as follows:

Guide word	Safety Expression
No	$Mixer \models \neg \bigwedge EF \langle\langle VolumeOfContent_{12} \rangle\rangle * \langle\langle \overline{mixerOn} \rangle\rangle * \langle\langle 1 \rangle\rangle$ $([VolumeOfContent_{15}] * [\overline{closeLiquidB}, \overline{addAgent}] * [1] ff \quad U_{sw}$ $\langle\langle agentAdded, currentSGLevel_x \rangle\rangle * \langle\langle \overline{addAgent} \rangle\rangle * \langle\langle 1 \rangle\rangle tt)$ where $x \in \{r:\mathbf{R} \mid r \geq 0 \wedge r \leq 1.5 \wedge r \neq 1.0\}$
Early	$Mixer \not\models \neg EF \langle\langle VolumeOfContent_{15} \rangle\rangle * \langle\langle \overline{closeLiquidB}, \overline{addAgent} \rangle\rangle * \langle\langle 1 \rangle\rangle$ $\langle\langle VolumeOfContent_{12} \rangle\rangle * \langle\langle \overline{mixerOn} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$
Late	$Mixer \models \neg EF \bigwedge \langle\langle agentAdded, currentSGLevel_x \rangle\rangle * \langle\langle \overline{addAgent} \rangle\rangle * \langle\langle 1 \rangle\rangle$ $\langle\langle VolumeOfContent_{12} \rangle\rangle * \langle\langle \overline{mixerOn} \rangle\rangle * \langle\langle 1 \rangle\rangle tt$ where $x \in \{r:\mathbf{R} \mid r \geq 0 \wedge r \leq 1.5 \wedge r \neq 1.0\}$

Table 7: Safety Requirements for Property Word Event (atomic action view).

The safety requirement check for the guide word 'Other Than' has been deliberately left out because the expression is too large and complex, refer to [3] for a fuller account of this case study. The second safety requirement of table 7

does not satisfy the safety property because the Mixer System perceives the final liquid B level before the actual level is reached placing the control transform "Control Valves" into a deadlocked consequently the heater is not turn off and the mixer is never turn on. Using proof tree analysis we can determine the faulty states in the systems. This can be demonstrated by animating the system using CWB tool. The rest of the safety requirements are satisfied. Additionally the system must verify for deadlock free and livelock free, that is:

Property	Formal Expression
Deadlock Free	$Mixer \models AG(\llbracket \varepsilon \rrbracket \langle \langle -\varepsilon \rangle \rangle tt)$
Livelock Free	$Mixer \models \nu Z. [\tau] Z$

Table 8: Deadlock and livelock Free Requirements.

There is only one failed system requirement, that is:

$$Mixer \not\models \neg EF \langle \langle VolumeOfContent_{15} \rangle \rangle * \langle \langle \overline{closeLiquidB}, \overline{addAgent} \rangle \rangle * \langle \langle 1 \rangle \rangle n \\ \langle \langle VolumeOfContent_{12} \rangle \rangle * \langle \langle mixerOn \rangle \rangle * \langle \langle 1 \rangle \rangle tt$$

The proof tree and the fault modes generated by the CWB is far too large to be included in this report. The formal safety analysis must be completed for Mixer System to be deemed safe, i.e. HAZOP study for the rest of WM event-response threads.

6 Conclusion and Future Work

This work has demonstrated the feasibility of providing a basis for formal analysis of safety-related real-time requirement models and we believe that the outcome of this research in methods integration will plays an important part in software engineering in both education and commerce. We have successfully shown firm links using the formal framework process model among the three methods HAZOPS, WM and TCCS. This approach is potentially a powerful verification method that allows us to prove or otherwise safety or other properties of the software mathematically where long term cost benefits of identifying problems at an early stage in the design and requirement process are enormous.

Tool supports has been developed to provide a better argument of adopting the integrated method in real projects. The ASCENT CASE tool is used for aiding the specification, documentation and procedure's supervision of the WM method and HAZOP Studies. The WM-TCCS translator tool to translate WM specifications to equivalent TCCS specifications and the CWB tool assists the specification, simulations, action traces, and property requirements verification of TCCS specifications. The CWB tool has been modified and adapted to support HAZOP study and proof tree analysis on WM-TCCS models. Future work intends to integrate and modify these tools to improve the efficiency of safety-related software development to accommodate a more realistic complex system.

It has been shown using modal mu-calculus logic is powerful enough to express functional and safety requirements of WM-TCCS specifications. The property templates were introduced to ease the formulation of functional and safety properties. Functional properties are formalised using information from the WM event-response list and safety properties are derived by applying property words and guide words to express hazardous deviations thus the safety requirements are deduced. These properties are verified using proof tree analysis to determine if it lacks or has error states in WM-TCCS design faults. Proof tree analysis is in the early stage of development and needs more research to identify the stages of failure in a design error.

In this report a simple case study, the Mixer System, is used to illustrate the points of formal analysis. A more substantial case study would be needed to fully demonstrate the validity of this proposed software development. This is left for future research.

Future work aim to localise the Ward and Mellor design faults from the identified TCCS hazardous states thus enhancing the process of developing safety-related software systems.

References

- [1] J. Bradfield and C. Stirling. Verifying temporal properties of processes. In *CONCUR 90*, volume 458 of *LNCS*, pages 115–125. University of Edinburgh, Springer, 1990.
- [2] K. Chan. The investigation into proof tree analysis for concurrent systems and its practical applications. Master’s thesis, University of Teesside, U.K., 1994.
- [3] K. Chan. Formal Analysis of Requirements Models. Technical report, University of Teesside, 1996.
- [4] K. Chan, C. Fencott, and B. Hebbbron. Formal support for the safety analysis of requirement models. In *Proceedings of SAFECOMP 95*, pages 75–89, Belgirate, Italy, 1995. Springer.
- [5] CISHEC. A guide to Hazard and Operability Studies. Technical report, The Chemical Industry Safety and Health Council of the Chemical Industries Association Ltd., 1977.
- [6] E. Emmerson and J. Srinivasan. Branching time temporal logic. *Lecture Notes in Computer Science*, 354, 1989.
- [7] C. Fencott and B. Hebbbron. The application of HAZOP studies to integrated requirements models for control systems. In *Proceedings of SAFECOMP 94*, pages 83–92, Anaheim, USA, 1994.
- [8] P. C. Fencott, A. J. Galloway, M. A. Lockyer, S. J. O’Brien, and S. Pearson. Formalising the semantics of ward-mellor sa/rt essential model using a process algebra. In *FME’94*, volume 873 of *Lecture Notes in Computer Science*, pages 681–702. Springer-Verlag, 1994.
- [9] C. P. Fenelon and B. D. Hebbbron. Applying hazop to software engineering model. In *Risk Management and Critical Protective Systems*, Altrincham, UK, 1994.
- [10] A.J. Galloway. Methods Integration: A look at requirements elicitation and analysis in Ward-Melor SA/RT and Value-passing SCCS. Technical Report IRIN - 96, Rapport de Recherche, Université de Nantes, Institut de Recherche en Informatique de Nantes, 1995.
- [11] A.J. Galloway. *Integrated Formal Methods with Richer Methodological Profiles for the development of multi-perspective systems*. PhD thesis, School of Computing and Mathematics, May 1996.
- [12] B. Hebbbron. Applying HAZOP to a Ward and Mellor behavioural models. Technical Report TEES-SCM-BDH-2-95, University of Teesside, 1995.
- [13] B. Hebbbron. Applying HAZOP to a Ward and Mellor environmental models. Technical Report TEES-SCM-BDH-1-95, University of Teesside, 1995.
- [14] B. Hebbbron. Applying HAZOP to a Ward and Mellor essential models. Technical Report TEES-BDH-001, University of Teesside, 1995.
- [15] S. Hooker and A. Galloway. Improving the Formal Semantics of Ward-Mellor SA/RT Essential Models using a Process Algebra. Technical report, University of Teesside, 1995.
- [16] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
- [17] M. Lockyer and G. Griffiths. *ASCENT: Automated Strict CASE Environment at Teesside*. University of Teesside, Cleveland, England, 1995. Technical Manual.
- [18] J. A. McDermid. Issues in Developing Software for Safety Critical Systems. Technical report, University of York, England, 1990.
- [19] R. Milner. *Communication and Concurrency*. Prentice-Hall, Hemel Hempstead, 1989.
- [20] F. Moller and C. Troft. A temporal calculus of communicating systems. In *CONCUR 90*, volume 458 of *LNCS*, 1990.
- [21] F. Moller and C. Troft. An Overview of TCCS. University of Edinburgh, 1992.

- [22] Faron Moller. The Edinburgh Concurrency Workbench (version 6.1). Technical report, University of Edinburgh, 1992.
- [23] Faron Moller and Perdita Stevens. The Edinburgh Concurrency Workbench (version 7). Technical report, University of Edinburgh, 1994.
- [24] J.M. Spivey. *The Z notation : a reference manual*. Prentice-Hall, Hemel Hempstead, 1989.
- [25] C. Stirling. An introduction to modal and temporal logics for CCS. *Lecture Notes in Computer Science*, 491, 1991.
- [26] C. Stirling. Model and Temporal Logics for Processes. Technical Report ECS-LFCS-92-22, University of Edinburgh, 1995.
- [27] C. Stirling and D. J. Walker. Local model checking in the modal mu-calculus. *Lecture Notes in Computer Science*, 351:369–382, 1989.
- [28] P.T. Ward and S.J. Mellor. *Structured Development for Real-Time Systems*, volume 1,2,3. Prentice-Hall, 1985.
- [29] E. Yourdon. *Modern Structured Analysis*. Prentice-Hall, 1989.