10

Object-oriented Safety Monitor Synthesis

J. Górski[1, 2] *and B. Nowicki*[1]

[1]*Institute of Software Engineering/ITTI*
Mansfelda 4, P.O.Box 31
60-854 Poznań, Poland
tel +48 61 483406
fax +48 61 483582
e-mail: {gorski, nowicki}@efp.poznan.pl

[2]*Department of Applied Informatics*
Technical University of Gdańsk
80-954 Gdańsk, Poland

Abstract

The paper presents a systematic method of safety monitor synthesis. The method is based on the object-oriented model of a given application. It is assumed that the valid object model of an application extended with relevant safety aspects is available. The method comprises four steps: identification, reduction, implantation and tuning of a safety monitor. The identification step selects this part of the object model which constitutes a preliminary monitor specification. In the reduction step the monitor model is simplified in order to eliminate all irrelevant details. The implantation step ensures that the monitor is driven by measurable events. Finally, tuning focuses on setting proper sensitivity of the monitor. The method results in the monitor specification which, while incorporated into the actual system, can strengthen its safety guarantees. The method is presented within the context of an example application - a gas burner system.

Keywords

safety analysis, safety monitoring, object orientation, formal specifications

1 INTRODUCTION

Safety is an attribute of the whole application and should be considered within a broad context including the control system, plant and environment. Often, safety depends on factors which are beyond the direct reach of the control system (Górski and Nowicki, 1995) . During the control system design there are many (direct or indirect) assumptions the validity of which is taken as granted, e.g. while designing a road crossing control system it is usually assumed that car drivers obey road signalling lights. No control algorithm is capable to prevent an accident caused by a driver ignoring road lights in such system. Although system designers has no means to enforce that the validity of such assumption is maintained throughout system operation, they still can equip the system with some additional mechanisms which observe the environment and check whether those assumptions are continuously obeyed.

The above considerations lead to the concept of a *safety monitor* - a device which continuously observes some safety related parts of the system and verifies if their behaviour conform to the pre-defined characteristics. Whenever any discrepancy is discovered the monitor raises an alarm that in turn can trigger some emergency actions in the system. The monitor can activate other safety devices which aim at preventing the accident occurrence (e.g. initiate moving the system to a fail-safe state). The idea of safety monitor follows international regulations on safety critical systems (International Electrotechnical Commission and Redmill, 1989) where the continuos on-line supervision is recommended in order to identify hazardous situations before they become accidents.

This paper presents a systematic approach to deriving a safety monitor from an object-oriented model of the application. The key idea of the approach is that the monitor should follow the behaviour of those parts of the system where the hazard actually occurs.

Throughout the paper we use the object oriented methodology presented in (Rumbaugh, 1991). The presentation refers to a well known case study - the gas burner system. As a starting point we assume the results of object oriented safety analysis of the gas burner system presented in (Górski and Nowicki, 1996).

2 SAFETY MONITORING

The set of all states possible for a given system constitutes its *state space*. This space can be split into subdomains which differ regarding the criteria of their selection. One possible distinction is between *correct states* (the states admitted by the system functional requirements) and *incorrect states* (the states which contradict the mission requirements). Another distinction, made from the safety standpoint distinguishes *hazardous states* (states directly leading to accidents) and *safe states* (those states that are not hazardous). Let us assume that for each hazardous state H we can define the *monitoring criterion* which selects a set of safe states surrounding H, called the *danger zone* of H. It is assumed that the system can reach H only by passing through the associated danger zone.

The *safety monitor* of a given hazardous state H is a device which continuously observers actual system states and compares them against the defined monitoring criterion of H. Whenever the system visits a state belonging to the danger zone of H the monitor raises an *alarm signal*.

Putting the above idea into practice encounters the following problems:

- identification of the monitoring criterion;
- access to the system state - not all relevant system parameters are directly measurable;
- complexity of monitoring device - a complex monitor increases complexity of the overall system and may introduce new threats and decrease the overall reliability.

Monitor is an additional safety mechanism which is to be incorporated into the existing system to strengthen safety guarantees of the whole application. Therefore, some important quality features are required from such mechanism:

- **High sensitivity**. Monitor should not 'overlook' any situation which would lead to a hazardous state.
- **False alarm elimination**. Monitor should not be 'oversensitive' in the sense of generating spurious alarms when it is not necessary.
- **Early warning**. Raising an alarm, the monitor should leave enough time for a proper reaction of the system, before the hazardous state occurs.
- **Feasibility**. Monitor should be physically and technically feasible within a reasonable cost limits.
- **Independence**. Both the algorithm of monitoring and the technology of its implementation should be different from those employed in the target system, in order to avoid common mode failures.
- **Simplicity**. With a simple device it is easier to meet a high reliability level. Using a monitoring device which reliability is lower than reliability of the target system would not be a wise solution.

3 THE SYNTHESIS METHOD

The method assumes that an object model of the considered application is available. As the conventional object oriented methodologies do not address the safety problems explicitly, we require the model to be extended according to the method presented in (Górski and Nowicki, 1996). This method provides for explicit distinction of safe and unsafe (hazardous) states in the behaviours of objects. We will assume that the model has been sufficiently validated and that it adequally represents the modelled application. The adequacy means that if the model were stimulated by exactly the same inputs as the original application its behaviour would exactly follow the behaviour of the application, i.e. the model would become the "shadow" of the actual application.

According to (Górski and Nowicki, 1996) *critical objects* are these objects to attributes of which the hazard definition refers. Throughout this paper we assume that the hazard is expressed in terms of only one critical object (generally, a hazard may refer to several objects). The critical object explicitly defines unsafe states. By the virtue of our assumption of sufficient validation of the model we can assume that whenever the model of the critical object enters an unsafe state, safety is also endangered in the actual system. Consequently, the model of the critical object can be considered as the first approximation of the safety monitor. The monitor can be implemented as a separate device and implanted into the actual application in such a way that it is driven by the actual events and values generated by the application. This is not enough however, because the unsafe state would be detected by the monitor exactly at the same moment when it occurs in the application. Therefore, as the next step the monitor is modified in such a way that it is capable to predict that the application is approaching the hazardous state well before it actually happens.

The resulting safety monitor synthesis method comprises the following steps:

Step 1: Identification

This step assumes that the object-oriented model of the application developed in accordance with OMT and extended with the unsafe states in accordance with the method presented in (Górski and Nowicki, 1996) is available. The critical object for the hazard of interest is identified and the associated monitoring criterion is defined in terms of the *critical attributes* of this object. We use the OMT methodology (Rumbaugh, 1991) to support creation of the object-oriented models. According to OMT, the behaviour of each object is represented by a separate state diagram which communicate with other objects through events and shared variables. The state diagram corresponding to the critical object becomes the first approximation of the safety monitor specification.

Step 2: Reduction

This step aims at simplification of the monitor specification through removing from it all elements which are not relevant to safety. In OMT, the formalism of statecharts (Harel, 1987) is being applied for object behaviour modelling. The specification is structured and expressed in terms of concurrent and nested states. First, we transform the monitor specification to a state machine without nested states. Then, we remove from this machine all those parts which are not relevant to the monitoring function. By definition, the monitor is a passive device (it only observes the actual system state without influencing it) so all events generated by the model to be received by other objects are removed. For the same reason, we can remove all actions updating the shared variables which are read by other objects. The resulting model is just a passive object driven by its environment (by receiving events and reading shared variables).

For the sake of simplicity let us assume that the hazard definition involves only one critical attribute. For each state of the monitor model we identify the influence the state has on the critical attribute. The following situations are considered:

- the critical attribute is continuously modified while the object is in a given state (an activity assigned to this state explicitly changes the critical attribute value);
- the critical attribute is modified only when a given state is entered (an action assigned to incoming transition assigns a new value to the critical attribute);
- while being in the state, the critical attribute remains unchanged.

Next, all actions and activities which do not influence the critical attribute are removed from the monitor specification. The state names (except unsafe states) are changed to the names of actions or activities influencing the critical attribute. Then the states bearing the same name (i.e. modifying the critical attribute in the same way) are joined together. All unsafe states are joined together as well and the resulting state is called ALARM. Joining the states requires

To support the selection process we take into account the following dependencies between an event and its potential replacement:

- causality - the occurrence of the event can be inferred from the occurrence of its replacement on the basis of the cause-effect relationship. It is done in both direction: the replacement can be either the cause or the effect of the unmeasurable event;
- delay - the time period between the occurrences of the event and its replacement;
- qualification - the strength of the cause-effect relationship between the event and its replacement. We distinguish the following situations:
    - *ALWAYS (A)* the occurrence of the causing event implies the occurrence of the effect event.
    - *SOMETIMES (S)* the occurrence of the causing event implies the occurrence of the effect event with high probability.
    - *UNDEFINED (U)* the relation between the two events is unknown.

Finding the replacements for the unmeasurable events of the monitor is done in the following steps:

Step 1: defining the set of measurable events
This step aims at identifying the set of measurable events which are candidates for replacements and would provide the monitor with an insight into the actual application state. The data-event flow diagram which explicitly presents inputs and outputs to and from the control system is of use here.

Step 2: determining cause-effect relationships
For each unmeasurable event from the monitor model we develop a set of relationships which link this event with some measurable event through a chain of cause-effect dependencies. A technique similar to Fault Tree Analysis can be applied here.

Step 3: determining qualifications
In this step we qualify the cause-effect dependencies in the relationships identified in the previous step. The resulting qualification of the link between the unmeasurable event and the candidate for its replacement is equal to the qualification of the weakest element of the chain.

Step 4: choosing the best replacement
The final choice of the replacement is based on the following criteria:

- the qualification of the cause-effect link between the event and its replacement;
- the number of intermediate events in the relationship;
- the cumulative time delay.

Step 5: handling weak replacements
In case there are weak replacements (i.e. with S or U qualifications) we assume that the pessimistic approach is followed and that the replacement occurrence moves the monitor closer to the ALARM state even if it does not correspond to what is actually happening in the application.

Step 6: specifying compensation actions
The monitor should closely follow the state of the application. However, introducing replacements (which occur either before or after the events of interest) can cause continuous state drifts. If cumulated, the drifts result in discrepancies so that the monitor can loose its sensitivity. To deal with this effect we have to introduce some compensation and synchronisation actions which periodically restore the full correspondence between the application and monitor.

5 MONITORING SAFETY OF THE GAS BURNER

In this section we illustrate the method by developing a safety monitor for a gas burner system. To save the space we do not include here the complete description of the problem and do not present the details of its object-oriented model development. An interested reader can refer to (Górski and Nowicki, 1996). Below we illustrate the steps of the proposed method of the monitor synthesis. The complete case study can be found in (Górski and Nowicki, 1996a).

Step 1: Identification

The primary hazard in the gas burner system is the situation where gas concentration in the burning chamber is higher than a given safety limit. We assume that the direct measurement of gas concentration is not possible. Therefore, the monitor has to predict this parameter from other system parameters. As the hazard definition refers to the burning chamber, it is considered to be the critical object. In Figure 1 the model of dynamic properties of the burning chamber is presented. This model constitutes the preliminary safety monitor specification.

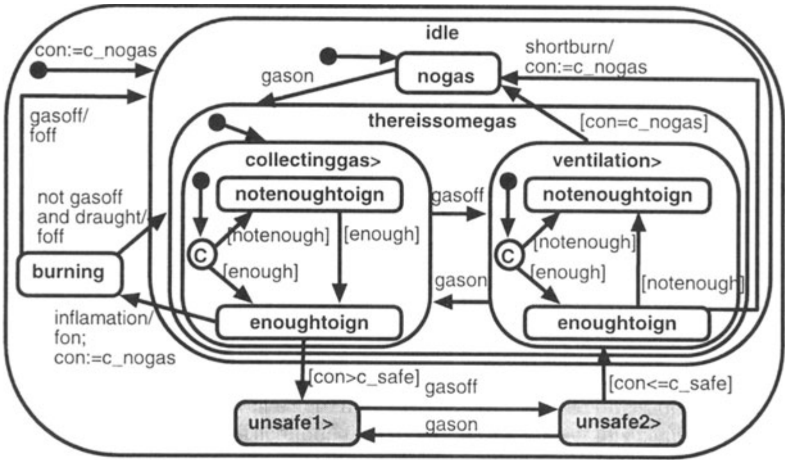

Figure 1 Model of the critical object.

The value of the `con` attribute represents the gas concentration in the burning chamber. This attribute changes linearly as the time is passing. The concentration increases in the `collectinggas` state (`con++`), and decreases in the `ventilation` state (`con--`). The burning chamber has two basic sates: `burning` and `idle` and two `unsafe` states. The initial state `nogas` (substate of `idle`) represents the situation where there is no gas in the burning chamber. When the gas is let in to the chamber (in effect of the `gason` event) the burning chamber moves to the `collectinggas` state and to its substate `notenoughtoign` which represents the situation where the concentration of gas is not sufficient to its inflammation. After the concentration goes above `c_ign` (this is represented by the `[enough]` condition) the burning chamber moves to the `enoughtoign` state. Then the `inflamation` event (meaning that there was `spark` occurrence under the condition that there was no `draught`) leads to the generation of the `fon` event (flame on), the concentration goes down to zero (`con:=c_nogas`) and the burning chamber moves to the `burning` state. Closing the valve (`gasoff`) generates the event `foff` and moves the burning chamber back to the `idle` state. If there is a `draught` while the gas is burning, the burning chamber moves to the `collectinggas` state and the whole cycle repeats. If in this state the valve is closed (`gasoff`) the burning chamber goes to a substate of `ventilation` state. Depending on the gas concentration value (conditions `[enough]`, `[notenough]`) this substate is either `enoughtoign` or `notenoughtoign`. While being in the state `enoughtoign` if `shortburn` event occurs (in effect of `spark`) the collected gas is instantly burnt out and the object moves to `nogas` state. An alternative way of arriving to `nogas` is when the gas concentration drops to zero due to normal ventilation. If while being in `ventilation` the valve is open again the burning chamber returns to an appropriate substate of `collectinggas`, depending on the current value of concentration. Unsafe states (marked in grey) have been identified according to the method presented in (Górski and Nowicki, 1996). They represent the situation where gas concentration is above the safe limit. In `unsafe1` the gas concentration still increases (`con++`) whereas in `unsafe2` the gas concentration drops down due to ventilation (`con--`). The `unsafe1` state is reached from the `enoughtoign` substate of `collectinggas` when the gas concentration goes above `c_safe`. After the valve is closed (`gasoff`) the burning chamber moves first to `unsafe2` state and then after the concentration drops below `c_safe`, it returns to the `enoughtoign` substate of `ventilation` (provided that there was no gas explosion in a mean time).

Step 2: Reduction

This step involves transforming the critical object model to the corresponding state machine without nested states, removing events sent to other objects, removing actions and activities updating the shared variables and having no influence on the critical attribute, and labelling the resulting states with the names of activities affecting the critical attribute. The result of those actions with respect to the model of Figure 1 is presented in Figure 2.

reconstruction of transitions:

- the transition between joined states are removed;
- the other outgoing transitions of the joined states become the outgoing transitions of the resulting state.

As the result, we obtain the monitor specification focusing exclusively on tracing the evolution of the critical attribute in the actual application. The monitor reacts to the same events which cause changes in the critical object. On the other hand it is entirely passive and has no influence on its environment.

Step 3: Implantation

Eventually, the monitor is to be implanted into the actual application. As the monitor is driven by the application events, the access to these events has to be provided. This means that the events have to be *measurable*, i.e. there must be some technical means to detect their occurrence. Normally, we can expect that all events exchanged through the interface between the plant and the control system are measurable (by utilisation of sensors, detectors and actuators). There is no guarantee, however, that all the events driving the monitor will be measurable as safety monitoring may require access to some application parameters which are beyond the control_system-plant interface. This problem is not trivial and sometimes can not be solved by just adding an additional sensor to the plant. This is because measuring of some parameters can be physically or economically infeasible. In this case we have to find a measurable *replacement* on the basis of which the occurrence of the event of interest can be inferred. The problem of replacements identification is discussed in the next section.

Step 4: Tuning

As the result of the previous steps we obtain the monitor which raises the ALARM signal at the moment the unsafe state is reached by the actual application. This is clearly too late as at this point the system is already unsafe. In this step we concentrate on building into the monitor the early warning facility. We concentrate on those states of the monitor specification which directly precede the ALARM state. Then we modify the transitions leading to ALARM by changing the enabling condition (to facilitate earlier firing of the transition). The exact modification is application dependent.

4 IMPLANTING THE MONITOR INTO APPLICATION

To implant the monitor into the actual application we have to ensure that the events which drive the monitor can be conducted from the application to the monitoring device. As it has been mentioned during discussion of STEP 3, some events may be unmeasurable and we have to find their replacements.

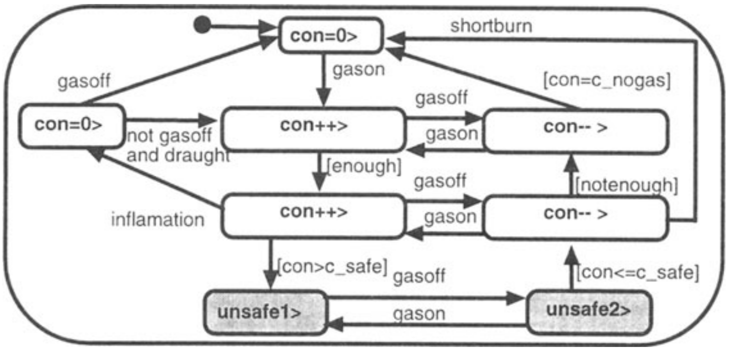

Figure 2 Reduced monitor model.

After joining together states with the same names and joining unsafe states into the **`alarm`** state, the model presented in Figure 3 is obtained.

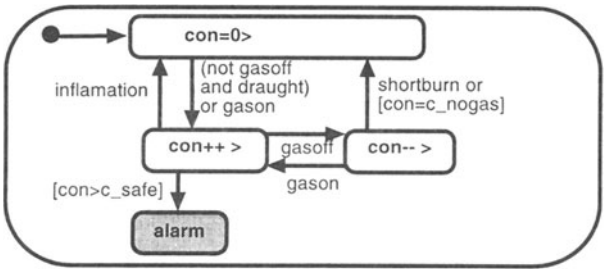

Figure 3 Model of the monitor after Step 2.

Step 3: Implantation

In the monitor model presented in Figure 3 all events (with exception of **`[con>c_safe]`** and **`[con=c_nogas]`** which refer to the internal variable of the monitor) are unmeasurable, i.e. there is no technical equipment which provides for detecting their occurrence. In this step the unmeasurable events are substituted by their measurable replacements. This is achieved by the following steps:

Step 1: defining the set of measurable events

The analysis of the event-data flow diagram of the whole application identifies measurable events presented in Table 1.

Table 1 Measurable events

event	*generated by*	*received by*	*description*
syggason	computer	gas valve	opens the valve
syggasoff	computer	gas valve	closes the valve
sygign	computer	ignition	initiates the **spark** generation
heatingon	operator	computer	external command to switch heating on
heatingoff	operator	computer	external command to switch heating off
off->on	temp sensor	computer	event telling that heat has begun to be produced
on->off	temp sensor	computer	event telling that heat has stopped to be produced

Step 2: determining cause-effect relationships

All information gathered during this step is shown in the Table 2. The column **event** contains unmeasurable events from the monitor specification (in bold). Columns **cause** and **effect** comprise potential replacements of the given unmeasurable event. If these events are also unmeasurable they are placed in the **event** column of the subsequent row and their replacements are searched in turn. The procedure is repeated recursively until a measurable cause(s) and/or effect(s) is found.

Table 2 Cause-effect relationships

q	*cause*	*delay*	*event*	*delay*	*effect*	*q*
A	**syggason**	1	**gason**			
A	**syggasoff**	1	**gasoff**			
			inflamation	1	**fon**	A
			fon	ttson	**off->on**	S
			not gasoff and draught	1	**foff**	A
			foff	ttsoff	**on->off**	S
U			**shortburn**			

Step 3: determining qualifications

The analysis of the model shows that whenever **syggason** or **syggasoff** occur, **gason** or **gasoff** occur respectively (with qualification A). Similarly, the occurrence of **inflamation** or **not gasoff and draught** causes always **fon** or **foff**. As the temperature sensor is characterised by some inertia only some **fon** or **foff** are sensed so the corresponding relationships with **off->on** and **on->off** have qualification S. Finally, we could not find any reasonable relationship of **shortburn** with other events (qualification U).

Step 4: choosing the best replacement

In this case there are no alternative replacements for the unmeasurable events.

Step 5: handling week replacements

For some monitor events only week replacements are available. In such case some decisions must be done on how to maintain the correspondence between the application and monitor.

We adopt the pessimistic approach. The monitor reaction (which may be 'do nothing') to the occurrence of a week replacement should move the monitor to a state (out of the two linked by the transition triggered by the replacement) which is 'closer' to the ALARM state. Following this approach, in Figure 3, as the **`shortburn`** event enables the transition to a state of a lower gas concentration, the transition is removed. The remaining events of the Table 2 are substituted by the corresponding replacements.

Step 6: specifying compensation actions
From the Table 2 we see that the replacement **`on->off`** occurs **`ttsoff+1`** time units after the **`not gasoff and draught`** event. Therefore, replacing **`not gasoff and draught`** involves adding to the transition an action (**`con:=ttsoff+1`**) which provides for compensation of the effect the time delay has on the **`con`** variable. Similar compensation actions have to be considered for the remaining replacements (note that the required compensation varies depending on if the replacement is a cause or effect of the original unmeasurable event and on the activities performed in the involved states).

Step 4: Tuning

To provide for early alarm warning we have to decrease the concentration limit which triggers transition to **`ALARM`** state. Therefore we replace **`c_safe`** with **`c_safe-margin`** constant. In general, this step is not trivial. In the case of our example we could exploit the fact that gas concentration is a continuous variable representing some physical attribute. The final version of the monitor is presented in Figure 4.

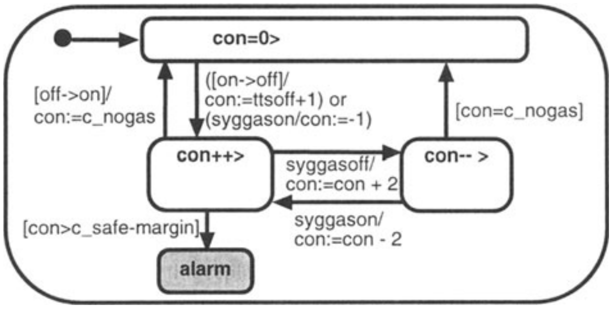

Figure 4 Safety monitor for the gas burner.

The model is driven by measurable events only and traverses the states which affect the value of the internal variable **`con`**. In case the value of **`con`** exceeds the specified limit the monitor moves to the **`ALARM`** state.

6 CONCLUSIONS

The method presented in the paper results in an application-specific safety monitor which is capable to reveal situations in which safety is about to be violated. The synthesis process starts with the object-oriented model of the whole application and the monitor is derived from the specification of a critical object. The prerequisite for the successful application of the method is the existence of a validated model of the critical object.

The key idea of the approach is that the monitor should primarily concentrate on a hazard regardless its causes. The method starts from the hazard and then works backwards to the extent which is necessary to implement the early warning facility.

The method starts from a hazard definition expressed in terms of critical object attributes and through consecutive transformations identifies the set of system characteristics which is sufficient to predict hazardous states. The method takes into account broad context of the application and is not limited to the control system only.

Although the steps of the method refer to the dynamic model, by virtue of the object oriented decomposition the scope of those steps is limited to selected objects (critical objects). Due to this limitation of scope we can expect a positive effect on the overall reliability of the proposed method. This is not possible in other net-based approaches not supported by similar decomposition (e.g. Leveson and Stolzy, 1987).

As the method is based on the object-oriented model of the application, some parts of the model can be re-used for other purposes (e.g. during development of the mission-oriented parts of the system). This requires some care, however, in order to address the problem of common cause failures.

We have performed several experiments with the gas burner monitor synthesised according to the presented method. They have confirmed that the monitor raises an alarm whenever safety is to be violated (satisfies the high sensitivity criterion). On the other hand, due to the inertia of the temperature sensor the monitor appeared to be too sensitive and in some situations raises false alarms.

7 ACKNOWLEDGEMENT

The authors would like to acknowledge the support of the EU Copernicus ISAT (Integration of Safety Analysis Techniques) project and to thank the anonymous reviewer for thoughtful comments.

8 REFERENCES

Górski, J. and Nowicki, B. (1995) *Object Oriented Approach to Safety Analysis*. Safety and Reliability of Software Based Systems ENCRES'95, Brugge, Belgium, September 1995

Górski, J. and Nowicki, B. (1996) *Safety Analysis Based on Object-Oriented Modelling of Critical Systems*. The 15th International Conference on Safety SAFECOMP'96, Reliability and Security, Vienna, Austria, October 1996

Górski, J. and Nowicki, B. (1996a) *Object Oriented Based Safety Monitor Synthesis*, EFP--RR167 (in Polish)

Harel, D. (1987) *Starecharts: A Visual Formalism for Complex Systems*. In Science of Computer Programming 8

International Electrotechnical Commission, *IEC 1508 Functional Safety: Safety-related Systems* (draft)

Leveson N. G. and Stolzy J. L. (1987) *Safety Analysis Using Perti Nets*. IEEE Transactions on Software Engineering, vol. SE-13, no. 3, March 1987

Redmill, F. J. ed. (1989) *Dependability of Critical Computer Systems 1, 2, 3.* Elsevier Applied Science

Rumbaugh, J. et al. (1991) *Object-Oriented Modelling and Design*. Prentice Hall Int.

9 BIOGRAPHY

Janusz Górski is the Head of the Software Engineering Institute in ITTI Poznań and a Professor in the Department of Applied Informatics, Technical University of Gdańsk, Poland. Published more than 100 papers. Lead several software projects in the areas of process control, telecommunications switching and databases. His present interests include computer software and system engineering, and systems safety, security and reliability. Served on IPC of many conferences related to those fields. Is the chairman of the *Safety Aspects of Distributed Systems* Working Group of EWICS TC7 and a member of IFIP WG5.4, EUROMICRO and IEEE.

Bartosz Nowicki graduated in 1992 from Technical University of Poznań. From 1992 to 1996 was affiliated as assistant in Software Engineering Group in Franco-Polish School of New Information and Communication Technologies. Presently works as a consultant in Institute of Software Engineering, ITTI in Poznań, Poland. His main research interests are safety analysis, safety monitoring and object orientation. He is a member of ISAT (Integration of Safety Analysis Techniques for Process Control Systems, programme Copernicus) project. He is a co-author of 5 publications on safety related problems.