THE ARCHIVES OF TRANSPORT
VOL. XXII NO 2 2010
10.2478/v10174-010-0015-0

The safety assurance method of railway control systems using object oriented languages

Marek Sumiła*
Andrzej Lewiński**

Received January 2010

Abstract

The paper deals with introduction of safety rules to the software designed for railway control systems. The basic assumption is related to software designed using high level language UML with possibility of modelling, verification, functional testing and simulation in CASE environment. The object methodology - the base of UML regards the software implementation with respect to safety and real time control corresponding to UIC recommendations and CENELEC standards.

Keywords: Railway Control systems, object programming, safety programming

1. Introduction

The computer railway control (RC) systems are treated as two layer systems: the hardware layer is a computer co-operating with external railway devices and software layer responsible for realization of control algorithms. These systems are designed according to the "fail-safe" rule assuring that each single system damage or fault is treated as safe (does not lead to the danger of system safety) and after its detection the initialization of special protection action takes place. In addition, the time necessary to detection of fault must be short adequately. The total safety measure of RC

* Faculty of Transport, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland, e-mail: sumila@it.pw.edu.pl
** Faculty of Transport and Electrical Engineering, Technical University of Radom, Malczewskiego 29, 26-600 Radom, Poland, e-mail: a.lewinski@pr.radom.pl

systems may be described using appropriate probabilistic and time characteristics well known from the reliability theory [22]. The safety of the software is connected with specification, coding, compilation method and application of appropriate tools in creation process. The methods recommended during design, implementation and maintenance of software dedicated to safety systems are documented in CENELEC standards [24][25][26].The choice of programming language for safety RC systems may be treated as an important. The earlier attempts of application of machine oriented languages, especially assemblers, show that such languages are a little efficient and even dangerous with respect to algorithmic structure languages such C, PASCAL, ADA and PLC [17]. (Now to this group the diagram oriented languages are enclosed.) Such approach gives the possibility to avoid the most part of syntax and semantic faults through clear relations between abstract algorithms defined in specification and data structure of created software.

The programming of processes connected with control and management in railway transport is designed according to software life cycle defined in the standard PN-EN 50128. The first stage of this cycle is formal specification. It must have the safety proof which will be implemented in appropriate way in the programming language as a adequate control algorithms. In the next stages the coding and testing of several new software modules are realised. Each of them is submitted for verification and validation procedure. From many formal and semi-formal methods recommended in appendix to the standard PN-EN 50128 the idea of finite state automata FSA may be very convenient to description of railway traffic control processes. This method is faithful because corresponds to typical non formal description (tables of dependencies) and is rather intuitive among railway engineers. The method presented in next part of the paper is consistent with Rational Unified Process method and corresponds to UML application to specification, simulation and verification of software designed for RC devices.

Another methods such Software Life Cycle Model applied to railway control base on cascade (Waterfall Method), evolutionary (Evolutionary Delivery), spiral or "V" models and ensure rather high level reliability of final project – safety software.

In the previous works the problem of correct software specification for RC systems is presented. The main approach to structured programming with formal specification of program correctness is rather sophisticated. The semi-formal methods may be also applied to design of proper implementation of control algorithms including techniques such redundancy and self-testing. The method proposed in the paper assumes using the semi-formal specification of railway control algorithms based on route method (tables of dependencies) and UML application.

The main feature of this method is using the object methodology OOM in which the basic railway control devices (points, semaphores, rail sections, etc.) may be defined as a sets of objects $\{O_1, O_2, \ldots O_i, \ldots O_n\}$. The control algorithm is implemented as a composition function of object behaviours. The object and connected functions are closed and may be used in special, controlled way. In the structured approach the formal correctness property may be defined as follows

$$C(IN_{1n}, Q_1, Q_n) \tag{1}$$

where

$$IN_{1n} = F\left\{IN_{ij}\right\}_{k=1}^{n} \tag{2}$$

Is a structural composition of elementary instructions IN_{ij}, $i = 1, 2, ...n$, $j = 1, 2, ...n$ between initial point "1" and final point "n". The Q_1 and Q_n are properties (expressed as mathematical relation form) before and after program execution. Such composition of global mega-instruction IN_{1n} may be decomposed to elementary instructions IN_{ij} and local states Q_i and Q_j proving correctness according to formal methods

$$F\left\{C(IN_{ij}, Q_i, Q_j\right\}_{i=1,2,...n, j=1,2,...n}^{m} \tag{3}$$

For instance, the partial correctness may be defined with respect to elementary instructions as

$$\left\{Q_i\left[IN_{ij}\right] \subseteq Q_j\right\}_{i=1,2,...nj=1,2,...n} \tag{4}$$

In the object oriented specification the correctness conditions assumed in (3) may be presented with respect to corrected implementation of object

$$F\left\{C(IN_{ij}, Q_i\right\}_{k=1}^{n}, |Q_i = Q_j \tag{5}$$

And now the partial correctness may be expressed as follows

$$\left\{Q_i\left[IN_{ij}\right] \subseteq Q_i\right\}_{i=1,2,...nj=1,2,...n} |Q_i = Q_j \tag{6}$$

It is mean that property of object Q_i is invariant ($Q_i = Q_j$) with respect to the execution of instruction IN_{ij} regarding the start (Q_1)and (Q_n) end of the program.

The validation of objects may be convenient during application of semi-formal techniques, especially testing and simulation. The verification of UML objects is necessary during software evaluation in the certification process of RC system (safety proof).

2. UML features vs. safety software of RC systems

In computer RC systems the control software is treated as a part with encryption of all (or almost) logical operation affecting to the whole device (system). Additionally, the software must operate with respect to real time restrictions, because the action of the device must be adequate quick to assumed events occurring in the environment. The appearance of each event is not tome determined, but the system response is always time limited. These systems are connected with special requirements corresponding to safety of passengers and trucks. The CENEELEC

recommendations (i.e. [24]) and UIC regulations demand the requirement of detection of all single catastrophic failures, and system reaction after failure leads to fail safe state [4]. It is obvious, that in computer oriented devices the main efforts are connected with software correctness and reliability.

The object, such mentioned earlier, is a some program creation characterized by strict limits of action with restricted access to its resources through external creations – other objects [9]. This characteristic property assigned to each object may have influence to the system safety designed in object oriented language. The object through introduced restrictions is resistant against occasional or intended changes of attributes and called methods.

The another feature of objects is treating as "black box" with precise defined input and output signals and assurance of autonomy of inner implementation. Such property has a possibility to create the objects in different ways using different groups of programmers. In the high safety systems the group of objects may be created realizing the same tasks but designed in different ways. The comparison of evaluation results obtained from these objects and putting to majority voting may be similar to "2 from 3" structures of computer systems [20][22][23].

The authors of UML attach the attention of object meaning for control systems and corresponding to this fact the classical definition of existence is extended to dynamic element[1].The introduced concept of active model contains the state machines (FSM automata) and input-output port sets. This concept will be assumed in the OMG and ISO standards several years later. The concept of active objects assumes that action of the object may be descript using states diagram with states and transitions passed by the objects. The change of state (passing via transition) may be caused by external event connected with object environment (i.e. after assumed time). Additionally, it is a possibility to restrict of initializing the action of transitions through defining the set of signals with authority for such action and other restrictions (i.e. range of values) without satisfaction which change of state will be not realised. The mentioned features of the object are elements with important influence to the safety of realised actions using software. For UML description of objects the object diagrams are used and for description of active objects the state diagrams and structure diagrams are used. The Fig. 1 presents the abstract diagram of object states.

In the OOM is assumed that each object of program is an instance of class from which is derived. It means that class is a given pattern from which the groups of object identical with respect to construction but different with respect to values of attributes and place of application are created. This feature is defined as reuse and corresponds both for class of passive (classical) and active objects. For safety of realised actions such feature is important because the correctness of all objects derived from the same class is assured upon condition that original class has been programmed correctly. For description of classes the class diagrams are used.

[1] The first result of such thinking has been version of Rose tool dedicated to real time systems.

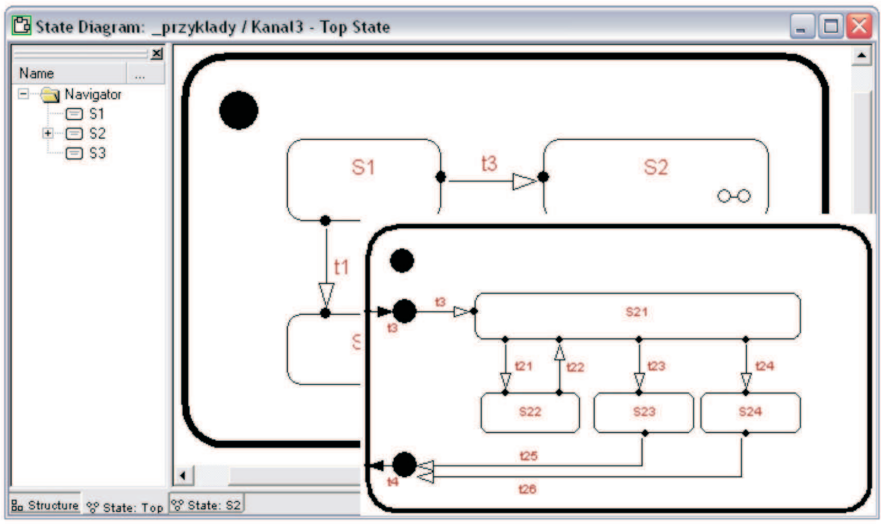

Fig. 1. Diagram. The nesting of states

The co-operation between objects is possible by relations defined between them prepared in the modelling process. The described earlier relations may be used to determine the co-relations between objects, acceptable number of objects (in the input and output of relations) and ways of exchange the signal (messages). Such defined relations restrict the situations with faulty calling the object or faulty signal transmission.

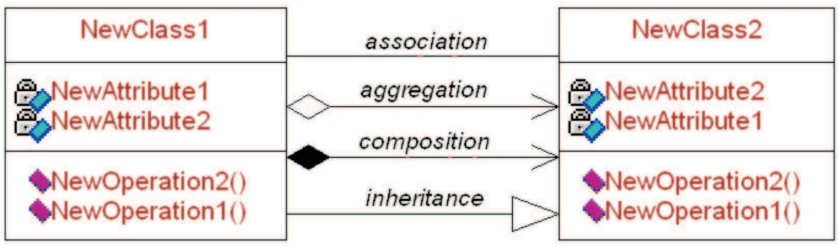

Fig. 2. The diagram of classes. The types of relations between two classes

For description of relation between objects (or class of objects) in UML language the diagram of objects (or classes of objects) is used, but for ways of transmission of signals – the diagrams of structure, co-operation and sequence. The example of such notation is presented on Fig. 2.

The authors of presented object methodology extend the mechanism of reuse through inheritance. It has a possibility of some detailed existences the inheritance of structure and behaviour of appropriate general existences. The application of such mechanism has leads to detailed general existences. Such approach has an influence

on the creation of many different variants of object classes for the same set of general assumptions, enforcement of readability of diagrams and decrement of probability of failure during programming of many similar but not identical programmable existences.

The extension of classical object methodology is a concept of relation describing interchange of signals between objects. Such interchange may be realised entirely via ports in which the model is equipped. The external and internal port are distinguished. The external ports are interfaces for communication between object and environment, the internal ports has a possibility of communication inside object. For each port (or rather for pair of communicating ports) the protocol containing set of input and output signals must be defined. The signals assumed for one port as an input for conjugate port are treated as output signals. Each signal may be additionally described by type and set of values possible to occur.

The relation object port – protocol is presented in accordance to class diagram. The relation of composition (Fig. 3) existing between object and port defines the close connection with protocol describing the communication rules via port. The protocols create the important element of monitoring and guarantee of safety communication in the system.

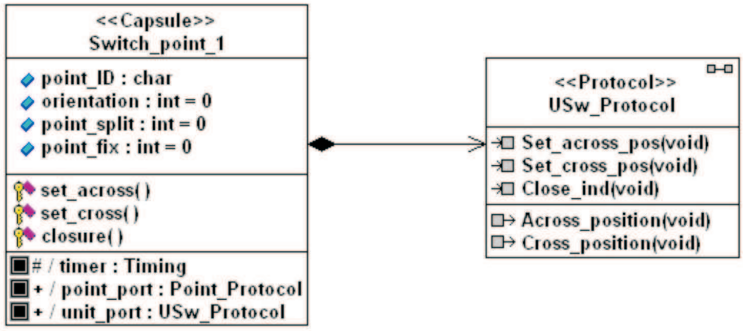

Fig. 3. The diagram of classes. The relation port – protocol

The diagram with more detailed description of port – protocol relation is diagram of structure. It presents the types of object ports (black squares) and channels with connected external objects (grey squares). The example of structure diagram is shown on the Fig. 4.

The all transport control systems are real time. Then it is important to define how the UML may satisfy the demand on service of time dependent events.

From the version 2.0 the UML gives the possibility of creation of software with information about passage of time. This information may cause the change of standard behavior of model or activation of one or group of models. For real time properties of the model the activity diagrams, state machine diagrams, sequence diagrams and time diagrams are applied.

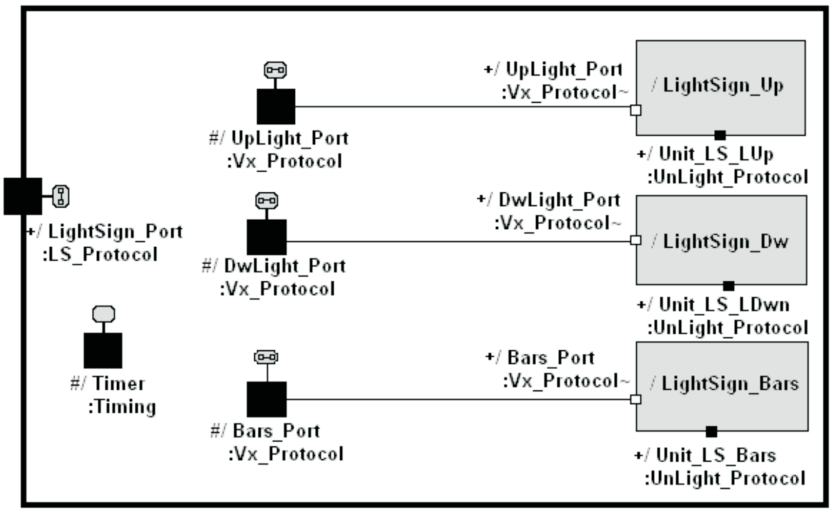

Fig. 4. The diagram of structure. The object of semaphore

In the Rose Real Time, which is not designed for service of time depended functions (in nowadays meaning) the TimerPackage has been introduced with main class RTTimespec. It gives the functions necessary for acquisition of clock signal – getlock(), the determination of time value – adjustTimeBegin() and adjustTimeEnd(), and many others.

Corresponding to needs connected with system specification the application of global clock or some local clocks with respect to the number of realized processes may be possible.

3. The creation of software model

The software of RC system is created as a result of information analysis related to specification. Such specification must be elaborated by customer and must satisfy the requirements defined in international standards and internal regulations of railway company [7].

The design of software with respect to specification in UML is easier through simple process of processing of information contained in specification to the programming language level.

The works of software design are divided into two basic stages. The first stage is connected with creation the software model for ideal conditions. Such model requires only the correct work of the system and its verification allows to interest of correct implementation of control algorithms and detection the most part of faults introduced during modeling. After this stage, the next stage takes place when the model will be extend with the service of events during system damage. The process

of model extension requires the introduction of additional objects and states with possibility of proper behavior in the situation different from assumed. The software model for real conditions of system work must respect:

1. The faulty action of traction vehicle (i.e. interrupt of sequence, back off to passed rail section, rip of point, etc.).
2. The faulty service of system.
3. The damage of rail circuit elements (the break of rail, the blow of semaphore lamp, the stoppage of point, etc.).
4. The damage of disposition computer devices (i.e. the loss of power, the reset of disposition computer, the untruth of transmitted data, the deadlock of control processes, the overrun the limit of waiting time for response, the loss of synchronization, etc.).
5. The faults of system operator (i.e. leading the vehicle to busy rail, the release the rail route with vehicle, etc.).

For system damages the following aspects may also affect:

6. The environment (i.e. high temperature, humidity, atmospheric discharges, etc.).
7. The animals (i.e. rodents, etc.).
8. The sabotage (i.e. terrorists).
9. The other unknown facts.

The application of OOM allows to fast and safety design of model with respect to the real conditions of system work. This result is achieved through the property of inheritance as safe way of extension the existing object and creation the new object specially devoted for service of potentially critical events.

The Fig. 5 shows intuitively the mechanism of extension the model created in the first stage to the full description as an aim of stage two. The platform of real system inherits the structure from the platform of ideal system with extension of additional state S_Alarm. The transition of automata to the state St_Safe is caused by the event from outside of typical disposition maneuvers (i.e. the external signal, the lack of reaction in assumed time, etc.).

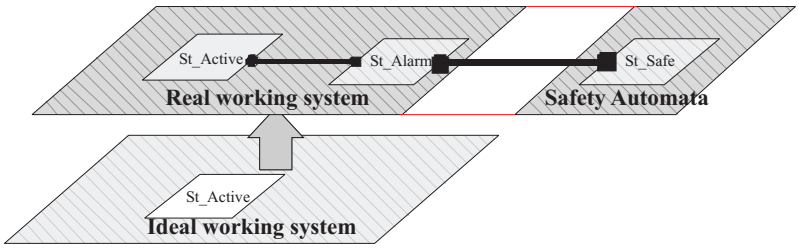

Fig. 5. The platforms of RC system model

Corresponding to mentioned earlier stages the model related both to software structure and automata description for several objects is created. In this method these two description are treated as two co-existing platforms: one contains whole infor-

mation about structured elements (objects, their number and connections, limits, port descriptions, protocols, module co-operation), the seconds reflects internal building of automata together from existing set of input-output signals, states, transitions, set of limits connected with execution time of tasks.

It is assumed that the class diagram is a basic diagram for description of the software structure. This choice is connected with high universality and capacity of descriptions represented by it. The presentation of structures of packages, classes, objects with description of attributes, methods, ports and connections between them. In addition, this type of diagram allows to document the structure of protocols and related signals, and assign the comments about number and limits of objects.

The second important type of diagrams of structural layer is diagram of structure. Using such construct the frame of object reaction, sets of external and internal ports, signal transmission channels are defined and protocols assigned to each port are chosen. From point of view of whole structure platform the diagrams of structure are important complement of diagrams of classes.

In the frame of structure layer the side of implementation of whole model into conditions of physical device is very important. It is connected both with the choice of processor for final device and choice of operating system creating the work environment of software. The specification of such special features of software is presented on the implementation diagrams.

For description of automata platform (layer) the state diagrams are always used. These diagrams reflect work of so called state machines. These diagrams contain the states in which the given object may occur and relations between states called transitions. The structures of states may be hierarchical, it is means that each from the states may have its own internal building. The transitions are activated as a result of existing the external and internal events received from input ports.

Beside the state diagrams the diagrams of sequences and activity are also used for description of software model. Using them the designer describes the relations between objects with respect to passing time. The verification of these diagrams always answers on the questions connected with incorrect system functionality.

The example of software redundancy modelling

The most popular architecture of safety system is Multi-Channel Voting Pattern architecture [9] [19][20][33]. It is based on the rule of "limited faith" of the decisions taking by the one module (subsystem, object, information processing channel), because the assigned task is realised by at least two independent modules and their results is compared. The two diverse architecture versions of architecture based on voting mechanism is assumed.

The Homogenous Redundancy Pattern applies the multiplication of number of channels with satisfaction of their identity. The architecture has a feature of

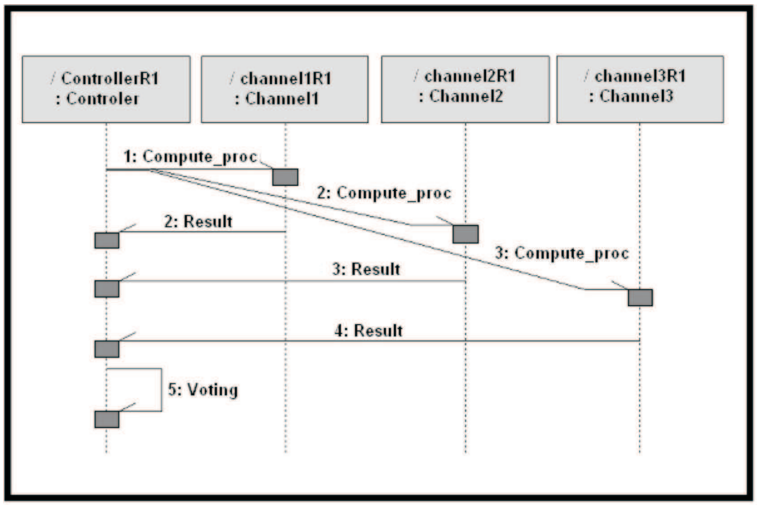

Fig. 6. The diagram of sequences. The redundant structure with voting

reliability structure because such solution assures against damages of single channel with respect to parallel work of all channels (in the structure with parallel taking of decision), but without protection against failures caused by improper implementation of channel.

The second from discussed redundant structure of choice by voting is Diverse Redundancy Pattern. Its implementation assumes that realisation of each channel may be different. It is connected with application of different coding algorithms, different program compilers, etc. Such diversity is also connected with applied hardware (i.e. sensors of different producers) or human resources (different programmers teams). The example structure composed with three evaluation objects and one voting object is presented on the Fig. 6.

The example of modelling of rail route elements

The physical elements of rail route are a factor affecting to the moving traction vehicles. The software of computer system must in safe way realize monitoring and service of RC devices. In the presented method the creation of separated group of objects dedicated for projection of action of such devices as a form of virtual elements is proposed. There are a type of interface connecting the interlocking subsystem with physical devices of rail infrastructure. These elements are made as a transformation of formal description contained in the specification to the form of class. They are part of the set of object describing the route rail.

As an example of such element is a rail point. It is responsible for driving the rail vehicles to the appropriate route rails. The corresponding object model is presented on the class diagram. On the Fig. 7 the relation of insulated rail section as

part of route rail is presented. Such relation of element function in UML is created by composition relation indicating the relation between two elements.

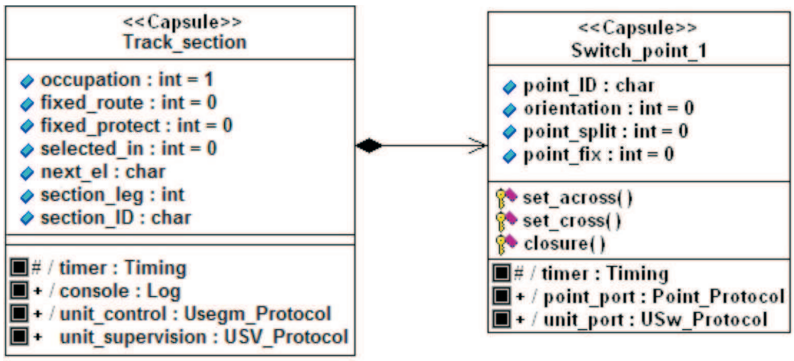

Fig. 7. Diagram of classes. The model of relation between two objects Switch_point_1 and Track_section

The more important attributes of point are described as follows:

– *point_ID* – the attribute containing the unique with respect to the whole rail route the point identifier,
– *orientation* – where its value is determined corresponding to inner action of the point automaton with actual state of the point,
– *point_split* – attribute defining the state of possibile rip of the point, the default value is assumed as zero – lack of the rip,
– *point_fix* – attribute of the object state checking informing about fact of confirming the point in the route,

The communication of the object with the environment is possible via two ports point_port and unit_port. The first of them is responsible for communication with another objects of interlocking system, the second is responsible for co-operation with physical device. The object ha been equipped in additional port Clock which allows to control of response time of physical element on thesent command.

As an example the Point_Protocol (Fig. 8) is assigned to port point_port. The signal received via such port have direct influence to the action of the state machine of object Switch_point_1. The list of signals transmitted between objects Set_up and Switch_point_1 contains the messages about changes of state related to executing action in the object and damage events such rip of the point.

The safety of the automaton of the point object is achieved by introduction appropriate states and transition between them.

The automaton of point from the Fig. 9 beside the states describing following settings of the point contains also the state St_FAILURE. The transition of automaton to this state is realised only in the case of the lack of the response of physical device in assumed time. The signal initialising the transitions leading to this state is Timeout signal from internal port of timer control. Additionally, the automaton occurring

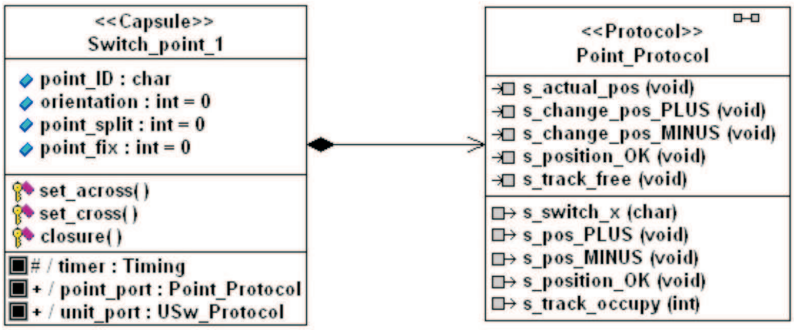

Fig. 8. Diagram of classes. Attachment of protocol Point_Protocol do active object Switch_Point

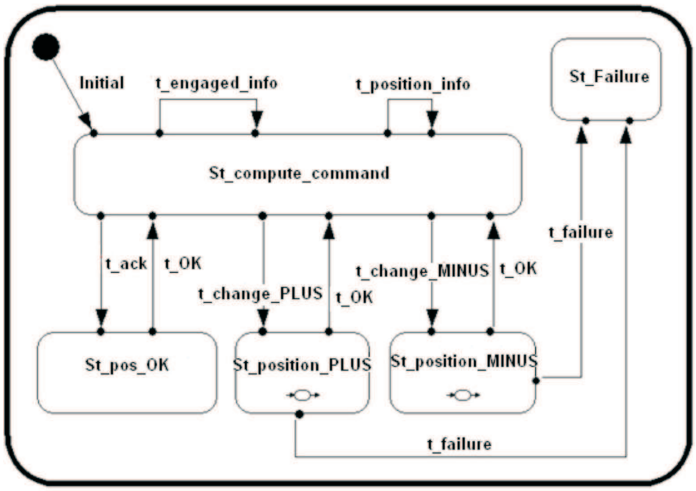

Fig. 9. Diagram of states. Automaton of Switch_point object

in the waiting state for control command St_compute_command the automatically initialises, after constant time period, the transition t_position_info. The main aim of such action is continuous supervision of the point state – position or rip. The signal initialising the transition is such previous the signal Timeout. The state St_FAILURE is an active state activating the automata responsible for the system safety.

4. Verification

The verification of correctness of presented method is based on realization of given number of tests corresponding to quality evaluation of elaborated software model together with design comfort [28]. The testing of the model elaborated in UML may be done through the analysis of content of diagrams and a result of simulation.

The comfort of creation the possibility of testing the models built in UML depends almost from CASE environment accessible for designer. In general the tests of UML models may be divided into two groups:

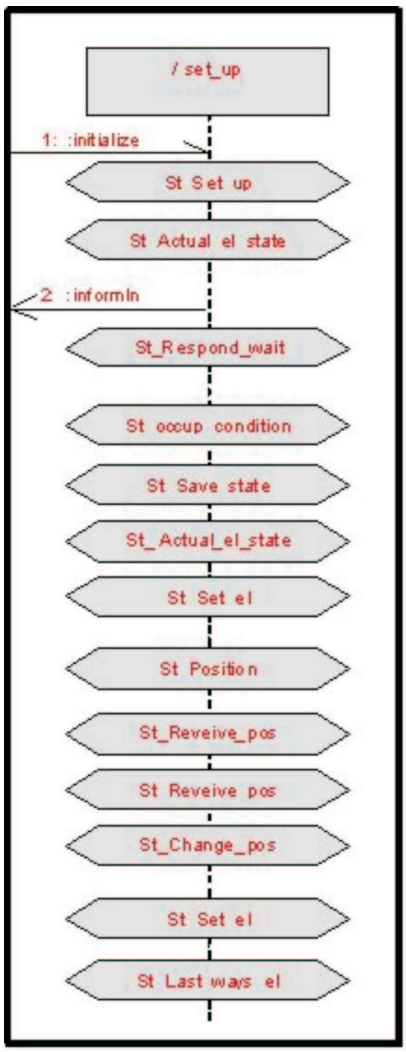

Fig. 10. Diagram of sequences. The flow of control process in object Set_up

– external tests,
– internal tests.

The **external tests** in many cases relay on the analysis of final code made after conversion of UML diagrams to compiler language. Such analysis is not used for correctness verification of whole system encoding g, but rather for to part or for

checking the corrected t6ranslation the chosen part of the system, automaton or function in target language.

For second group of the external tests may be enclosed the tests of communication of working program with the external simulation environment. As an example may be realised attempts of connecting the control systems executed in Rose Real Time environment with Matlab/Simulink environment.

The **internal tests** belong to the group of of test executed in the UML modelling environment. Corresponding to applied CASE tool may be accessible:
– verifiers of correctness and consistency of diagrams and code,
– simulators of state machines automata,
– diagrams of sequences of system work,
– monitors of message flow,
– monitors of supervision of realisation time of events.

The verification of sequence of executed actions

The verification test of executed action sequence may as a main aim the correctness validation of system processes. Thanks of object oriented approach in UML the analysis of single objects is possible exactly. As a result of analysis the observations of simulated state diagrams, sequence of signals from object ports, information transmitted to the screen of operator console and automatically generated diagrams of transitions creating as a result of simulation may be accessed. It is possible to restore the list of events occurring in assumed time, but such possibilities will be presented in the next chapter. The example presenting the verification described earlier may be object Set_up. It is co-operates with initiating object Choice and object Switch_point_1 responsible for execution commands. As a result of simulation of action of this object is a diagram presented on the Fig. 10. It presents the proper setting the point with respect to the route assumed in the object Choice.

The Fig. 10 shows all following states in which the system appearance during realisation time of setting task.

With respect to base of obtaining diagram the verification of occurrence the object automaton in given states and connecting with it consequence is possible. The all exeptions from assumed sequence of states, earlier (not assumed) termination and looping of automata are detected immediately by such tests.

The analysis of system time responses

The analysis of system reaction time for environment events is especially important transport control systems. Such type of verification of events only during simulation process with CASE tool is strongly restricted with respect to the work

outside target environment (controller) and lack of contact with physical devices. But it gives the possibility of detection the many specific disturbances typical for real time systems such deadlocks, non terminated processes, loss of memory content et. [9][14][19].

To execute the analysis of object reaction time to the external events (signals) the so called "Trace monitor" is used. The Trace monitor is given type of analyser with main task- the documentation of time periods connected with occurring the object in determined states. Additionally, the reading of transition caused to the initialising the event ma be possible. Using Rose RealTime tools the monitors may assign to any object, port or state.

The most popular test is analysis of input and output information from assumed port of the object. The Port Probe is installed in the structure diagrams. The figure presented below presents the structure diagram of object Set_up with attached probe of trace monitor. Corresponding to the base of list of traces is possible to test the time, direction, priority and value of input or output signal of the object via examined port (set_up_portA).In the case of the monitoring of states the analysis of communication via object ports is possible together with signals transmitted to and from the object according to the state. This state is presented on the Fig. 11.

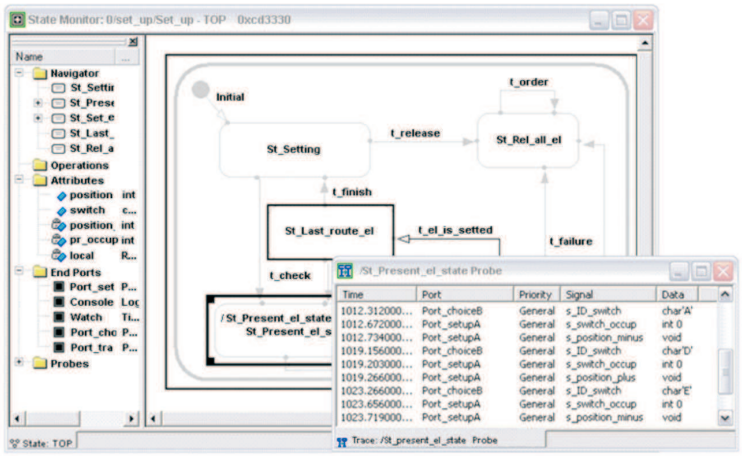

Fig. 11. The verification of events in the Rose RealTime environment for object Set_up

Verification of protection against system work interferences

The method presented earlier have the aim – checking the software behaviour with respect to situations not corresponding with assumed action plan. It is assumed that following situations ma be connect with:

– lack of answer for calling (overrun the time limit),
– the reception of faulty type of signal by object port,
– the reception of signal carrying the data besides assumed boundaries.

The all mentioned situations are connected with faulty communication in the system. The reasons of its occurrence are explained in the example of object Switch_point_1.

The lack of answer (response) in assumed time in the object switch_point_1 may exist after sending the command (signal) to the rail device enforcing the change of actual point setting. If such task is not executed in assumed time period the activation of transition t_failure will be caused. This transition is activated using the system predefined signal Timeout.

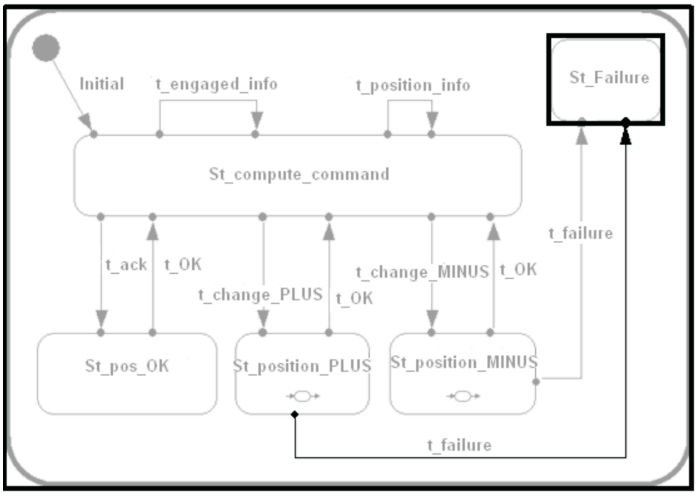

Fig. 12. Diagram of states. Simulation of object. Event – Lack of answer In assumed time

The second test of object behaviour after interference event is introduction the signal not corresponding to the specification of actual state during service. The UML implemented in the CASE tool reacts on such situations through sending the appropriate message to the screen of system operator and occurrence in last correct state.

The third test verifying the safety behaviour of the system is introduction of signal carrying of information besides range of assumed values.

The UML has very simple mechanisms of verification the data introduced to via port to object and it is connected only with type of receive signal. If the signal is defined as an integer and transmitted value is a float type then the fault will be detected and blocked by the operation not permitting to transmit the signal outside port. If the type of variable is correct but only the value is besides the range of assumed values then such situation will be not detected. As a solution may be the

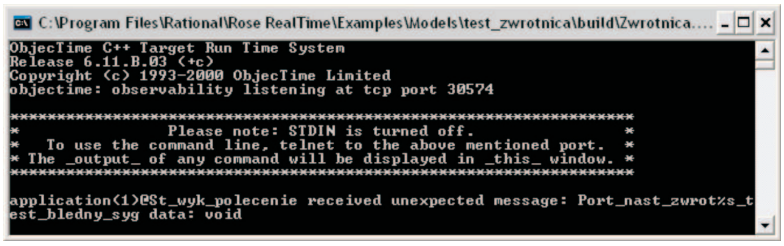

Fig. 13. The screen of terminal. The message about receiving the unidentified signal

additional description with acceptance of signal with respect to input condition of the state. Such description ma be done in target compiler language.

In presented example the values besides the range are numbers different than zero and one. The example code in C++ has the following form:

```
int receive = *rtdata;      //reading the value from signal and
                            assigning to argument received.
If (receive == 0)
{
actual_occupy = 0;          //if received value is 0, it is mean that
                            element is not occupied
}
else
{
Actual_pos = 1;             //for each other case the safety state may
                            be establishded - occupied.
}
```

5. Conclusions

The main aim of the paper is presentation of software design method for RC systems assuring the more simple implementation of high reliability and safety requirements. The object methodology used during creation process has the possibility of application the encapsulation and renewal using tools as elements supporting the process of safety software creation. It is possible through creation the software model working in ideal conditions and until after proof of such implementation correctness the necessary elements connected with the influence on the behavior in the damage situations are attached.

The introduction of UML 2.0 as a software structures description language gives the possibility of requirement the time related tasks in the system. The system

reaction mechanism is important for assumed time event from point of view of the safety of realized control processes.

The presented examples of OOM and UML application may be treated as an representative for most number of software module models of transport control systems.

In the final part of the paper the way of verification of models using CASE environment has been presented. The realised experiments show that the verification range depends In the great measure from assumed CASE environment [31]. The presented simulation methods must be treated as non obligatory for another tools.

References

1. Bluemke I.: Evaluation of object metrics in a CASE. Proceedings of the IASTED International Conference Software Engineering. ACTA Press, Innsbruk, 2004.
2. Booch G., Rumbaugh J., Jacobson I.: UML przewodnik użytkownika. WNT, Warszawa, 2001.
3. Christov Ch.: Problems of safety electronic systems of railway signalling. Monography, The Universitet of Technology in Sophia, Transport Departament, Sophia, 1988.
4. CNTK 1060/23: Wymagania bezpieczeństwa dla urządzeń sterowania ruchem kolejowym. CNTK, Warszawa, 1997.
5. CNTK 4T12C00529: Wpływ nowych technologii informacyjnych na poprawę funkcjonalności i bezpieczeństwa ruchu pociągów. Warszawa, 2006.
6. Coad P., Yourdan E.: Analiza obiektowa. WNT, Warszawa, 1993.
7. Dąbrowa-Bajon M.: Podstawy sterowania ruchem kolejowym. Oficyna Wydawnicza PW, Warszawa, 2002.
8. Derezinska A., Bluemke I.: A framework for identification of dependency areas in UML designs. Intern. Conf. Software Engineering and Applications SEA'05, Phoenix, AZ, USA, Acta Press, 467 – Software Engineering and Applications – 2005.
9. Douglass B. P.: Doing Hard Time – developing Real-Time Systems with UML, Objects, Frameworks, and Patterns. ADDISON-WESLEY, Massachusetts, 1999.
10. Dyduch J., Kornaszewski M.: Systemy sterowania ruchem kolejowym. Wydawnictwo Politechniki Radomskiej, Radom, 2004.
11. Evans A.,France R.,Lano K.,Rumpe B.: Developing the UML as a Formal Modelling Notation. In Proceedings of UML'98 – The United Modeling Language. Beyond the Notation. Volume 1618 in Lecture Notes in Computer Science. Springer-Verlag, 1998.
12. Grochowski L.: Narzędzia informatyczne w transporcie, Prace Naukowe Politechniki Warszawskiej, TRANSPORT, z.45, Warszawa, 2001.
13. Grochowski L.: Strukturalne i obiektowe metody projektowania systemów informatycznych, Prace Naukowe Politechniki Radomskiej TRANSPORT, 1, 15/2002.
14. Hooman J., Mulyar N., Posta L.: Validating UML models of Embedded Systems by Coupling Tools. Embedded Systems Institute, Eindhoven 2004.
15. Huzar Z.: Semantyka czasu rzeczywistego map stanów w UML. Materiały konferencyjne – IX Konferencja Systemy Czasu Rzeczywistego, Ustroń, 2002.
16. ISO/IEC 19501: Unified Modeling Language Specification. Version 1.4.2. Formal/05-04-01.
17. IEC 61131-3: Programmable controllers. International Electrotechnical Commission.
18. Jaźwiński J., Ważyńska-Fiok K.: "Bezpieczeństwo systemów", PWN, Warszawa, 1993.
19. Kopetz H.: Real-Time Systems. Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.

20. Lewiński A., Konopiński L., Siergiejczyk M.: Niezawodność i bezpieczeństwo wybranych komputerowych systemów sterowania ruchem kolejowym. Materiały konferencyjne – Krajowa Konferencja Bezpieczeństwa i Niezawodności KONBiN '99. Zakopane-Kościelisko, 1999.
21. Lewiński A., Sumiła M.: The semi-functional and reliability modelling of railway control systems. Zeszyty Naukowe, TRANSPORT, z. 51, Gliwice 2003.
22. Lewiński A.: Problemy oprogramowania bezpiecznych systemów komputerowych w zastosowaniach transportu kolejowego. Monografia. Wydawnictwo Politechniki Radomskiej, Radom, 2001.
23. Meyer B.: Programowanie zorientowane obiektowo. Helion, Warszawa, 2005.
24. PN-EN 50126: Railway Applications: The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS). European Standard CENELEC, September 1999.
25. PN-EN 50128: Railway Applications: Communication, signaling and processing systems – Software for railway control and protection systems. European Standard CENELEC, March 2001.
26. PN-EN 50129: Railway Applications: Safety related electornic systems for signaling. European Standard CENELEC, 1998.
27. Subieta K.: Obiektowość w projektowaniu i bazach danych. Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1998.
28. Sumiła M., "Metoda tworzenia oprogramowania sterującego w systemach sterowania ruchem kolejowym", Wydawnictwa Politechniki Warszawskiej, Warszawa, 2007.
29. Sumiła M.: Projektowanie systemów sterowania z wykorzystaniem Rose RealTime. Materiały konferencyjne – IX Konferencja Systemy Czasu Rzeczywistego Ustroń, 2002.
30. Sumiła M.: Próba realizacji oprogramowania bezpiecznego systemu srk z wykorzystaniem technik obiektowych. Prace Naukowe Politechniki Radomskiej, TRANSPORT 1(17), Radom, 2003.
31. Sumiła M.: State of the art of the simulation in tools to design complex control systems. 5th European Conference Of Young Research And Science Workers In Transport And Telecommunications, Zilina, 2003.
32. Wawrzyński W., Kochan A.: The Object-Oriented Modelling of Control Systems in Transport. Prace Naukowe Politechniki Warszawskiej, TRANSPORT, Warszawa, 2001.
33. Wawrzyński W.: Bezpieczeństwo systemów sterowania w transporcie. Biblioteka Problemów Eksploatacji, Warszawa-Radom, 2004.
34. Yourdon E., Argila C.: Analiza obiektowa i projektowanie – przykłady zastosowań. WNT, Warszawa, 2000.
35. Zabłocki W.: Model stacyjny urządzeń sterowania ruchem. Prace Naukowe Politechniki Radomskiej, TRANSPORT 1(13), Radom, 2000.
36. Zabłocki W.: Modelowanie stacyjnych systemów sterowania ruchem kolejowym. Prace Naukowe Politechniki Warszawskiej, TRANSPORT, z.65, Warszawa, 2008.
37. Zabłocki W.: Modelowanie systemów sterowania ruchem kolejowym struktury informacji i elementy opisu formalnego. Prace Naukowe Politechniki Warszawskiej, TRANSPORT, z.57, Warszawa, 2006.
38. Zabłocki W.: Modelowanie systemów sterowania ruchem kolejowym. Prace Naukowe Politechniki Radomskiej, TRANSPORT 1(17), Radom 2003.