Journal of Polish CIMAC
Gdansk University of Technology
The Faculty of Ocean Engineering and Ship Technology

DYNAMICS OF THE COMPUTER-AIDED SYSTEM FOR COMPLEX TECHNICAL OBJECT MAINTENANCE IN THE UML LANGUAGE

Andrzej Erd
Technical University of Radom
ul. Malczewskiego 29, 26-600 Radom Poland
tel. +48 48 3617740
e-mail:andrzej.erd@gmail.com

This paper is an attempt to use modeling methods employing the UML language for computer modeling of the complex object maintenance systems. A particular emphasis was put on the description of the system dynamics understood as changeable behavior resulting from the interaction with environment. The application case which has been selected here is a computer-aided system for railway track vehicle maintenance.

***Keywords**: UML application, maintenance system design, track vehicle*

1. Introduction

Economy of any modern country possesses many objects of great value and a relatively long life-cycle within which the phases of utilization and maintenance occur interchangeably in accordance with the rules of object utilization (the means of transport can serve here as an example, e.g. road vehicles, boats or airplanes). The cost of an object purchase is frequently much lower than later outlays on its operation and maintenance in the course of its life-cycle. At a certain moment the rational approach to the problem of maintenance can lead to a decision about a withdrawal of technically still worthy vehicles but outdated because new objects appear on the market whose technical parameters are much better or the costs of their maintenance are much lower [4].

A synthetic analysis of the phenomena and making the right decisions related to them is impossible without appropriate computer systems providing support for business processes. The need to create the Computer Aided Maintenance Systems (CAMS) results also from changes in the service systems and replacing traditional maintenance-repair routines by servicing based on the actual state which in turn is based on current diagnostics. Thus, it is vital to gather data on reliability and maintenance parameters for particular objects which undergo maintenance. In this case assistance is provided by diagnostic systems on board. However, in order to collect information referring to the entire number of objects and to draw more general conclusions, its aggregation is necessary.

Whereas representation of economic events is fairly well developed, the technical events are often insufficiently illustrated and in practice many attempts of developing such systems have failed [8]. The major reason for such a state of affairs is the immense complexity of systems [7]. Apart from other causes mentioned in this paper [7], the said failure results also from:

- a great number of events of extreme variety which must be taken into account,

- communication problems within designers' teams as well as among their members and users in the phase of formulating assumptions.

Consequently, the final product requires multiple changes, corrections and supplements which lead to a serious extension of the time scheduled for work and significant exceeding of the preplanned budget [8]. As a result, the systems created – if they are created at all – are in a version significantly reduced in relation to original plans and their costs are relatively high.

Another serious difficulty is the fact that each time the system is created from scratch without taking advantage of the existing partial solutions. Partial results are most often inaccessible as they are the property of the companies which ordered them or software manufacturers.

Among many [2] possible ways of reducing complexity, one of the most promising is system modeling before its execution is started, or, consequently, creating design models. This paper is an attempt to apply the modeling methods with the use of the UML language for a design of a complex object maintenance system. The application cases were developed on the basis of the computer aided rail vehicle maintenance system.

2. System modeling

Creating model systems has this advantage that a model is not only a fragment of the documentation of the created system but it is possible to verify the correctness and completeness of the system being developed before it is actually created. In this way the number of indispensable modifications required for the preparation of a final product is limited.

Together with the growing complexity of systems, the number of people involved in the design grows. Consequently, significant communication barriers arise and then models become a basic tool of documenting mutually agreed solutions. Naturally, to this end the programming tools from the CASE group are necessary.

A model system consists of:

- Information components – static description – database architecture
- Result information and the way of its presentation – user's interface
- Data processing method – application logic

This is one of the most frequently used system divisions – a division into layers. The essence of the layer architecture is to construct the application structure in such a way as to make the functions of particular layers independent of one another.

Some authors distinguish more layers. For example, a 5-layer architecture is outlined in work [6].

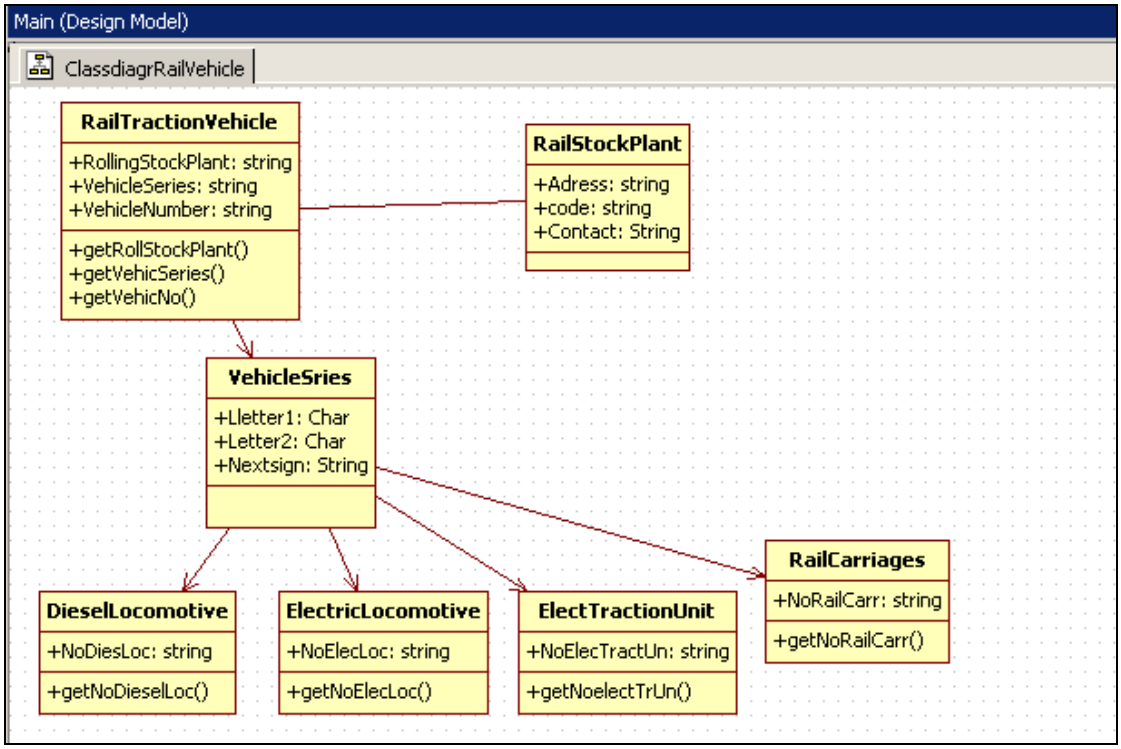

Fig. 1 Simplified model of the rail traction unit description class

A significant part of the static description is possible to achieve by means of class diagrams. They show the attributes constituting a class as well as methods operating on them. Figure 1 shows a simple model of classes included in a description of a rail traction vehicle which is the subject of the rail traction vehicle CAMS.

This model enables grasping the interactions among descendant class attributes contained in a description and the methods operating on them. It also allows us to distinguish between private and public methods.

The advantage of class diagrams and models achieved with their help is a possibility of automatic generation of programme codes on their basis.

```
Katalog: D:\_PR\Publikacyjne\Cimac11\railvehicle
10:55               254 DieselLocomotive.cpp
 10:55               354 DieselLocomotive.h
 10:55               258 ElectricLocomotive.cpp
 10:55               362 ElectricLocomotive.h
 10:55               257 ElectTractionUnit.cpp
 10:55               363 ElectTractionUnit.h
 10:55               244 RailCarriages.cpp
 10:55               339 RailCarriages.h
 10:55               349 RAilStockPlant.h
 10:55               368 RailTractionVehicle.cpp
 10:55               474 RailTractionVehicle.h
  10:55               280 TractVehicle.h
  10:55               341 VehicleSries.h
```

Fig. 2 Files generated on the basis of a class diagram presented in Fig. 1

```
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : RollingStockPlant
//  @ File Name : RailTractionVehicle.cpp
//  @ Date : 11-06-16
//  @ Author : Andrzej Erd
/

#include "RailTractionVehicle.h"

void RailTractionVehicle::getRollStockPlant() {
}

void RailTractionVehicle::getVehicSeries() {
}

void RailTractionVehicle::getVehicNo() {
}
```

Fig. 3 Contents of a File „ (RailTracktionVehicle.cpp)"

Depending on the tool applied, the code can be generated in different languages, e.g. C/C++, Java, Visual Basic, Delphi, JScript, VBScript, C#, VB.NET. For the sake of example the StarUML programme was used and the language set for generation was C++. The effect is presented in Figures 2 and 3.
It is noteworthy that all necessary code files (.cpp) as well as header files (.h) were created. The files generated contain all attribute declarations and methods, which can be clearly seen in the presented file "PojazdTrakcyjny.cpp" (RailTrackVehicle.cpp). It is obvious that in the targeted model of the system, the sets of attributes for each class should be more extensive [3]; the same refers to the list of available methods.

It is extremely important for tool programmes to check the correctness of declarations. Correctness is checked in its formal aspect. What is also checked is completeness of all declarations and coherence between individual classes. It is obvious that the method code depends on the programmer's will and must be completed by him. However, support of the tools is also significant

3. Models of system dynamics

The term "system dynamics" denotes system changeability, not only in terms of parameters but also in terms of behavior. What happens to the system is visible outside through the user's interface. The Use Case Diagram (UCD) illustrates users' interaction with the system as well as the system's interactions with environment. In this diagram particular groups of users, the so called "actors" are joined to the rounded rectangles representing activities by means of arrows. The scope and degree of information processing depend on the recipient and the goal. Hence the Use Case Diagram allows us to grasp all forms of collaboration between the environment and the system. In the description of activities which is an indispensable supplement to UCD, all aspects of the user-performed activity should be specified.

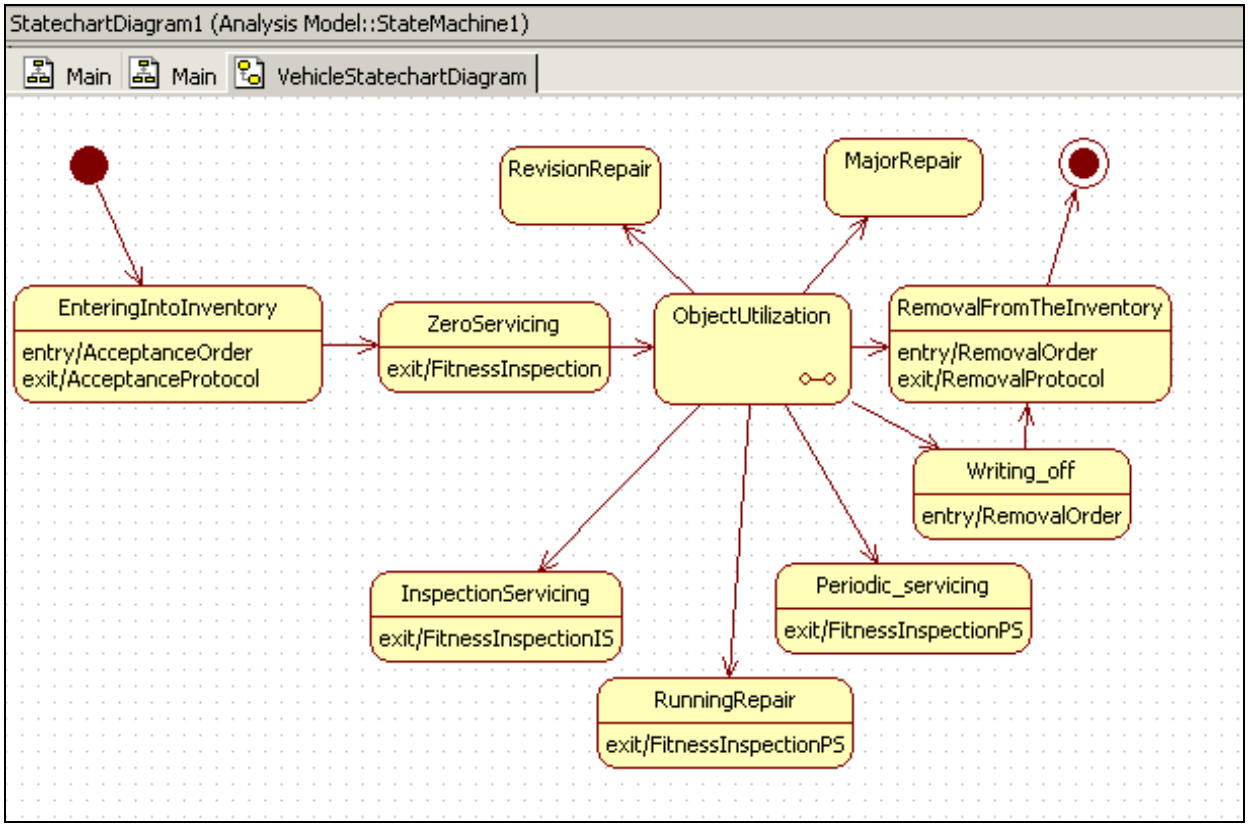

Fig. 4 Statechart Diagram of Vehicle

As a result of entering data by a user (or data obtained independently by a system from other external systems) changes occur in its elements. Generally, individual elements are characterized by certain parameters which allow us to classify them into distinguishable groups. Group affiliation changes depending on activities performed on the system elements. Hence we can say that individual elements are in subsequent states and their changeability (state transition) is illustrated by the State Machine Diagram.

In a rectangle illustrating the state, apart from its name, an activity which is to be performed in an initial state can be added as well as an activity to be performed on exit from a given state. In the case of more extended state diagrams, they can contain sub-states as it is depicted in Figure 5. Basically, the state “Periodic Inspection” takes place when a use-cycle prescribed by regulations for a given series of vehicles is exceeded (this is a condition of entering the state).

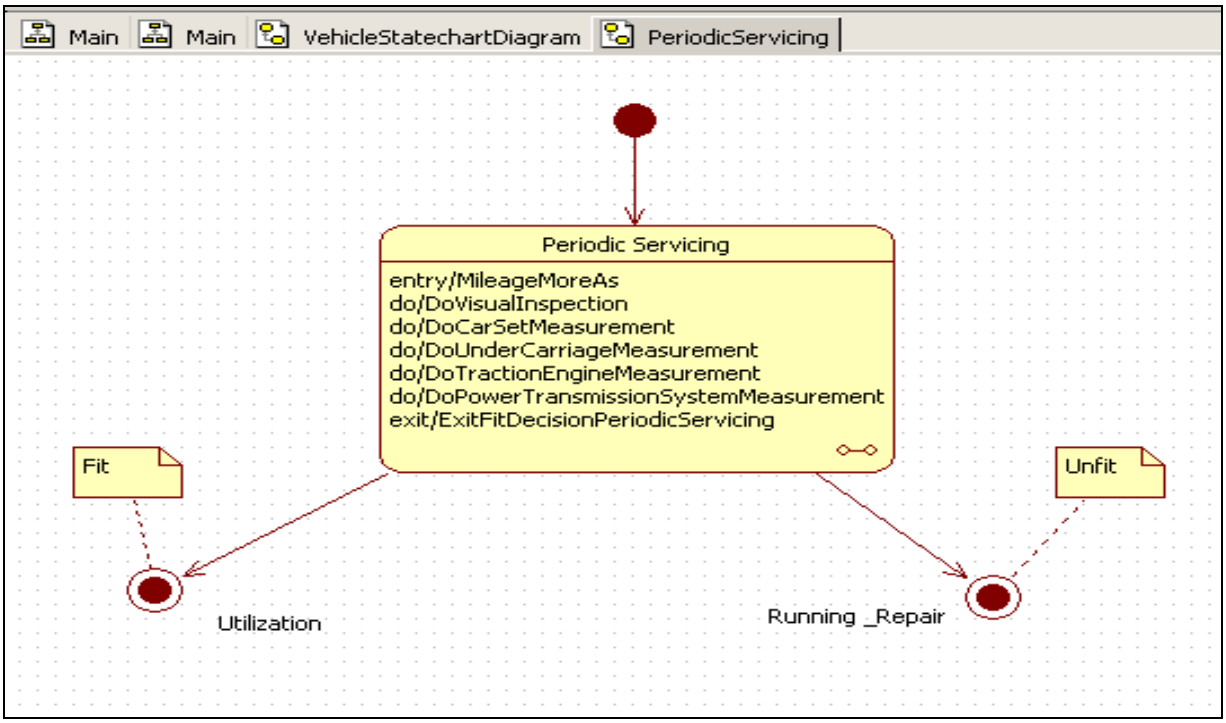

Fig. 5. Diagram of the Periodic_Inspection sub-state

Subsequent methods, such as DoOgledzinyZewn (external inspection), DoPomiaryZK (wheel set measurements), DoPomiarPodwozia (undercarriage measurements), DoPomiarsilnikaTrakc (rail traction engine measurements), DoPomiarUklNapedSS (Diesel engine power transmission system measurement/Diesel engine vehicle) are performed in order to carry out the superior state. In practice they can include:

- records of performing measurement activities on the object,
- methods evaluating the state of particular sub-assemblies on the basis of the system contained data,
- tasks to be performed by external systems, e.g. measurements and inspection.

Depending on whether the method can be implemented immediately or its result will be specified in the future, a decision if a vehicle or object is fit or unfit for further use determines its leaving the final state.

For the sake of example – the method "Measurement of the Diesel Engine Power Transmission System" is a separate task consisting of a number of component procedures covering such measurements as:

- external characteristic at load state U=f(I) of the main generator ,
- characteristics at no-load state but at different rotational speeds,
- dynamic characteristics of the power unit,
- elementary fuel consumption of a diesel engine at characteristic operation points,
- fuel consumption per hour at nominal load.

From the above set of tasks which are to be performed one can conclude that obtaining a final result may take even several hours. Consequently, remaining in the Periodic Inspection State can be equally long. Also the Running Repair State may take several days. In such a situation it is important to take into account time interdependencies between events occurring in the system. Sequence diagrams suit this end perfectly well.

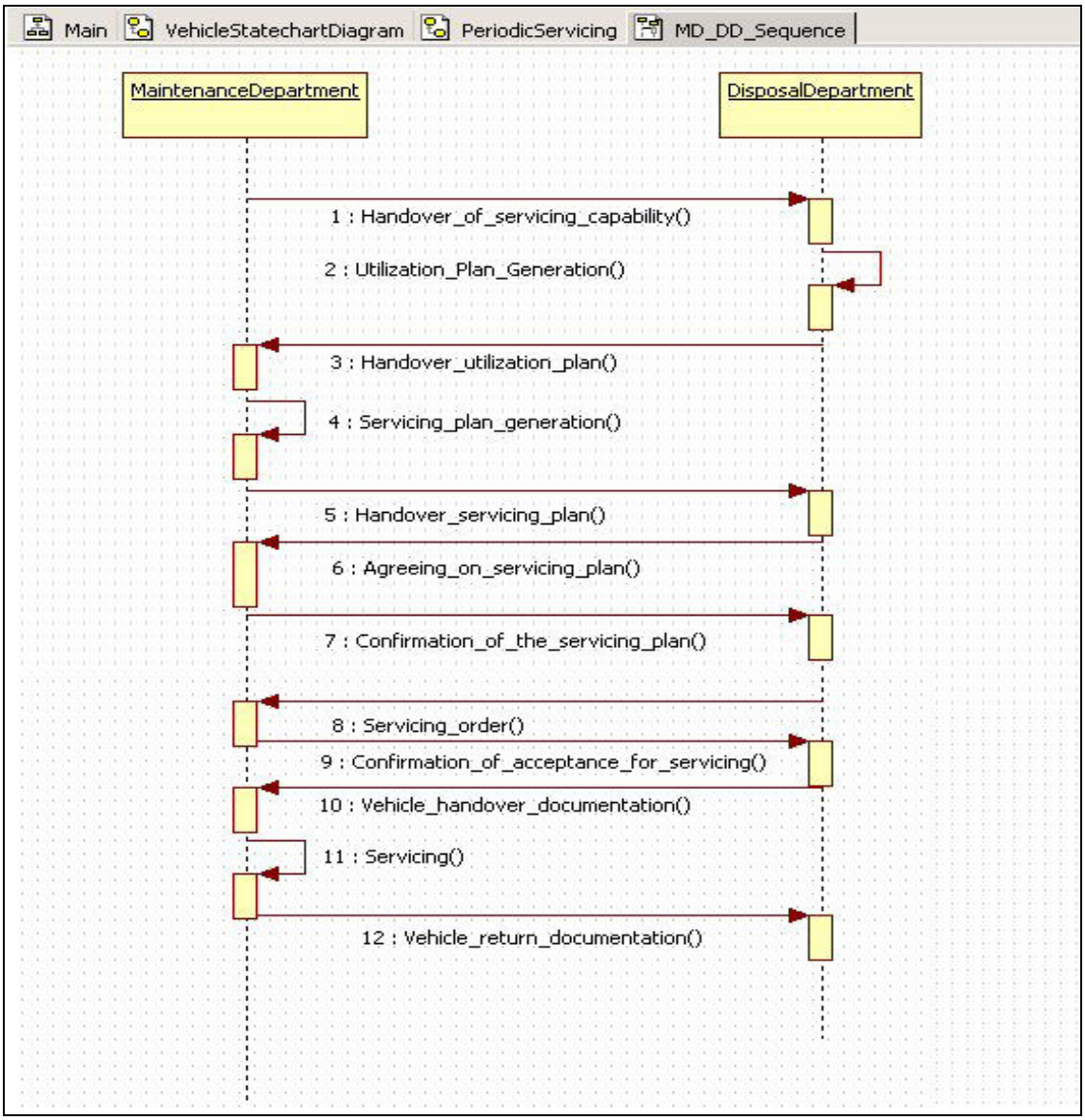

Fig.6 Diagram of communication sequences in the course of a vehicle handover for repair

On the one hand, CAMS should illustrate activities of the Repair Plant and on the other of the Department responsible for the vehicle use. Therefore it seems reasonable to divide the system into two sub-systems: one deals with the Rolling Stock Disposal (DD) and the rolling stock use whereas the other deals with maintenance activities (MD). What combines the two sub-systems are Service Orders on behalf of the Disposal Department and confirmation of service performance and information about the readiness to perform the service on behalf of the Maintenance Department. As services must be performed within a specific time therefore a superior maintenance plan covering time-schedules of the rolling stock use and servicing is necessary. Time interdependencies related to these activities are depicted in Figure 5.

4. Summary

The UML 2.0 specification (ISO/IEC 19501 standard) presented as a model by the Object Management Group, an organization assembling creators of object-oriented methods, includes many other types of diagrams. They allow a model to consider the less common aspects of system operation, e.g. component, collaboration or implementation diagrams. The initial tendency to

create separate object-oriented methods depending on application [1] was driven out by the UML language which was originally devised as a tool of modeling and software documentation but it also turned out to be an excellent tool for modeling maintenance and business processes. A further consequence of using the models of the computer-aided maintenance systems seems to be development of design models of such systems which are more universal than the currently existing commercial versions.

References

[1]- Booch G., Rumbaugh J., Jacobson I., Język UML. Przewodnik użytkownika, Seria: Inżynieria oprogramowania, Warszawa, WNT, 2002.

[2]- Erd Andrzej – Elements of the UML Model of the Rail Vehicle Maintenance System – Journal of POLISH CIMAC vol. 4, Gdansk University of Technology, Polish Academy of Science, Transport and Mechanical Engineering Committee's Gdansk 2009.

[3]- Erd Andrzej, Szychta Leszek – Model systemu eksploatacji uwarunkowany diagnostycznie. Unpublished work. Politechnika Radomska, Radom 2010.

[4]- Life Cycle Analysis Handbook - State of Alaska - Department of Education & Early Development 1st Edition - Education Support Services / Facilities 1999.

[5]- Miles Russ, Hamilton Kim – Learning UML 2.0 O'Reilly Media Inc. 2006.
Polish edition – Helion Gliwice 2007.

[6]- Shalloway A., Trott James R., Projektowanie zorientowane obiektowo. Wzorce projektowe. Gliwice, Helion, 2005.

[7]- Subieta Kazimierz- Obiektowość w projektowaniu systemów baz danych. Akademicka Oficyna Wydawnicza PLJ Warszawa 1998.

[8] - Yourdon Edward - Marsz ku klęsce. Wydawnictwa Naukowo Techniczne Warszawa 2007.