Journal of POLISH CIMAC
Faculty of Ocean Engineering & Ship Technology
GDAŃSK UNIVERSITY OF TECHNOLOGY

ELEMENTS OF THE UML MODEL OF THE RAIL VEHICLE MAINTENANCE SYSTEM

Andrzej Erd

Politechnika Radomska
Ul. Malczewskiego 29, 26-600 Radom, Poland
Tel. 483617723
e-mail:andrzej.erd@gmail.com

Abstract

The paper accounts for the construction of the rail vehicle maintenance model. It points to selected, possible to use, software tools applied for modelling in the Unified Modelling Language. Commonly accepted solutions are presented aiming to represent the maintenance system in UML. Such a model can be an input datum for the application generator in order to generate a code automatically. A more general application of this model is to create a design pattern to construct other maintenance assistance system of, e.g. ships or power station equipment.

Keywords: *system modelling, Unified Modelling Language, design pattern, maintenance, diagnostics systems, Maintenance systems*

1. Introduction – purpose of system modelling

Looking at the development of computer systems from the historical perspective one can notice a significantly delayed development of software in comparison to hardware. Hardware development was often accomplished abruptly. New series of microprocessors were introduced, new computer architectures appeared which led to significantly improved results. New memory types were constructed expanding the data storage capacity and decreasing the time of access to them.

Simultaneously, the progress of less spectacular nature, within the framework of existing technology, occurred. The path width within integrated circuits was reduced, which allowed for more data packing. The rotational speed of hard discs was increased, the transfer speed within network interfaces was accelerated.

Looking in the same way at the development of software it is much more difficult to see such breakthroughs. Undoubtedly the subsequent steps which affected the way in which software was created were the stages in the development of methodology, first – structural one and then – object-oriented one. Although the foundations of the latter were presented in the 1980s, object-oriented programming is still the dominating way of presenting reality in the information systems. The reason for such a state of affairs is the immense complexity of systems [2].

The sources of system complexity are rooted in:

The problem domain – a great number of interdependent factors occurring in the real world.

Design technologies – necessity to reproduce the real world in the virtual world

Design team - a group of people of limited abilities of perception, information transfer and information throughput.
Programming technologies – a vast number of standards to be observed, a wide range of required skills
Users – necessity of transition from the virtual world to the real world and interpretation problems related to this. Limited abilities and limited knowledge of the system.

The ways to limit the problems of complexity are as follows:
1. Wider usage of computers in system design and creation.
2. Development of new methodologies of analysis and development.
3. Data and system modelling.
4. Creating new generations of programming languages
5. Construction of tools automatically generating the object code on the basis of the model.
6. Creating model patterns.

This paper is a contribution to points 3 and 6, i.e. with the help of modelling tools an attempt was undertaken to reproduce the maintenance system elements of the rail track vehicles, especially those with the combustion engine. Similar problems appear in the maintenance of other complex objects, such as ships or power stations.

2. Applied tools

In the simplest case it is enough to have paper and a pencil to create a concept of the system model. However, relatively early the programming tools were developed to enable simplification of editing activities by means of objects inserted into text editors. A particularly convenient and popular tool is provided by the universal VISIO package (Microsoft) which in its menu has the *Software* option and within it, among other techniques of data representation, the sub-option *UML system*. It contains symbols typical for the most important types of diagrams, i.e.:
UML Static Structure
UML Collaboration
UML Deployment
UML Statechart
UML Sequence
UML Use case
Some programmes for system modelling allow for automatic generation of the software code. One can mention here Visual Paradigm UML (Visual Paradigm). A possibility of importing diagrams coming from VISIO is its interesting feature. Contrary to VISIO, this system can check automatically the object code integrity and generation on the basis of created diagrams.
This system is a transitory product for a typical CASE (Computer Aided Software Engineering) software, i.e. sets of tools for analysis, design, modelling and programming.
Such packages are most often paid and the best known among them include:
IBM Rational Rose is one of the oldest CASE solutions taken over by IBM. Apart from the tools for visual modelling, it contains the functions of code generation in languages such as Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ and Visual Basic.
Enterprise Architect – a tool running under the control of Windows and Linux systems.
Sybase Power Designer – a solution of the leading manufacturer of the systems of database management
Visual Paradigm SDE – an UML modelling tool and which later transfers the model into a code in the C# or Java languages to the Eclipse platform.
Apart from paid tools there is also a great number of programmes based on the GNU licence:

Argo UML – enables generation of the code based on the Eclipse platform.
Atlas Transformation Language – allows for a transformation of the UML model to Java code
ESS-Mode – a tool for reverse engineering of the UML model from Delhi or Java code
Star UML – a simple UML diagram editor (at the moment one must pay for it, but there is a possibility of obtaining of a full value, time restricted trial version after registration on the producer's home page).

3. System modelling

The system construction consists of the determination of:

- information elements allowing for a full description, also static one (database construction)
- outcome information expected by the user (User's interface)
- the way of input data processing into output data (application logic)

Thus the problems are naturally divided into 3 layers, i.e. data access, logic and presentation.
It is reflected in the latest design trends as the so called three-layer architecture. The essence of layer structure consists in such a construction of applications as to make the functions of individual layers independent from one another and to prevent the changes within each of them from affecting the remaining ones.

Tab. 1: Layers of system architecture

Layer of	**Function**
Presentation (User's interface)	Receiving user's commands, Entering data, deriving outcomes
Logic	Correct application logic
Data access	Maintaining source data, communication with external bases, reception of external communiqués

Principally, the basic element of a model are classes representing its components. The class possesses attributes (parameter values) and methods, which operate on these attributes. An example of attributes of the class of locomotive data is shown in Table 2..

Tab. 2. Attributes of the Locomotive Class

```
Class _ Locomotive Data

   Series                              : SeriesType;
LocomotiveNumber             : LocomotiveTypeNumber;
LocomotiveAcceptanceDate : word;
AcceptanceOrderNumber        : string20;
AcceptanceOrderDate             : word;
ConstructionYear                    : word;
IntroInServiceDate                          : word;      Warranty        : Word;
LastPeriodicRepairSymbol      : string3;    ZNTKCode      : Word;
LastRepairCourse                  : longint;    CurrentState    : string[3];
LastInspectionSymbol            : string3;    SBData            : word;
LastInspectionDate            : word;      SBChange        : word
LastInspectionCourse                        : longint;
```

Component assemblies are described in a similar way although in their case there are more attributes used to describe the measured values. The assemblies are aggregated and belong to the locomotive, which is illustrated in Figure 1.

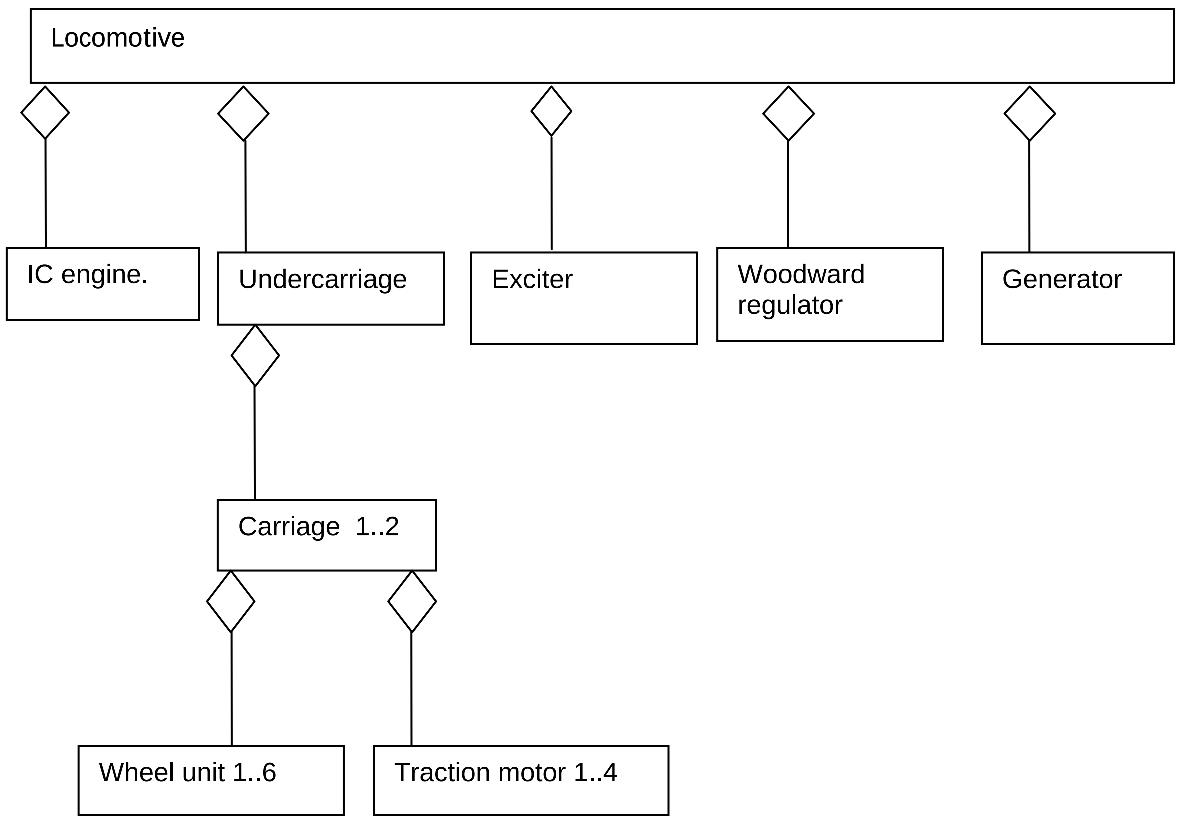

Fig. 1. Aggregation of component classes into the Locomotive class

Classes can describe physical elements but also can constitute a description of measurement results, e.g. the class of oil-water analyses presented in Table 3 below.

Tab.3 Attributes of the Oil-Water Analyses Class

Class_ OilWaterAnalysesData

Series : SeriesType;
LocomotiveNumber : LocomotiveNumberType;
InspectionRepairDate : word;
InspectionRepairType : string3;
OilContamination : real;
OilViscosity : real;
WaterContent : real;
AlkalineReserve : real;
OilAssessment : char;
OilContent : real;
OilToWaterRatio : real;
OilType : string10;
WaterAssessment : char;
IntroCode : word;
Comment : string20;
NotifyingPerson : string20

The range of data indispensable to describe the system is very broad and Table 4 below presents only examples of further elements of this type without a detailed description of attributes.

Table 4: Range of static data of the diagnostic system and vehicle maintenance(examples)

Constant data	
	Data of the railway rolling stock dispatching unit
	Data of the repair plants (address , repair plant Number)
	Number of a vehicles in stock
	Configuration data of the vehicle series (assembly types and their numbers)
	Configuration data of vehicles (what types of assemblies are components)
	Configuration data of assembly types (description of parameters – name, number, unit)
	Configuration data of analysis (what is included in analysis – parameter, scope)
	Model characteristics of engine and power transmission system measured at the test stand

Measurement data	
	Measured characteristic of the power transmission systems
	Measurement results of oil consumption
	Measurement values of wheel sets
	Measurement values of traction motors
	Measurement values of the main generator
	Analyses results

Table 4: Range of static data of the diagnostic system and vehicle maintenance(cont.)

Variables	
	Register of vehicles within the railway rolling stock in the plant
	Composition of a vehicle unit – configuration of elements – actual specimens identified by registration numbers

Principally, these data can be divided into 3 groups:

- Non-changeable (constant) data – dictionary-wise, they are characterised by stability or the changes are very rare.
- Measurement data – results of diagnostic inspections; their registration and analysis is the basis of making decisions about replacements and qualifying vehicles for repair. Individual results can be in the form of single numbers but they can also be records of more complexity, e.g. sets of data from the measurements of wheel sets or characteristics of a power transmission system measured at the test stand.
- Changeable data (variables) – illustrating the turnover of vehicles in the register and also, which is even more intense, the turnover of assemblies within vehicles (replacements)

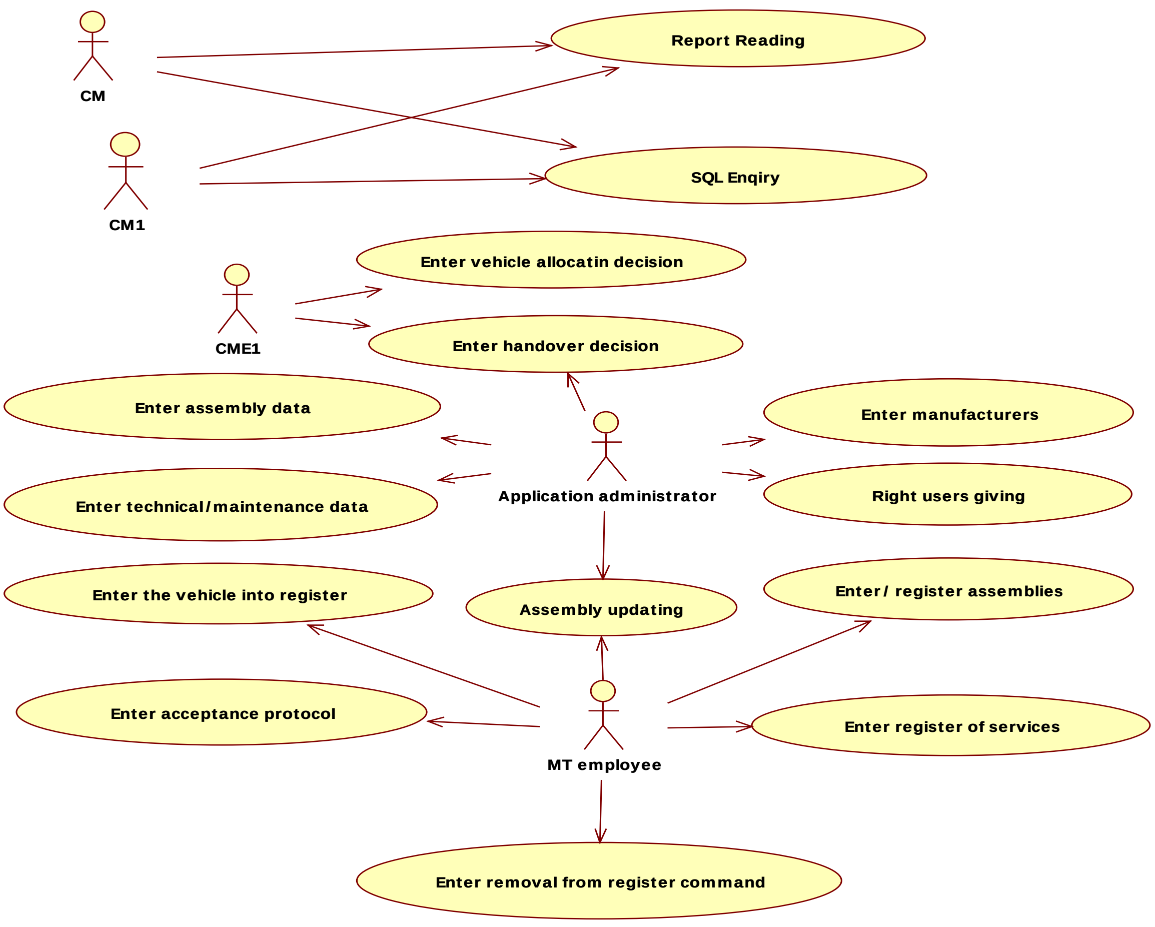
CM
CM1
Report Reading
SQL Enqiry
CME1
Enter vehicle allocatin decision
Enter handover decision
Enter assembly data
Enter manufacturers
Application administrator
Enter technical/maintenance data
Right users giving
Enter the vehicle into register
Assembly updating
Enter/ register assemblies
Enter acceptance protocol
MT employee
Enter register of services
Enter removal from register command

Fig. 2 Use case diagram (limited)
CM- Directors. CME- Departament of Traction and rolling railway stock maintenance;
MT- Railway Rolling Stock Plant (used StarUML)

The scope and degree of data processing depends on the information recipient and its intended use. In order to facilitate a further analysis of the real system, the diagrams of use cases are created. A fragment of such a diagram is shown in Figure 2. The use case diagrams are particularly useful when it comes to specifications of the system element behaviour as seen from outside. Owing to them the sub-systems and classes become easier to understand. They allow to grasp who the information producers and recipients are and which thematic ranges of the information take part in individual actions. They facilitate testing of a ready system in the start-up phase.

A subsequent stage of system modelling is representation by means of change diagrams in the course of the individual actions implementation. While creating them, a lot of attention is given to the place and manner of data taking and the algorithm for data processing. Thus, it can be assumed that these charts are included in the second layer of application, i.e. the layer of logic.

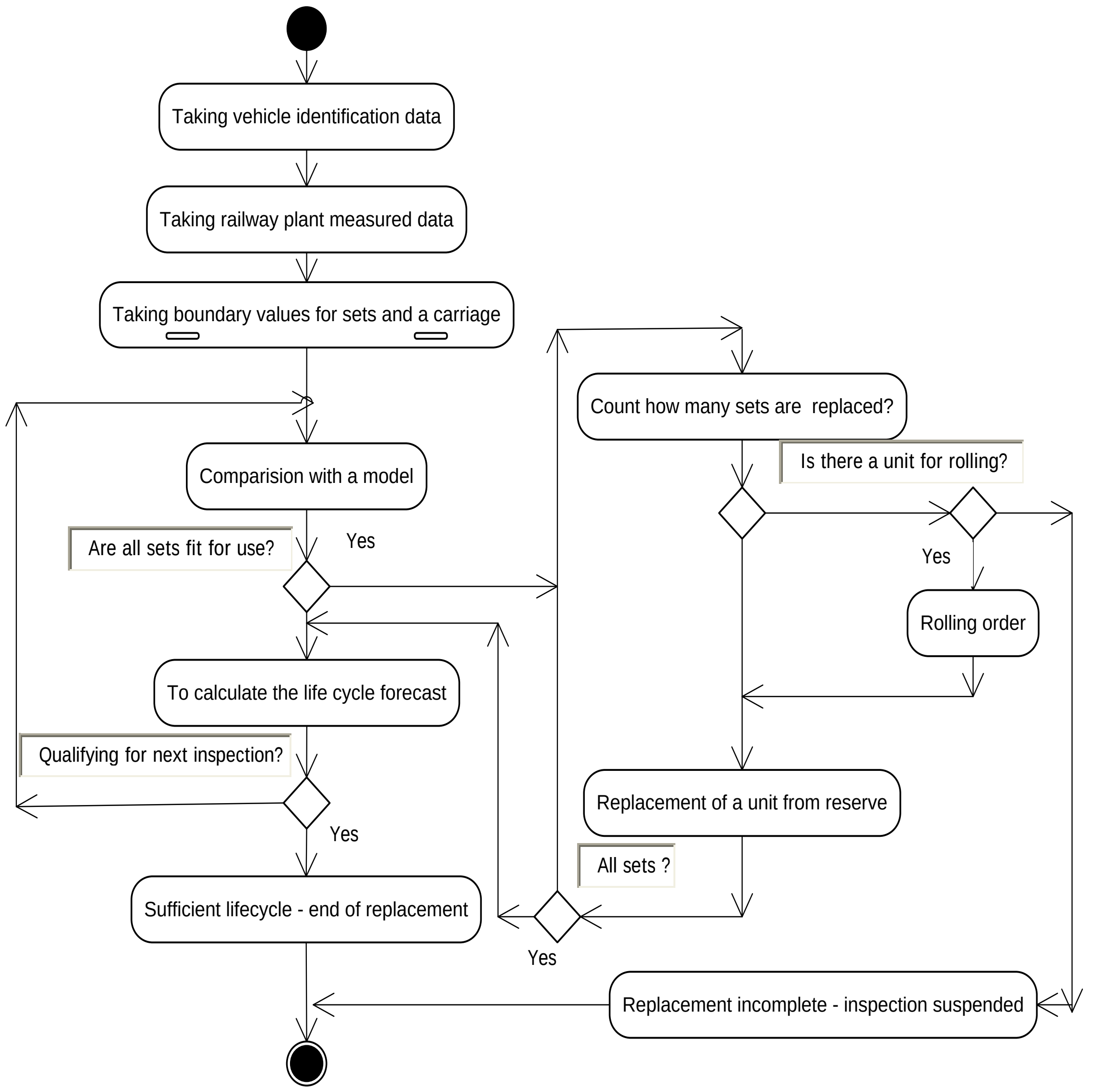
Taking vehicle identification data
Taking railway plant measured data
Taking boundary values for sets and a carriage
Count how many sets are replaced?
Is there a unit for rolling?
Comparision with a model
Are all sets fit for use?
Yes
Yes
Rolling order
To calculate the life cycle forecast
Qualifying for next inspection?
Replacement of a unit from reserve
Yes
All sets ?
Sufficient lifecycle - end of replacement
Yes
Replacement incomplete - inspection suspended

Fig. 3 Change diagram for replacement of wheel sets – parts of use case – Updating of sub-assemblies.

Basically, the state chart (change diagram) is so detailed that by following it, one can create the processing procedures implementing specific tasks. If manual coding is forecast, individual SA blocks are usually implemented in the language of the third generation. In the case of automatic generation it is necessary to write out further algorithm blocks up to the level of single variables.

4. Effects

The model developed is the starting point for creating a model design of the maintenance assistance system. Having it will allow for significant shortening of the period of creating computer systems meant for diagnosis and maintenance.

The essence of a design pattern (model) is the existence of an application skeleton which could be used in a possibly broad range of applications. The role of analysts, designers and programmers is to adjust the model to user's needs. The UML model is a tool to show the interdependencies between the system elements (static part) and its operations (dynamics). In the traditional approach the „manual coding” of the entire designed system is necessary. It is then followed by mundane testing. A tremendous amount of work from programmers is required to re-build the existing solution, if there is one. Technology of the 4th-generation languages, which is supposed to reduce the coding process, may turn out to be extremely useful provided one has suitable models. In general, the human work outlays are shifted from coding to a better reproduction of reality.

References

[1] Boch Grady, Rumbaugh James, Jacobson Ivar, UML przewodnik użytkownika. WNT Warszawa 2002

[2] Subieta Kazimierz. Obiektowość w projektowaniu systemów bazach danych. Akademicka Oficyna Wydawnicza PLJ. Warszawa1998

[3] IBM developerWorks Live – IBM materials. Warszawa. 2004.

[4] IBM Software Evaluation Kit – IBM electronic materials Warszawa 2004.

[5] Rational Software Atlantic Launch IBM materials, Warszawa 2006.