Proc. of 10th IEEE International Conference on Methods and Models in Automation and Robotics, vol.2, 2004, pp.1131-1136

IMPORTANCE OF EARLY DESIGN PHASE IN MECHATRONIC DESIGN

ZBIGNIEW MROZEK

Cracow University of Technology, Warszawska 24, PL 31-155 Krakow, Poland
zbigniew.mrozek@pk.edu.pl, http://www.cyf-kr.edu.pl/~pemrozek

Abstract: Mechatronic approach helps to achieve synergy and high ratio of performance to cost index. Author uses software engineering tools and RUP methodology to manage the design process. UML notation is used as common language during design. It helps to integrate subsystems of different nature and to eliminate main risks of project failure. Implementation and detailed design is not described in the paper. Examples of UML diagrams are enclosed.

Key Words: mechatronics, RUP, UML,

1 INTRODUCTION

Mechatronic is synergistic combination of mechanical engineering, electronics, control systems, computers and software [1, 22]. Other disciplines are used in design, if needed. Mechatronic products are often better than their mechanical equivalents. For many years, interdisciplinary products could not be well optimised as they were designed without knowledge of its mechatronic properties.

In mechatronic approach, all subsystems should be fully integrated early in the design process and they should be treated as equally important during optimisation, irrespective of their physical nature. This is not easy, as specialized modelling and simulation tools are effective only in their own area, and as they are not suitable for concurrent engineering of the whole system [20-23]. Using RUP (*rational unified process*) methodology and UML *(unified modelling language)* notation, it is an important step towards unification of models, early integration of subsystems and concurrent design of the whole system.

UML models describe future system architecture and behaviour on high level of abstraction, without need to decide on physical nature of subsystems and its details. Using presented tools and methodology, designer may identify and eliminate, early in design process, main risks of project failure. It is important that designer has possibility to verify his ideas using virtual UML models, without building physical prototype or even without detailed simulation model.

2 DESIGN IS AN IMPORTANT PART OF THE PRODUCT LIFE TIME

A typical design process [9,10] may be presented as sequence of:

- early design phase, which include requirements elicitation (inception) and conceptual design (elaboration) with off-line simulation and testing
- building (construction phase) which include detailed design, HiL simulation and prototyping, implementation and testing of the final design

If design is successful, the product life cycle may include:

- production and distribution (transition)
- exploitation and maintenance.
- recycling of used product at the end of its life cycle.

Design is not strictly sequential but iterative. Parts of design phases are repeated several times in small loops (design cycles or micro-cycles), during iterative design. If result of any step of design is not satisfactory – the designer should return to one of previous steps, as shown on figure 2.

3 RATIONAL UNIFIED PROCESS

Rational Unified Process (RUP) is a software engineering methodology based on best practises, learned from thousands of successful projects [17,18]. Some parts of this methodology are useful in mechatronic design.

There are four development phases in RUP: inception, elaboration, construction and transition. There are also well defined conditions to be fulfilled (milestones), when new phase of design is started. **Inception** corresponds well with requirements elicitation of early design phase described earlier. **Elaboration** is focussed on analysis the problem domain, establishing future system architecture and elimination of the main risk of project failure. It extends conceptual design phase. **Construction** phase may include prototyping, detailed design and implementation. **Transition** means production and deployment of the product.

An important difference between RUP and methodology presented in this paper is possibility of going back to any previous step of design (some conditions may apply), even if it is beyond already achieved milestone.

3.1 Spirit and best practices of RUP

In author's opinion, best practices [17], "spirit" and essentials of RUP [15] may be concluded in few following guidelines useful in mechatronic design:

1. Manage requirements and accommodate changes early in the project
2. Identify major risks risk and attack them early, or they'll attack you
3. Use component-based architectures
4. Model the system visually
5. Develop iteratively, make quality a way of life, not an afterthought
6. Work closely together as one team
7. Ensure that your deliver value to your customer

Although the nature of mechatronics is much more general than software engineering, the above guidelines were found to be very effective,

3.2 Design with UML language

Inventors of RUP were motivated to create notation for their unified methodology. They invented UML, a language for specifying, visualising, constructing and documenting the artefacts of software systems. From version 1.1, UML language is non-proprietary and open to all. It is maintained by the standards organisation: *Object Management Group* (OMG). In mechatronic design, UML diagrams may be used for modelling on high level of abstraction [7-10, 12].

Many attempts were done to extend the software engineering methodology and UML modelling language in areas beyond informatics. McLaughlin and Moore [6] were probably the first to describe real time control process (conveyor belt transport subsystem) using UML class diagram in year 1998.

UML notation helps to describe and understand functions, services and activities of any system, regardless of its physical nature. This is very useful during all design phases. Models are essential for communication between members of interdisciplinary team of designers. UML helps to find and correct errors and omissions in specification of requirements on very early phase of design.

Designing UML diagrams on computer screen is supported with CASE software. Best-known packages are Rational Rose [16], System Vision and Real-time Studio [19]. Some CASE tools offer simulation and animation of UML models. This helps to see behaviour of the system under design. Simulation is even more realistic if virtual console with animated dials and gauges is prepared.

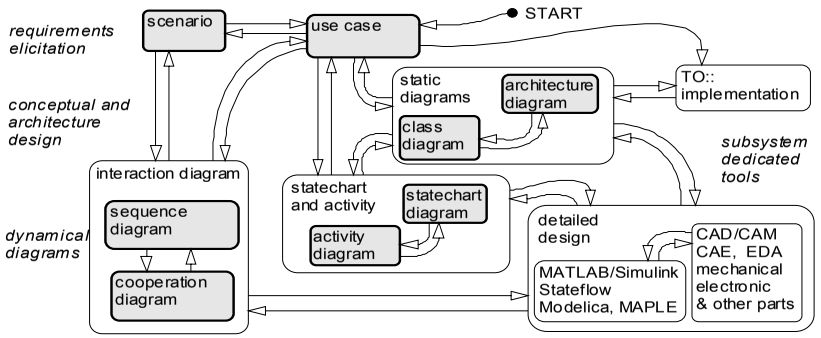

Figure 1 Many different UML diagrams are prepared during design of mechatronic system,

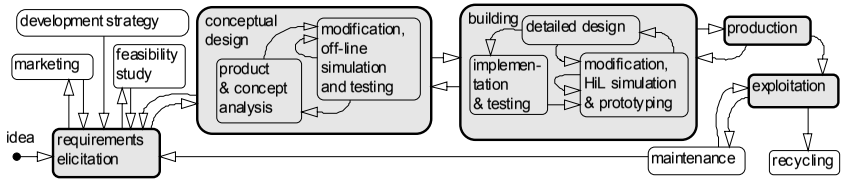

Fig. 2 Designing is an iterative process, an important part of product life time.

There is no need to use all possible types of diagrams during design (figure 1). It is sufficient to have a deep knowledge of a small subset of carefully chosen diagrams. As Brugge [2] points out; "You can model 80% of most problems by using about 20% UML". In the author's opinion, use case diagrams, scenarios, class diagrams, sequence diagrams and state diagram are most important. System architecture diagram (supported by RtS [19]) shows communication links between parts of system and is very well suited for designing mechatronic and control subsystems. Component and deployment UML diagrams are less useful in this area [8,10].

4 BEGINNING EARLY DESIGN PHASE

Design starts when vision of new or improved product came into sight. Next step is transformation of vision into specification of requirements, followed by conceptual design (figure 1). The main result of early design phase is to ensure that the architecture, requirements and plans for detailed design are stable enough and that main risk of project failure is sufficiently mitigated [17]. It is important as time delay and costs of modification are much higher, if done later, during detailed design, implementation or final design tests [10, 20-23].

During requirements elicitation phase, borders and external behaviour of new product or system are defined. and criteria of consumer satisfaction are set. This may be influenced with development strategy of producer. During conceptual design, feasibility study, estimation of needed resources and business plan for product development (eventually including production and distribution costs) is estimated.

4.1 Elicitation of requirements:

The goal of elicitation of requirements is to describe what the system should do and (which seems to be equally important) to agree with customer on this description [17]. This is an important job, as original textual problem description may be incomplete; some requirements may conflict with others. Even meaning of the same phrase may be different to different people. Blaming the client for a defective problem statement is not acceptable.

The main tools used for requirements elicitation are use cases and scenarios. Use case diagram describes behaviour of the system and shows how it interacts with external actors. Actor is a human user, another system, sensor or anything located outside of the actual system, which will interact with the system. Defining actors is essential to set the border between the system under development and its external environment. Actor is depicted as a simple icon of a man.

Scenario (see fig. 5) is a textual description or set of messages in natural language, describing the sequence of actor and system interactions. It describes details of use case functionality. Use case captures subsystem functionality as seen from the point of view of end user or domain expert. It helps to understand how the system should work. Use case is represented by an ellipse on use case diagram. An example of use case diagram for RoboCop football player robot [12] is given on figure 3.

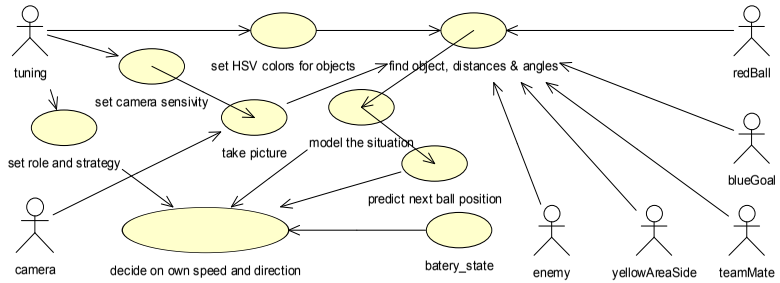

Figure 3. Use case diagram for football goal keeper [12]

5 CONCEPTUAL DESIGN

Conceptual design (elaboration in RUP terminology) is the most crucial part of design process. The objective is to establish a sound architectural foundation, to develop the project plan and to eliminate highest risk elements of the project [17]. Preliminary architecture is refined many times, as new UML diagrams are prepared and iteratively analysed, modified and verified.

5.1 Deciding on architecture of the system

At the beginning, one should analyse use cases and scenarios to identify objects, their responsibilities, activities and parameters. Similar objects are generalised into classes. This leads to preliminary version of class and object diagrams. Later objects and classes are used to build other diagrams.

An internal structure of the system is presented on class diagram. It shows classes and relationships that exist between them. Figure 4 shows an example of class diagram for automatic toaster [7]. Class diagram is redesigned when new class, new operation or new attribute is needed to build other diagrams. Name of class or object is given in upper compartment of rectangular icon, e.g. "**timer**". Its attributes (variables and parameters) are kept in middle compartment, e.g. "**time_to_release**". Operations (services and responsibilities of a class) are given in bottom compartment, e.g. "**start_timer**". Relations are shown using different lines.

It is not a good idea to try to design a complete class or object diagram before other diagrams are prepared. Instead, an **iterative approach** is used. Changes in object hierarchy, naming, operations and attributes are inevitable, when other diagrams are under design. This is especially true with sequence or state diagrams. If a good CASE tool is used, the same classes on different diagrams are synchronized and updated if class name, type or attribute is changed on any other diagram.

5.2 Verification of scenario and class diagram

Building UML diagrams automatically verifies specification of requirements and scenarios against omissions and inconsistencies Sequence and cooperation diagrams are graphical models of chosen scenario. They show what objects do, to implement chosen scenario. Missing classes and objects are defined at this stage. New operations and attributes needed for actual diagram are added into respective classes on class diagram. Those not used are considered for deletion. All changes are performed easily on computer screen. But designer should remember that any changes may affect other designers, using same class diagram.

Figure 5 shows how typical sequence diagram looks like. Actors and objects are shown as rectangular icons. Time flows down the vertical time lines. Object may send messages (shown as horizontal line with an arrow) to ask some services from another object (e.g. "**user**" asks "**toast_slider**" object to "**move_toast_inside**"), as described in scenario. Diagram may be annotated with text. Timing marks may be added to show exact time constrains.

Collaboration diagram (fig. 6) provide essentially the same information as sequence diagram. In absence of time lines, the sequence of messages is given by numbers.

Constrains (restrictions or rules applied to various elements of a model) may be included in use case diagram, sequence or in statechart diagram. This is especially useful in real time systems, where timing constraints for latency of messages and processing time limits for operations should be followed. OCL (*object constraint language* [13]), pseudocode, annotations or text (inside a special *notes* icon) can be used to show constrains in UML models.

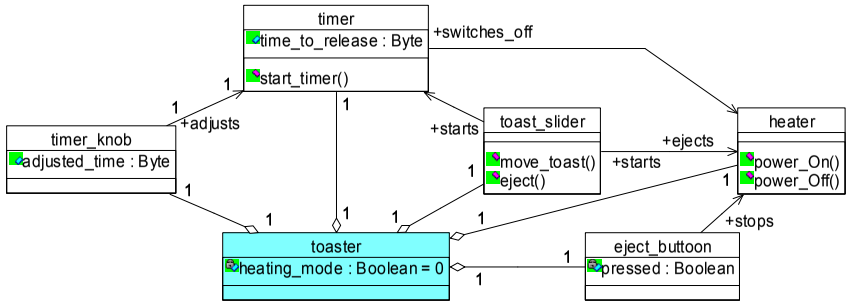

Figure 4. Static structure for automatic toaster is presented on class diagram [7]

Mr. Brown puts fresh toast into toaster. Toast goes down and electric heater is switched on. When heating time is elapsed, heater is switched off and toast jumps out of the toaster

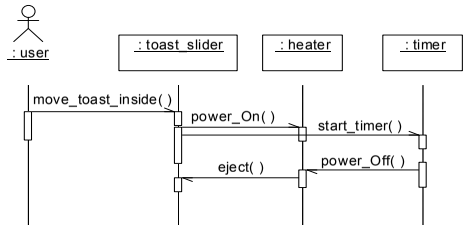

Figure 5. Scenario and sequence diagram for automatic toaster [7].

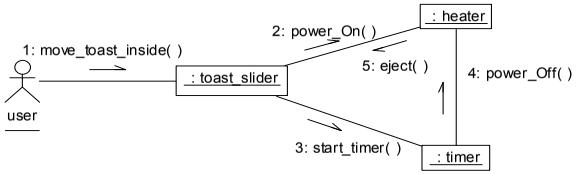

Figure 6. Collaboration diagram gives almost the same information as sequence diagram [7].

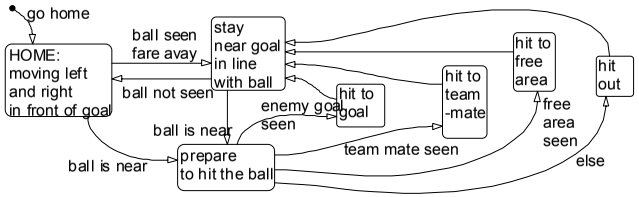

Figure 7 State diagram for goal keeper robot [12]

5.3 Verifying responsibility of objects and subsystems

Statechart diagram [8,9] is used to describe behaviour of the system or its part (subsystem, use case, object). Comparing with sequence diagram, the statechart shows all states the element may go through during its life, rather than states described by single scenario.

Each state represents a named condition during the life of an object. Object stays in actual state as long as it satisfies some condition or until it is fired by some event (or condition) to other state. A black ball is connected with a starting state. The end state (if exists) is shown as black ball in a circle. Lines with arrow (fig. 7) show possible transitions from one state to another one.

Building statechart diagram verifies functionality of whole system or its part. Statechart diagram presented on figure 7 was prepared in MATLAB environment with Stateflow [11] software. This diagram models behaviour of the goal keeper in different situations, including response to external events and fulfilled conditions [24].

If concurrent activities are needed, one may use an activity diagram. It is well suited to describe set of sequential and parallel actions as preparing the welding gun to work in MIG/MAG welding mode (fig. 8). Short horizontal or vertical thick lines are used for synchronisation of actions.

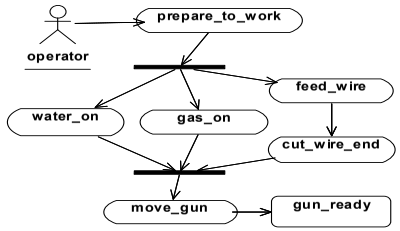

Figure 8. Activity diagram for preparation of welding gun to work in MIG/MAG mode [9]

6 CONCLUSIONS

RUP methodology and UML diagrams were invented for use in software engineering, but they may be efficiently used in design of mechatronic systems. Some of RUP best practices have already been intuitively used in mechatronics [20, 22, 23] without knowledge of this methodology.

UML models describe future system architecture and behaviour on high level of abstraction, without need to decide on physical nature of subsystems and its details. Preparation of diagrams automatically verifies specification of requirements and scenarios against omissions and inconsistencies. Ineffective architecture, wrong assumptions and errors in UML diagrams are also easily found. This helps to identify and eliminate (early in design process) main risks of project failure and improves future system quality.

Using commercially available CASE packages, UML may greatly improve productivity of design team by cutting down development time and improving final product quality (in accordance with ISO 9000 standards).

ACKNOWLEDGEMENTS:

Author wants to express his gratitude to Premium Technology Sp zoo (Poland), ONT (Kraków, Poland) Rational Software Corporation (USA) and The MathWorks Inc (USA) for free evaluation licenses for computer software presented in this paper.

7 REFERENCES

1. Bishop R.H., Ramasubramanian M.K. What is mechatronics?, [in:] Bishop RH (ed), The mechatronics handbook, CRC Press 2002, pp 1-1,1-10
2. Bruegge B, Dutoit A, Object-Oriented Software Engineering: Conquering Complex and Changing Systems, Prentice-Hall, 1999,
3. Dymola. Homepage: http://www.dynasim.se/
4. Harel D., Statecharts: A visual formalism for complex systems, Sci of Comp. Programming, 1987, No 8, pp 231-274.
5. Jacobson I., Booch G, and Rumbaugh G. The Unified Software Development Process Addison-Wesley, 1999.
6. McLaughlin M.J., Moore A., Real-Time Extensions to UML, Timing, concurrency, and hardware interfaces, Dr. Dobb's Journal December 1998
7. Mrozek Z, Computer aided design of mechatronic systems, International Journal of Applied Mathematics and Computer Science, vol 13 No 2, 2003, pp 255-267
8. Mrozek Z, UML as integration tool for design of the mechatronic system, Second Workshop on Robot Motion and Control, pp 189-194, ed. Kozlowski K, Galicki M, Tchoń K, Oct 18-20, 2001, Bukowy Dworek, Poland.
9. Mrozek Z, Methodology of using UML in mechatronic design (in Polish), pp.25-28, Pomiary Automatyka Kontrola 1, 2002.
10. Mrozek Z, *Computer aided design of mechatronic systems* (In Polish), Zeszyty Naukowe Politechniki Krakowskiej, seria Inżynieria Elektryczna i Komputerowa, nr 1, 2002
11. Mrozek B, Mrozek Z, *MATLAB i Simulink, poradnik użytkownika* Helion, Gliwice 2004
12. Mrozek Z, Tao Wang, Minrui Fei, *UML supported design of mechatronic system, Proc of Asian Simulation Conference/5-th Int. Conference on System Simulation and Scientific Computing*, Shanghai, China, Nov. 3-6, 2002.
13. OMG *Unified Modeling Language* Homepage http://www.omg.org/uml
14. Petko M. *Implementacja układów sterowania w układach ASIC/FPGA*, Pomiary Automatyka Kontrola Nr 1, pp.18-21, 2002,
15. Probasco L. *The Ten Essentials off RUP, the essence off an effective development process*, TP- 177 9/00, Rational Software Corporation, 2000 .
16. *Rational Rose, Rational Rose RT* Rational Software Corporation, http://www.rational.com/products/rose/
17. *Rational Unified Process, Best Practices for Software Development Teams*, Rational Software White Paper, TP026B, Rev 11/01,
18. *Rational Unified Process*, Homepage http://www.rational.com/products/rup/
19. *Real-time Studio*, ARTiSAN Software Tools, Inc. . Homepage: http://www.artisansw.com/,
20. Uhl T, Bojko T, Mrozek Z, Szwabowski W, Rapid prototyping of mechatronic systems, *Journal of Theoretical and Applied Mechanics,* pp.655-668, vol.38.3, 2000
21. Uhl T, Mrozek Z, Petko M Rapid control prototyping for flexible arm, [in:] Isermann R. [ed.], Mechatronic Systems, vol. 2, pp. 489-494, Elsevier Sci, Amsterdam, 2001
22. Uhl T, Bojko T, Mrozek Z, Petko M, Szwabowski W, Korendo Z, Bogacz M; *Selected problems of mechatronic design.* Techn. Report 7T07B 01414, (in Polish).Chair of Robotics and Machine Dynamics, Univ. of Science and Technology (AGH), Cracow,
23. Uhl T,Bojko T, Mrozek Z, Szwabowski W, *Rapid prototyping of mechatronic systems*, Journal of Theoretical and Applied Mechanics, pp.655-668, vol.38.3, 2000
24. Wei-Min Shen et al, *Building Integrated Mobile Robots for Soccer Competition*, Proc. IEEE Int. Conf. on Robotics & Automation, pp.2615-2618, Leuven, Belgium May 1998