AN EFFECTIVE GRAPHICAL APPROACH TO DEFINE OBJECTIVES AND STRUCTURE OF A CONTROL SYSTEM

Zbigniew Mrozek[1]

Cracow University of Technology (Politechnika Krakowska)
24 Warszawska Str., PL 31-155 KRAKOW, Poland,

Abstract: Author uses software engineering tools and RUP methodology to manage design of control for complex systems. UML diagrams are used to describe requirements and to model the system on high level of abstraction. Graphical approach with RUP and UML helps to integrate subsystems of different nature and to eliminate main risks of failure of the control subsystem project. Examples of different UML diagrams are enclosed. Implementation and detailed design is not described in the paper.
Copyright © 2005 IFAC

Keywords: Requirements analysis, RUP, UML, Modelling, Validation, Computer aided control system design (CASE).

1. INTRODUCTION

The paper is inspired by work of Nayfeh (2004), who designed an effective controller supporting work of crane operator. Regardless of crane type (rotary, overhead, etc.), the economic and safety constraints require that the payload be transferred in minimum time and with minimum pendulation while travelling. These objectives are in conflict, as transferring in minimum time requires large acceleration which induces large pendulation.

Crane operation is nonlinear and internally nonstationar due to purposeful and often change of payload and (even worst) due to change of pendulation frequency of payload, when cable length is changed during hoisting. As a result, linearized models are not exact. Additionally, during typical crane operation, the exact end-point coordinates of payload are not known in advance. Near landing area, operator looks for assistance of auxiliary staff member, who will guide him in precise docking of the payload. It is easy to see that classical approach to this challenging problem (without knowledge of end-point position) is not effective.

The main idea of Nayfeh approach is presented later. His result was almost complete damping of payload pendulation. He verified his result by computer simulation, then experiments in his own laboratory (with 1/24 scale model of 20 ton auxiliary crane ship known as T-ACS and finalised with industrial tests done in IHI Japan with 1/10 scale model of 65 ton container crane.

Nayfeh paper shows importance of client satisfaction based on deep understanding of real industrial problems. He gives 25 references in his work. Other references may be found in papers of Bartolini et al. (2002) and Szpytko and Smoczek (2004). Useful description of an embedded system for control and damping oscillations of flexible arm is described by Uhl at al. (2001)

This paper shows systematic and general approach to achievement of client satisfaction using methodology and tools from area of software engineering:

- RUP (*rational unified process*) methodology
- CASE (*computer aided system engineering*) tools based on UML (*unified modelling language*)

[1] *E-mail address:* pemrozek@cyf-kr.edu.pl
Home page: http://www.cyf-kr.edu.pl/~pemrozek

An example of graphical approach to define objectives and structure of a control system for the crane operation is presented in this paper.

2. METHODOLOGY AND MOTIVATION

UML models may describe architecture and behaviour of the original system and its future control subsystem. UML models have high level of abstraction and they are independent of physical nature and details of subsystems.

Designer has possibility to verify his ideas using UML models on high level of abstraction, without building physical prototype or even without detailed simulation model. He may identify and eliminate (early in design process) main risks of the project failure, delay or over-budgeting. The RUP approach forms a good base for future detailed design and prototyping of the control system. It is also an important step towards mechatronic approach to concurrent design and early integration of subsystems of different nature.

2.1. The design process

A typical design process may be presented as a sequence of:

- early design phase, which include requirements elicitation (inception) and conceptual design (elaboration). Off-line simulation may be used for testing of the design,
- building (construction) phase, which includes detailed design, HiL simulation and prototyping, as well as implementation and testing,
- deployment (transition) of achieved solution.

2.2. Rational Unified Process (RUP)

Planning, management and monitoring of team work may be done using Rational Unified Process (2001, 2004) or using any other effective methodology. RUP is the software engineering methodology based on best practises, learned from thousands of successful projects. Some parts of this methodology are useful in design of complex control systems.

There are four development phases defined in RUP: inception, elaboration, construction and transition. There are also well defined conditions (milestones) to be fulfilled, when new phase of design is started. **Inception** corresponds well with requirements elicitation of early design phase. **Elaboration** is focussed on analysis of 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. Figure 1 shows typical work effort of different disciplines (modelling, requirements, analysis, etc.) during each design phase.

Design is not strictly sequential but iterative, with many small design steps (micro-steps). After each step, the affected part of system is tested against requirements. Very often result of a step or micro-step is not satisfactory and the designer then returns back to one of previous steps and repeats part of the design. Iterative sequence of testing and redesigning may be repeated many times in a loop (figure 2) – until satisfactory result is achieved.

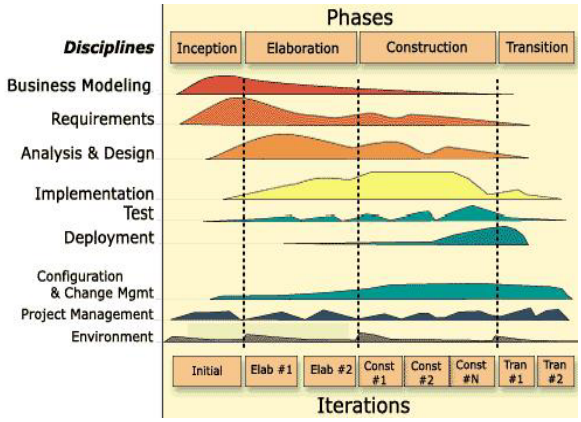

Fig. 1. Typical work effort during run of project design (Rational Unified Process, 2001).

2.3. The RUP methodology guidelines

In author's opinion, *best practices*, *spirit* and *essentials* of RUP (Probasco, 2003) may be concluded in few guidelines, useful in design of control systems:

1. Identify major risks and attack them early, or they'll attack you
2. Model the system visually
3. Develop iteratively, make quality a way of life, not an afterthought
4. Ensure that your deliver value to your customer

Although the designing of control system is much more general than software engineering, the above guidelines fit well to designing of control systems.

2.4. Using UML for modelling and inter-team communication

Inventors of RUP were motivated to create notation for their unified methodology (Jacobson at all, 1999). The result was UML, a language for specifying, visualising, constructing and documenting the artefacts. From version 1.1, UML language is non-proprietary and open to all. It is maintained by the standards organisation: *Object Management Group* (OMG, 2004)

Many attempts were done to extend the software engineering methodology and UML in areas beyond informatics. McLaughlin and Moore (1998) were probably the first to describe real time control process (conveyor belt transport subsystem) using UML-like class diagram. Now UML diagrams are

used for preparing different models on high level of abstraction (Mrozek, 2001-2004; Mrozek et al, 2002)

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-nary team of designers.

Designing UML diagrams on computer screen is supported with CASE software tools. Best-known packages are Rational Rose (2004), Rhapsody (2004) and Real-time Studio (2004). 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 shown on computer screen.

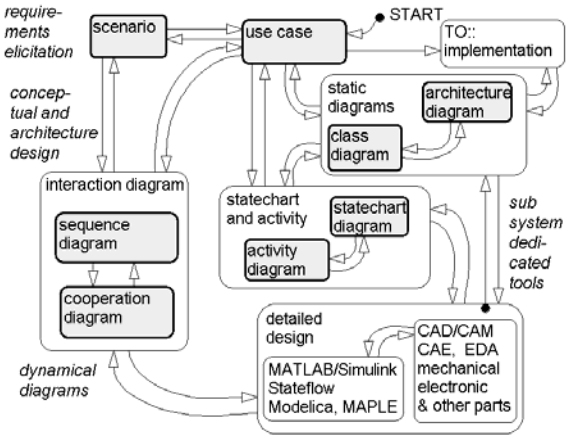

Fig. 2. Many different UML diagrams are prepared during system design.

As Brugge and Dutoit (1999) points out; it is sufficient to have a deep knowledge of a small subset of UML to use it ("You can model 80% of most problems by using about 20% UML"). There is no need to use all possible diagrams during design (figure 2). In author's opinion, use case diagrams, scenarios, class diagrams, sequence diagrams and state diagram are most important. Additional *system architecture diagram* (supported by Real-time Studio, 2004) may show communication links between parts of system and is very well suited for detailed design of the computerised control subsystems. Component and deployment UML diagrams are less useful in area of control.

3. AN EXAMPLE CONTROL PROBLEM

Control of a crane is investigated. One of successful strategy of crane control (Nayfeh, 2004) is to kill existing payload pendulation and to avoid exciting pendulation due to intentional movement of payload. It means, the payload movement derivatives (first and second or second only, all in 3D) should be always minimized with auxiliary excitation of crane, when payload reaches its zero-pendulation angle. And pendulation should be completly killed, when payload reaches the end point.

4. BEGINNING EARLY DESIGN PHASE

Design starts when need of new or improved solution came into sight. Next step is its transformation into specification of requirements, followed by conceptual design (figure 3).

4.1. Elicitation of requirements

During elicitation of requirements, borders and external behaviour of the system are defined and criteria of consumer satisfaction are set. This may be influenced with development strategy of a company.

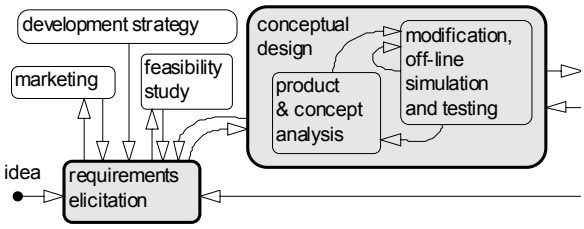

Fig. 3. Early design phase: elicitation of requirements and conceptual design (named in RUP as inception an elaboration).

4.2. Graphical representation of requirements on Use Case Diagram.

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. This is an important job, as original textual problem description may be incomplete or some requirements may conflict with others. Even meaning of the same phrase may be different to some members of design team. Blaming the client for a defective problem statement is not acceptable, as consumer satisfaction is one of main objectives of design.

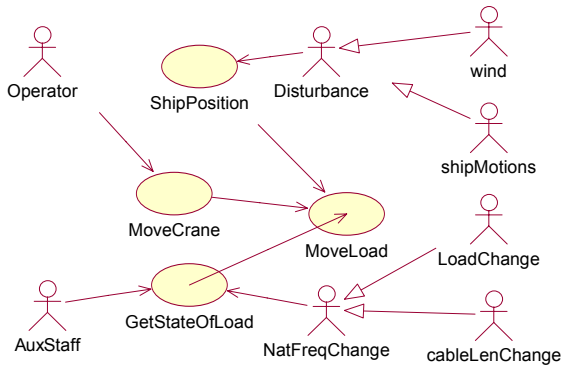

Fig. 4. Use case diagram for auxiliary crane ship.

Use case diagram (figure 4) shows actors, use cases and interactions between them. It describes how the system may be used by user and how the system interacts with other external actors.

Actor is a thing outside the actual system (human user, another system or external signal, connected to sensor or interface), which interacts with the actual system. Actor is depicted as a simple icon of a man.

Use cases are system boundaries identifying what the system should do. They capture subsystem functionality as seen from the point of view of end user or domain expert and help to understand how the system should work. Use case icon is an ellipse. Actors and use cases set the border between the system under development and its external environment It.

4.3. Describing actor and system interactions in scenario

Scenario 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. There are at least three actors in this scenario: Crane operator, auxiliary staff member and unknown external disturbances.

> *Crane operator has to move the payload to desired location. He starts with hoisting the payload about 3 m up and then moves it horizontally in desired direction. To save time, horizontal and vertical movement may be performed concurrently. Any pendulation of the payload should be damped. Near landing area operator looks for assistance of auxiliary staff member, who will guide him in precise docking of the payload.*

There are at least three actors in this scenario: Crane operator, auxiliary staff member and unknown external disturbances (compare fig. 4)

.

5. CONCEPTUAL DESIGN

Conceptual design (elaboration in RUP terminology) is the most crucial part of design process. During conceptual design, feasibility study, estimation of needed resources and business plan for development (including implementation costs) is estimated. The objective is to establish a sound architectural foundation, to develop the project plan and to eliminate highest risk elements of the project. Preliminary architecture is refined many times, as new UML diagrams are prepared and then iteratively analysed, modified and tested.

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 diagram. 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.

An example of class diagram for crane control subsystem is presented on figure 5. Name of class or object is given in upper compartment of rectangular icon, e.g. “Sensors”. Attributes (variables and parameters) are kept in the middle compartment. Operations (services and responsibilities of a class) are given in bottom compartment. Relations are shown using different lines, with or without arrows.

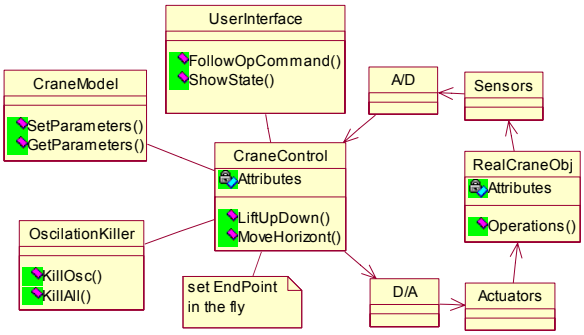

Fig. 5. Static structure of crane control subsystem is presented on class diagram

It is not a good idea to design complete diagrams sequentially. Instead, an iterative and concurrent approach is advised. Changes in object hierarchy, in naming, operations and attributes are inevitable, when other diagrams are under design. This is especially true when sequence or state diagram is prepared. If a good CASE tool is used, the same classes on different diagrams are synchronized and updated if class name, attribute, type or operation is intentionally changed on any other diagram. New classes, operations and attributes are added to respective class diagram – if needed for actually designed or any other diagram. Other (if not used} are considered for deletion. All changes are performed easily on computer screen; in framework of chosen CASE tool (e.g. Rational Rose).

5.2. Verification of requirements and system architecture

Building UML diagrams automatically verifies specification of requirements and scenarios against omissions and inconsistencies Scenario is verified with sequence diagram, which shows what objects does to implement this scenario. Sequence diagram is a graphical model of the scenario (figure 6).

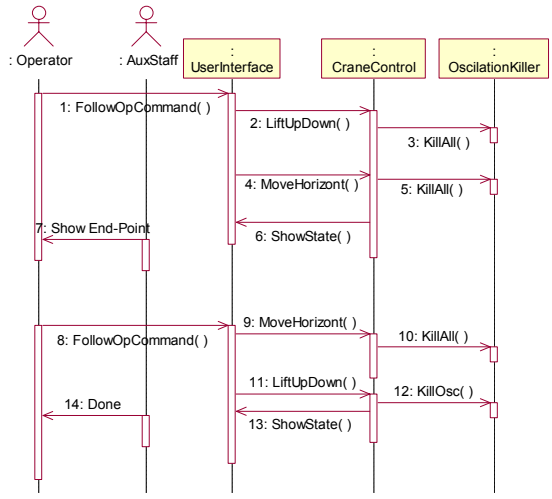

Fig. 6. Sequence diagram for crane operation

Actors and objects are shown on top of sequence diagram. Time flows down the vertical time lines. Object may send messages (horizontal line with an arrow) to ask needed services from another object (e.g. “UserInterface” may ask “CraneControl” to LiftUpDown(+3)” the payload), as described in scenario. Diagram may be annotated with text. Timing marks may be added to show exact time constrains.

Collaboration diagram provides essentially the same information as sequence diagram and is not described here.

6. VERIFYING RESPONSIBILITY OF OBJECTS AND SUBSYSTEMS

Statechart diagram of Harel (1987) is used to describe and verify behaviour of the system or its part (subsystem, use case, object). Comparing with sequence diagram, the statechart shows all states the system or its part may go through during its lifecycle, rather than states described by single scenario.

Each state represents a named condition during the life of an object (osc & hyperbolic on fig. 7). Object stays in actual state until it is fired by some event or when given condition (that must be fulfilled before the transition) will occur. Lines with arrow show possible transitions (change of state). A black ball shows a starting state. The end state (if exists) is shown as black ball in a circle.

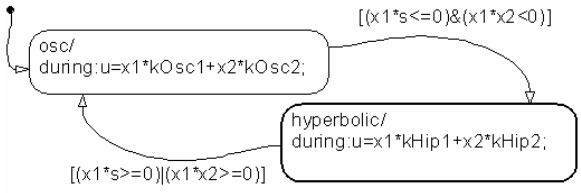

Fig. 7 State diagram for sliding mode controller (Mrozek and Tarasiewicz, 2001)

Statechart diagram presented on figure 7 models behaviour of the sliding mode controller, useful in hardly nonlinear and non-stationary environment (Mrozek and Tarasiewicz, 2001). It was prepared with Stateflow, in the MATLAB/Simulink software environment (Mrozek B, Mrozek Z. 2004). An important advantage this environment is possibility of simulation (of-line, on-line real-time, HiL: hardware in the loop) and prototyping. Later, simulated virtual model may be automatically compiled with MATLAB/RTW/StateflowCoder and loaded into FPGA or other target hardware.

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 (Mrozek, 2003).

7. CONCLUSIONS

RUP methodology and UML diagrams were invented for use in software engineering, but they may be efficiently used in design of any complex systems. Some of RUP best practices have already been used beyond this area, without knowledge of RUP methodology, e.g. incremental design is used in prototyping, see Uhl *at all*, (2001).

UML model 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. This helps to verify idea of future system, on very early phase of design. 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.

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 the project failure is sufficiently mitigated. This is important conclusion, as time delay and cost of modification are much higher, if corrections are made later, during detailed design, implementation or (it is the worst case) during final design tests.

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).

8. ACKNOWLEDGEMENTS:

Author wants to express his gratitude to Rational Software Corporation (USA) and The MathWorks Inc (USA) for free evaluation licenses for software presented in this paper.

More systematic description of RUP, UML and Stateflow may be found in cited literature

REFERENCES

Bartolini G., Pisano A., Usai E. (2002) Second-order sliding-mode control of container cranes *Automatica 38, pp 1783 – 1790.*

Bruegge B, Dutoit A, (1999). *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*, Prentice-Hall.

Harel D. (1987*).* Statecharts: A visual formalism for complex systems*, Sci of Comp. Programming, ,* No 8, pp 231-274.

Jacobson I., Booch G, and Rumbaug (1999). *The Unified Software Development Process.* Addison-Wesley.

McLaughlin J. and Moore A (1998) Real-Time Extensions to UML, Timing, concurrency, and hardware interfaces, *Dr. Dobb's Journal* December

Mrozek Z. (2001). UML as integration tool for design of the mechatronic system*,* In: *Second Workshop on Robot Motion and Control*, (Kozlowski K, Galicki M, Tchoń K (Ed)), pp 189-194, Bukowy Dworek, Poland.

Mrozek Z, Tarasiewicz S (2001) Attempting sliding mode controller to mobile robot arc welding process, *Proc. of III Krajowa Konf. Metody i Systemy Komputerowe* (Tadeusiewicz R., Ligęza A., Szymkat M. (Ed)), pp:369-373, Kraków.

Mrozek Z. (2002a). Methodology of using UML in mechatronic design [in: Polish], *Pomiary Automatyka Kontrola. No 1. pp.25-28,*

Mrozek Z. (2002b). Computer aided design of mechatronic systems [in: Polish], *Zeszyty Naukowe Politechniki Krakowskiej, seria Inżynieria Elektryczna i Komputerowa*, no 1, Kraków.

Mrozek Z, Tao Wang, Minrui Fei. (2002) 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,.

Mrozek Z. (2003). Computer aided design of mechatronic systems, *International Journal of Applied Mathematics and Computer Science*, vol **13** No 2, pp 255-267

Mrozek B, Mrozek Z. (2004) *MATLAB i Simulink, poradnik użytkownika* Helion, Gliwice.

Mrozek Z (2004). Importance of early design phase in mechatronic design. *In: Proceedings of 10th IEEE International Conference on Methods and Models* in *Automation and Robotics* (Domek S., Kaszynski R (Ed)), vol **1** pp 17-28, Miedzyzdroje, Poland.……..

Nayfeh A.H. (2004). A Smart Controller for Cranes. *In: Proceedings of 10th IEEE International Conference on Methods and Models* in *Automation and Robotics* (Domek S., Kaszynski R (Ed)), vol **1** pp 17-28, Miedzyzdroje, Poland.……..

OMG (2004) *Unified Modeling Language,* Homepage *http://www.omg.org/uml*

Probasco L.(2000) *The Ten Essentials off RUP, the essence off an effective development process, TP- 177 9/00, Rational Software Corporation,.*

Rational Rose (2004) *Rational Rose RT* Homepage*: http://www.rational.com/products/rose/ http://www-306.ibm.com/software/rational/*

Rational Unified Process (2001). *Best Practices for Software Development Teams,* Rational Software White Paper, TP026B, Rev 11/01,

Rational Unified Process (2004). *Homepage http://www.rational.com/products/rup/ http://www-306.ibm.com/software/rational/*

Real-time Studio (2004) *ARTiSAN Software Tools, Inc.* . Homepage*: http://www.artisansw.com/,*

Rhapsody (2004). *i-Logic, http://www.ilogix.com/*

Szpytko J. and Smoczek J. (2004), Fuzzy Logic Control Improve Transport Devices Quality, In: *Proceedings of 10th IEEE International Conference on Methods and Models in Automation and* Robotics (Domek S., Kaszynski R (Ed)), vol **2** pp 1391-1396, Miedzyzdroje, Poland.

Uhl T, Mrozek Z, Petko M (2001) Rapid control prototyping for flexible arm, *Mechatronic Systems* (Isermann R.(ed.)) *, vol. **2** , pp. 489-494,* Elsevier Sci, Amsterdam,