OVERVIEW OF DESIGNER 2000 PROJECT

BŮCHA JINDŘICH, KÁRNÝ MIROSLAV, NEDOMA PETR, BOHM JOSEF, ROJÍČEK JIŘÍ

Institute of Information Theory and Automation, Academy of Sciences, Pod vodárenskou věží 4. 182 08 Prague 8, Czech Republic, phone: +420-2-6605 2280, fax: +420-2-6884 903, bucha@utia.cas.cz

Abstract: The Designer 2000 project is briefly presented. Its objective is to support the usage of already existing tools for the design of adaptive systems. Bayesian approach to the design of LQG multidimensional controllers is used. In the project, there are identified requirements. OO CASE, UML, knowledge base and other CAD approaches are used to satisfy them. High attention is paid to communication with an ordinary user and some elements of this communication are mentioned.

Keywords: Adaptive control, Bayesian approach, Knowledge-based systems, Computer aided control systems design

1. INTRODUCTION

In this paper, the Designer 2000 is briefly presented. It is a project under development. The overall **project objective** is to *improve the accessibility and usage* of algorithmic and software *tools for the design of adaptive systems*[1]. The Designer 2000 (shortly Designer) supports a systematic probabilistic design of LQG (Linear models, Quadratic criterion, Gaussian disturbance) adaptive multidimensional controllers based on auto-regression models with external inputs (ARX). The project is based on the availability of necessary theory, algorithms, software and expertise. It is the continuation of a long-term development in the authors' Department, see Peterka [11], Kárný et al [6].

The **theoretical basis** of the project is Bayesian approach to the estimation and control [6, 11]. That gives a common point of view to these traditionally separated problems and is capable to respect their mutual influence. The probabilistic insight accompanies most of steps of the design and offers a more realistic interpretation of results. The mentioned algorithmic and software **tools and expertise** are available in the Department, but their usage is restricted mainly to the Department. There has been a **goal** to spread tools' usage outside, for a long time, see Kárný and Halousková [4, 5]. A significant work has been done in this direction. A complete design cycle of adaptive controller has been covered. There have been prepared the **previous** versions of **Designer,** providing corresponding software tools. The last one is available as ABET (Adaptive Bayesian Estimation Toolbox), see Nedoma et al [8, 9], Nedoma [10]. The last versions were written in MATLAB [7]. It provides an experimental environment, but from the point of view of the **ordinary user**, i.e. ordinary control engineer, it is still very hard to start with it. The user can hardly overcome this barrier. To **solve this problem** we intend to use "all available means to succeed".

This **paper** is divided into several **sections**. In the project we use object-oriented (OO) approach to the analysis and design (A&D) of the Designer. The same is valid for the illustrations, where we use the corresponding graphical language, UML (Universal Modeling Language), see Fowler [3]. Analysis, UML and analysis results are shortly described in the Analysis section. The next section, Architecture, describes the main architectural features of the Designer, with special attention paid to object-oriented CASE (Computer Aided Software Engineering) and knowledge base (KB). The next section, Communication, describes the main features of communication between the user and Designer. The next section, Processing, describes the main features of Designer processing. The last section is the Conclusion. Some project details are briefly illustrated in Figures. A more detailed

[1] As "project" and "design" are synonyms, our goal is "design of design". To distinguish these designs we use "design" for the design of adaptive control, we use "project" for the overall analysis and design of design of adaptive control and we use "proposal" for the design (without analysis) of design of adaptive control, mainly.

project description can be found in Bůcha et al [1, 2].

2. ANALYSIS

The analysis is mainly oriented both to identification of users' requirements and to the previous version of the Designer. We recognize four types of **users**, i.e. ordinary user, expert user, administrator and developer. *In cooperation with potential users*, we created a list of main **requirements** on the Designer [1]. The aim of the third part is to understand why the mentioned goal (to spread usage) has not been reached. We begin with a context.

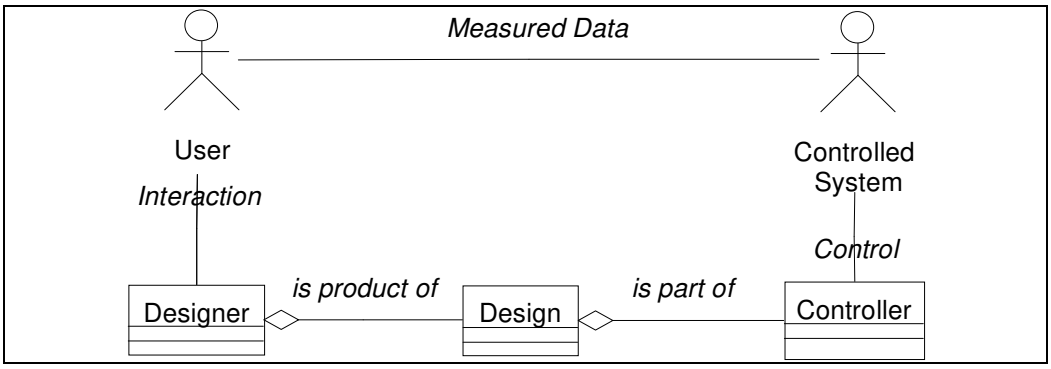

Figure 1: Main Class Diagram

The **overall context** of the previous Designer is shown on the Figure 1. We use UML graphical language in following Figures. The short explanation of used **language symbols** follows: Named *rectangle*, with two additional lines inside, means a class, e.g. Designer. The class describes its objects. Named *man icon* means an active (external) class, e.g. User. Controlled System is considered active as the man icon indicates. (Named) *line* connecting two classes means an association relation, e.g. Interaction. If a line is oriented with arrow then it means client – server relation. It is used for a data flow, event and other relations. (Named) *arrow with a diamond* head means that a class is a part of another class, e.g. Design is part of Designer.

The main classes in this context are Designer, User and Controlled System. There are also Design and Controller classes that are parts of Designer and Design class. They are considered in Figure 1 because they have a special relation to the context. The named lines indicate relations between classes. The meanings of classes and relations are self-evident. Controlled System can be both real and simulated. There are also classes corresponding to association relations. An example is Interaction. For simplicity, they are shown neither in Figure 1, they are shown in Figure 2.

The context of previous Designer is practically the same as the context of the (new) Designer. The differences are on lower architectural levels, i.e. in the *contents, exact meanings and implementations* of classes and their relations. The ABET contains about 150 m-functions, i.e. 170 KB of MATLAB source code, other C++ code, other code in demonstration programs etc. It is a set of programming tools, i.e. subroutines, classes, objects, helps, case examples and documentation that allow to an expert user to produce design, i.e., to *create and run a program* that can produce control action(s).

We can say that the previous approach has its pros and cons. **Pros** are that in this bottom-up way the necessary foundations for the Designer have been built. **Cons** are that to use the previous Designer means to program in its language and system. It is not easy as the problems are not simple, system and language is neither *stabilized*, nor well documented, nor very *understandable*, nor sufficiently *reliable*, nor fully *implemented*. The **requirements** resulting from the analysis are clear: *Make it easy, stabilized, …*

3. ARCHITECTURE

The Designer architecture has the following **main features**: 1) OO CASE as a nucleus of the project, 2) knowledge base as a nucleus of the Designer control, 3) efficient users' communication, 4)

use of prior knowledge, 5) extension of the existing algorithmic and software tools if it is necessary to ensure completeness of the system.

1) The **nucleus** is needed to stabilize and standardize somewhere the project terminology, concepts and approaches. The **CASE** is needed to provide for project methods, documentation, dynamics, reliability, implementation support and user involvement. The CASE changes the character of the project from the currently prevailing bottom-up to top-down, too. The **OO CASE** is needed to provide for the solution of project complexity and for developers cooperation via UML. All main features are interdependent. The object-orientation and CASE-orientation influences importantly all the other architectural features. This is why we put this feature on the top of main feature list.

The orientation on OO CASE is extended into a more general use of **CAD** tools to support the project, i.e. use e.g. GUI (Graphical User Interface), MATLAB, expert system shell, testing tools, to further support high quality, correctness, requirements attainment, dynamics of development, easy future development and administration of the project and product.

2) **KB** is needed to allow using *knowledge about the existing tools and about design process* to integrate these tools into one comprehensible and reliable system. Knowledge is spread in the documentation, examples and helps and among expert designers in the Department. There is general and specific **knowledge** used by the Designer. This more *general* and/or more *stable* one is held in KB, that more *specific* and/or more *variable* one is held in database (DB), as DB can be more easily actualized. There is technical and problem-oriented knowledge. The **technical** knowledge specifies how to use ABET functions and objects, what are relations among them, etc. It is hidden to the ordinary user and it is implemented by standard programming tools. However, this knowledge is accessible to the expert user and in this case, it is implemented in KB. The second part of knowledge, the **problem-oriented** knowledge specifies how to manage the whole design in order to reach a successful solution. It has mainly a heuristic character. The part of this knowledge is already identified. The extent of the rest of this knowledge is dependent on experiments that will be done with the early versions of the system. This processing knowledge is accessible to the user. In this part of KB, there can be "contradictory" knowledge, dependent on expert who provided it.

For example, specific characteristics of m-functions like their input and output parameter names, their usage of and utilization in other functions, their hierarchical relations, their characteristics of computational complexity, characteristics of information processed by these functions like their location, extent and form, their hierarchical relations are held in DB. The general characteristics like specification of cooperation of these functions, knowledge, how to derive the overall computational complexity, are held in KB.

KB extent is not large, at present. It has about 20 rules. This state is also caused by our concentration on the creation of the first prototype, where only a restricted set of rules is considered. The rules are fairly complex. Several rules are illustrated in the next sections of the paper.

KB is **object-oriented**. The justifications are: a) There is a fairly complex hierarchy of objects in the design, with composition and inheritance relations, with needs for encapsulation. For example, our steps and/or covering functions (see section Processing) are self-contained and the concept of hiding unnecessary information (see section Communication) aims at encapsulation and polymorphism, too. There are send, receive, plot operations of polymorph type. b) The previous approach had some object-oriented features. There are, e.g., the following classes of objects: Project, Step, M-function, Problem, data flows, e.g. Design Specification, Control Action, User and Expert User.

4. COMMUNICATION

Efficient users' communication is needed to decrease the complexity of usage and to provide a better users' understanding of the system. It is solved by proposing a comprehensible *communication* language, by *providing* the user with "*exactly*" as much information as he needs. This means 1) hiding as much information as possible from him, 2) providing as much information as possible, if he needs it, 3) providing exact and consistent information and 4) knowing "exactly" what are his needs in various situations.

To implement these concepts, the interaction with the user, GUI, is a) in a *top-down* manner, b) with a *hypertext-oriented help*, produced directly from the project CASE depository, c) with a *good*

documentation, also produced directly from the project depository, d) with access to the system via *Internet*.

For the check of applicability, we specified precisely the **range of control problem** that can be solved (and must be checked with the user): There are the following **rules**:

1) Designer is *to be used mainly* when the classical *solution has reached its limits*. It means, use Designer when the control problem is currently being solved by set of mutually independent control loops (or when it can be solved in such a way), e.g. there is a similar system or there is a corresponding project, but this solution has reached its limits because it is a too complex problem

2) Designer is *to be used also* when some loops are more complex, e.g. the solved *control problem has long delays or variable delays, there is feedforward, or need for multivariate solution*

3) Designer is *to be used only* when the solved *control problem can be linearized*, e.g. the user knows that it is not necessary to add nonlinear elements to solve the control problem, and when the (economical) importance of the problem is not high, e.g. it means that it is not worth of hiring a Ph.D. level professional to build and/or operate a specific nonlinear model

4) Designer is *not to be used*, when the solved *control problem is a simple one*, e.g. when PID control is sufficient.

There are further preconditions of the use, but they are on a different level, they cannot be checked in advance:

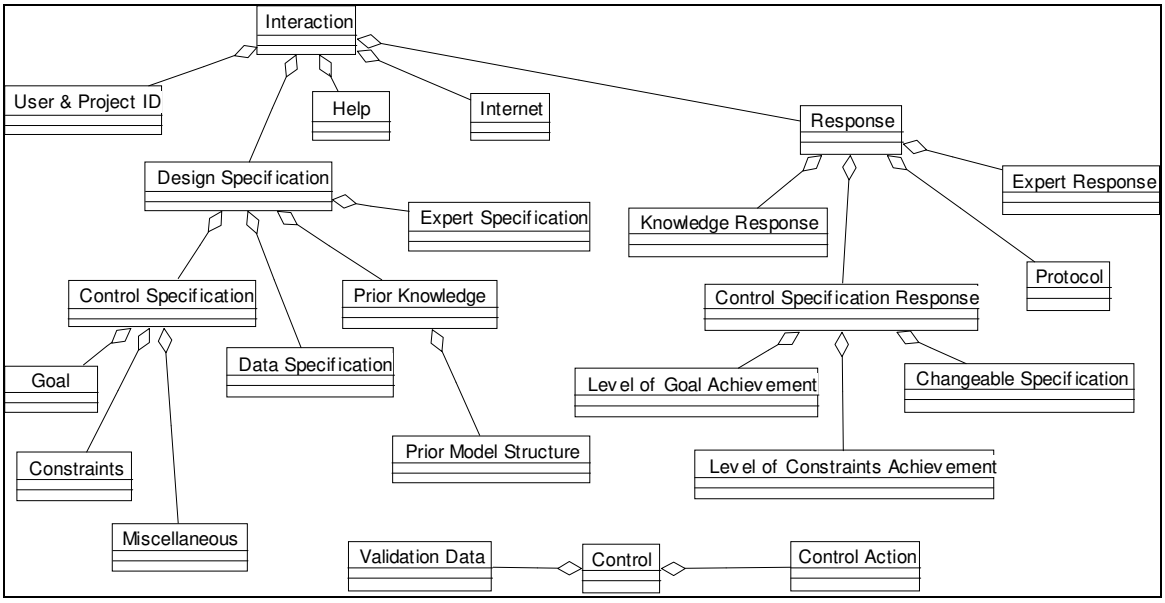

Figure 2: Interface Structure

1) Good result could be obtained if the data are measured on a *sufficiently excited system* and/or a *sufficient prior knowledge* exists. However, this can mostly be recognized only after the majority of the design is performed.

2) The design *goal* should be at least approximately describable by a *quadratic loss*.

In the range of the concept of hiding information, the user *learns as little as possible*. GUI is made *as familiar to him as possible*. For example the whole GUI is in the style of MS Word (Office) and/or MATLAB that are typical environments for the user, i.e. File, Edit, View, Options, Window, Help in the Main menu, their pull-down menus, tool-bars etc. Within this, the concept of providing *as much as necessary information* is implemented. The user can obtain all definitions of used concepts in a hypertext form, including all usual *synonyms*, *explanations* and *justifications* of the chosen solution.

To illustrate these generally specified directions we list some of the user **inputs and outputs**. At the most abstract level, the user can specify structures (classes) shown in Figure 2. In the Prior Knowledge part he can specify e.g. Steady State Signals, System Static Gain and Cut-off-Frequencies. The user can obtain various results as a Response e.g. Level of Goal Achievement, Level of

Constraints Achievement. All “mathematical” and/or “formal” information, like model structures and theirs parameters, penalization matrices, theirs probability distributions, covariance matrices and guesses, internal ABET objects and theirs pointers are hidden to the ordinary user. They are available to the expert user, only.

5. PROCESSING

In Figure 3 there are shown the main steps of the Designer processing with data flows between these steps and/or User and Controlled System. Some of these data flows are described in the Communication section.

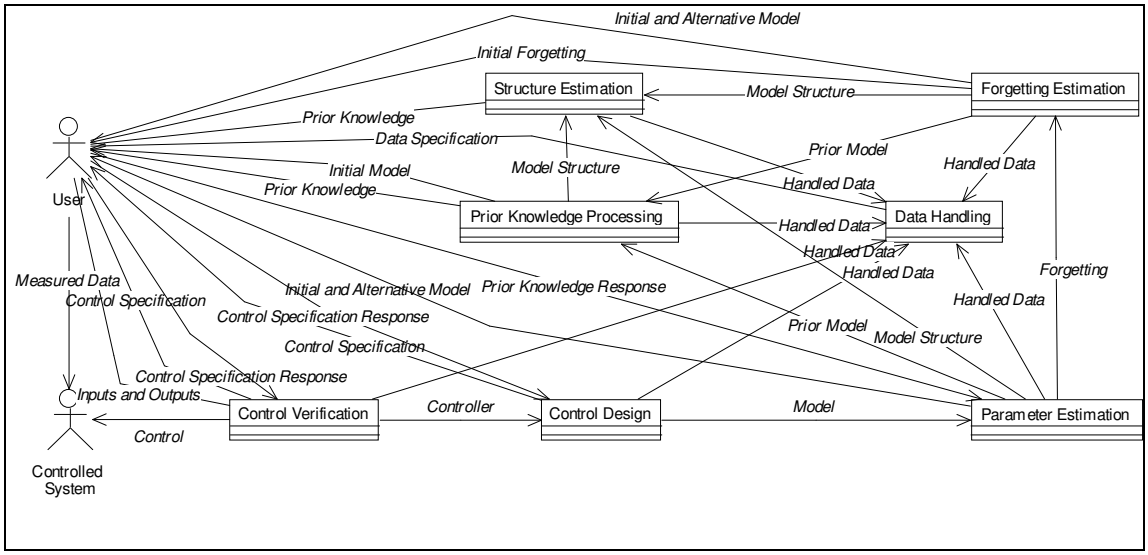

Figure 3: Class Diagram of Steps

The design has seven main **steps**: Data Handling, Prior Knowledge Processing, Structure Estimation, Forgetting Estimation, Parameter Estimation, Control Design and Control Verification. These more or less traditional steps have special features regarding so the possibility to include user’s knowledge. The step of Prior Knowledge Processing computes Initial Model (Parameter Distribution) from Prior Knowledge. The step of Data Handling provides the information obtained by handling (measured) data, Handled Data. Currently, two functions are supported in this step, the elimination of outlying measurements and the normalization of measurements. The step of Structure Estimation searches for the best Model Structure of the Controlled System, i.e. that qualitative dependency of outputs on delayed inputs and outputs that maxized posterior likelihood. The step of Forgetting Estimation selects the optimum forgetting technique, Forgetting. It allows Parameter Estimation step to compute Model from Initial Model and Handled Data. Model including Model Structure is the description of Controlled System model. The step of Control Design builds and sets up the Controller that tries to reach the desired closed loop behavior. Step Control Verification checks the set up of the Controller either on a real system or on some user-provided model. The steps are run in a sequential order, mainly. The successful conclusion of one step is a prerequisite for the following step. For the batch version of the system, so called **covering functions** are provided that correspond to each step. To illustrate the use of knowledge we present two simplified rules that guide the overall operation and interaction with the user.

Main Loop Rule: If Expectation of Input Range is greater then Desired Expectation of Input Range, then this “space” can be “traded” for Expectation of Output Range by increasing Desired Expectation of Input Range, and vice-versa. If there is no space either in Expectation of Input Range or in Expectation of Output Range, then try to gather more data in order to decrease the reserve needed for covering uncertainities.

Rule Applied after Step Structure Estimation: If some output is independent of inputs then Data or Prior Knowledge is insufficient. In many cases, lack of dependence of output on inputs is

caused by lack of Prior Knowledge.

The system allows and provokes the user to use his *knowledge of controlled system* to support successful design. The concept of using this knowledge is fundamental for the Designer.

6. CONCLUSION

The discussions with potential users assured that the system like the projected one is very much needed. Our work confirmed the project's assumption that there are tools that allow building the Designer. One of the important project's assumptions was the existence of experience and knowledge about the utilization of Bayesian methods in analysis and design of controllers. This assumption was confirmed only partially. There is same knowledge that can be used, but it is not sure if any significant knowledge can be further gathered. More research is needed in this direction.

In the paper, we are using many, roughly 50, not precisely specified terms. They are used to illustrate the project on some more detailed level. We have identified about 310 such terms in the project. All need to be more analized with OO CASE. The details can be found in [1]. But, OO CASE is not only A&D tool for current project. In its UML representation it is an important tool for a future cooperation on the further development.

An important effort has been done to create the foundations for the project. We estimate it at a level of hundred(s?) research-years. The rough estimate of overall necessary effort for this project is on level of 4 man-years. We are now (at the time of writing this paper) at the beginning of detailed Analysis and Design stage. The project roughly follows a classical division into project stages, i.e. Feasibility study, Analysis and Design, Implementation and Operation stages. From this point of view, the future is predictable.

7. REFERENCES

1. BŮCHA J., KÁRNÝ M., NEDOMA P., BOHM J. and ROJÍČEK J.: Designer 2000, Feasibility Study, Technical Report, UTIA, AVCR, Prague, Czech Republic, 1998.
2. BŮCHA J., KÁRNÝ M., NEDOMA P., BOHM J. and ROJÍČEK J.: Designer 2000, In Preprints of Control 98 (accepted), 1998.
3. FOWLER M. and SCOTT K.: UML Distilled, Addison-Wesley, N.Y., USA., 1997.
4. KÁRNÝ M. and HALOUSKOVÁ A.: Pre-tuning of self-tuners, In: Advances in Model-Based Predictive Control, D. W. CLARKE (Ed.), Oxford University Press, Oxford, UK, 1994, pp. 333-343.
5. KÁRNÝ M. and HALOUSKOVÁ A.: Preliminary tuning of self-tuners, In: Advanced Methods in Adaptive Control for Industrial Application, K. WARWICK, KÁRNÝ M. and HALOUSKOVÁ A. (Eds.), 158, Springer Verlag, Germany, 1991.
6. KÁRNÝ M., HALOUSKOVÁ A., BOHM J., KULHAVÝ R. and NEDOMA P.: Design of linear quadratic adaptive control: Theory and algorithms for practice, Kybernetika, 21, Supplement to No. 3, 4, 5, 6, 1985.
7. MATLAB: Using MATLAB, Version 5, The Math Works, Natick, Mass, 1996.
8. NEDOMA P., KÁRNÝ M. and BOHM J.: DESIGNER: Preliminary Tuning Adaptive Controllers, Proc. of the 2nd Process Control, ŘÍP 1996, University of Pardubice, Horní Bečva, Czech Republic, 1996.
9. NEDOMA P., KÁRNÝ M. and BOHM J.: Project DESIGNER, Proc. of the spring school Adaptive and Predictive Control, Oxford University, Oxford, UK, 1996.
10. NEDOMA P.: ABET97, Adaptive Bayesian Estimation Toolbox, A Tutorial, Technical Report, UTIA, AVCR, Prague, Czech Republic, 1998.
11. PETERKA V.: Bayesian system identification, In: Trends and Progress in System Identification, P. EYKHOFF (Ed.). Pergamon Press, Oxford, UK, 1981, pp. 239-304.

Acknowledgement: This project is partially supported by the grant "Adaptive Systems: Theory, algorithms and software practice" GA AV ČR No. 102/97/0118.