0

Engineering of Systems for Application of Scientific Computing in Industry

by
W. Loeve

National Aerospace Laboratory (NLR),
P.O. Box 90502, 1006 BM Amsterdam, The Netherlands

Abstract

Mathematics software is of growing importance for computer simulation in industrial computer aided engineering. To be applicable in industry the mathematics software and supporting software must be structured in such a way that functions and performance can be maintained easily. In the present paper a method is described for development of mathematics software in such a way that this requirement can be met.

1. INTRODUCTION

The methods and the software for computer simulation in many cases are produced in a research environment by industry engineers, by specialists of the type that in the past was involved in physical experiments, by theoretical physicists or by mathematicians. This is why computer simulation often is called scientific computing. The concerning so-called mathematics software or scientific software in general suffers from a number of shortcomings.

The shortcomings of mathematics software generated in a research environment concern:

- limited accessibility as a result of the implicit assumption that users are skilled in numerous areas such as mathematics, numerical algorithms, computer programming and computer operating systems in addition to their own application domain.
- limited availability and integrity as a result of difficulties for the method developers to organize, re-use and integrate a continuing growing amount of mathematics software and supporting utility programs.
- limited continuity as a result of the fact that industry engineers and the scientists mentioned above are not experienced in software engineering and as a consequence the software structure and the description of it are not suitable for cost effective maintenance.
- insufficient possibilities to ensure that software becomes available timely.

The existing situation leads to a waste of time for scientists and industry engineers and to underexploitation and misuse of existing software and data generated with the help of this software. The situation only can be improved if it is realized that engineering of software for application in industry requires a different mentality than research. For engineering of this software use has to be made of proper methods, techniques and tools to help to ensure the required performance of the mathematics software. This requires a contribution of computer scientists in addition to the contribution of the type of specialists mentioned above. In the present paper a method is described for solving technical and organizational problems that arise in cooperative development, design, production and validation of mathematics software.

2. THE DECOMPOSITION OF THE DEVELOPMENT AND PRODUCTION PROCESS OF MATHEMATICS SOFTWARE AND SPECIALISMS INVOLVED

The production process of new mathematics software starts after preliminary investigations and feasibility studies have indicated that it is desirable and feasible to produce the software. The first four phases of the production process concern modelling and mathematical aspects. The last six phases concern software production. In the present paper a rather complete set of phases is given as a basis of the software development method. This method is under development at NLR since 1979 [1]. The following phases can be distinguished in realization of mathematics software for simulation purposes:

1. Definition of the mental model.
   In this phase decisions are made concerning technical or physical aspects that will be simulated. The decisions cause modelling errors. In this phase contributions are required from eventual users who have to apply the resulting mathematics software and from specialists that are familiar with the aspects to be simulated.

2. Definition of the mathematical model.
   In this phase the mental model is described in mathematical terms. In this phase also assumptions are made with respect to the various aspects of the mental model. This causes representation errors. In this phase in addition to the specialists that contribute to phase 1, mathematicians have to contribute.

3. Definition of the discretized mathematical model.
   This phase is required to enable solution of continuous mathematical models in all cases for which these cannot be solved analytically. In simulation of flows the model is a finite set of often non-linear algebraic equations. In this phase discretization errors are introduced. This phase requires knowledge in the area of numerical mathematics to generate a proper set of equations.

4. Solution of the finite set of equations.
   In this phase the method to solve the set of equations is defined. In this phase solution errors can be introduced. In practice application of the solution method will require a computer. As a consequence the result of this phase can be considered as the software requirements which form the basis of the software production process. Knowledge about numerical mathematics and knowledge about computer science is essential.

5. Functional design.
   In this phase a structured description is produced of the coherent set of functions that will be performed by the software. Information sets and aspects of the interaction between the user and the combination of software and computer equipment are described. This phase results in a functional design. It is restricted to "what" the software will do. Again the cooperation is required of all types of specialists already mentioned.

6. Technical design.
   In this phase the top level description is given of the technical design of the software that concerns "how" the software will be structured. An important task is software engineering. Eventual users have to contribute to the testplan, the user manual and to the definition of the user interaction.

7. Detailed design.
   In this phase a detailed structured description is given of functions to be performed and of corresponding technical solutions. This phase results in the detailed design for which mainly software engineering expertise is required.

8. Implementation.
   In this phase software source code is generated and implemented on the available computers. Tests are executed to verify that the software entities are implemented correctly. This phase mainly requires software engineering capabilities.

9. Integration and validation.
   In this phase finally the completed software is integrated on the computer. Again tests are executed to verify that the software is a correct representation of the definition in step 4. In the validation process in this phase the relation between results from simulation and reality has to be indicated. Reference information for this comes from other simulation methods or physical experiments that both contain errors. All specialisms mentioned in the preceding phases have to contribute to the integration and validation phase.

10. Maintenance.
    This phase is meant to ensure continuity of the software. It concerns the system life cycle after production has been completed. This phase requires high quality contributions, mainly from users and software engineers.

The phased approach described above is characterized by strong interactions between the various phases. In principle these are caused by the alternate application of the described top down strategy and of a bottom up approach that is required for many aspects. The first three modelling steps and step 4 in which the finite set of equations is solved are mutually dependant. It requires sufficient expertise of each specialist involved to ensure rapid convergence of the modelling effort. In practice for new technical critical aspects mostly a feasibility study is required in advance to be executed in a research environment. Management has to make sure that efforts in the feasibility study are limited to technical critical elements. The cooperation of all specialists involved under the circumstances mentioned above requires organizational measures.

3. ORGANIZATIONAL ASPECTS OF THE METHOD FOR COOPERATIVE DEVELOPMENT AND PRODUCTION OF MATHEMATICS SOFTWARE

The quality of the development, of the production process and of the product depends strongly on the skills of the specialists involved. These skills have not yet been standardized sufficiently. This means that many ad hoc technical and organizational measures have to be taken to realise cost effective cooperative development and production of mathematics software without undesirable shortcomings. However, a number of general measures can be described.

Cooperative development and production tasks are usually assigned to ad hoc project groups. These groups contain specialists that have between them the skills that are relevant to the task. The project groups often have part time members. These members contribute to tasks of several project groups simultaneously. The resulting organizational form resembles the Adhocracy described by Mintzberg [2]. As has been described already by Mintzberg [2] the success of cooperative execution of complex and untried tasks strongly depends on the motivation, good will and mutual respect among the members of the project group.

Three types of cooperative work can be distinguished [3]:

- Coordination. This is a cooperative process where individuals coordinate their actions with those of others. Each action needs results from other actions and/or creates results that are input for other actions. The success of coordination depends on appropriate decomposition of the total task in sub-tasks.
- Collaboration. This is a form of cooperation in which individuals work together in order to achieve a single goal. The prime requirement to make collaboration possible is sharing of information. The second requirement is that a common understanding exists of the goal and the process for achieving it. The third requirement is that each individual knows what he is responsible for.
- Codecision. This occurs in cases that a group of individuals must reach a joint decision. Again a requirement is a common understanding of the goal and the process of achieving it.

The circumstances that are required for successful cooperative engineering of mathematics software can be derived from the requirements that are defined for the three types of cooperation. These are:

1. Common understanding of the goal and the process of achieving it.
2. Appropriate decomposition of the total task.
3. Synchronization of actions.
4. Assignment of responsibilities to each contributing individual.
5. Sharing information.

The first requirement means that the goal of the cooperative actions has to be defined. For this use is made of a Predesign in which the toplevel architecture is presented and justified.

For the Predesign document use is made of results from feasibility studies. The first requirement also means that the process of achieving the goal has to be described. This is described in the so called Project plan. It contains a description of the phases that are distinguished as described in section 2. It also contains a description of the result of each phase.

To fulfil the second requirement mentioned above, in the project plan a specific work breakdown is based on the information presented in the predesign document.

To fulfil the third requirement mentioned above a time schedule is presented in the project plan for the actions that are related to each task.

To fulfil the fourth requirement, in the project plan all key persons are mentioned and responsibilities are assigned to them explicitly in the project plan.

The fifth requirement means that relevant information has to be accessible for any member of the project group on a need to know basis. Information becomes available in three basic ways:

- Documents with written text and diagrams such as generated in modelling and design phases of the software development. This form is also applied for user manuals.
- Software as a result of implementation.
- Data such as generated by the software or from external sources.

Management must be able to be sure that the required quality of the information is realized. The quality related aspects are described in the quality assurance plan (q/a) in which a description is given of the methods, techniques and tools that will be applied for execution of the tasks described in the project plan. As mentioned at the beginning of this section for mathematics software the quality depends largely on the skills of the people who perform the tasks. In relation to this in the quality assurance plan a description is given of the education and the experience of the people who will contribute to the cooperation. In the q/a plan it also is described how execution of the distinguished subtasks is controlled to ensure proper results.

Creation of all the information mentioned requires an adequate attitude of specialists involved. Making sure that software can be regarded as information that can be shared requires technical measures.

4. TECHNICAL ASPECTS OF THE METHOD FOR COOPERATIVE DEVELOPMENT AND PRODUCTION OF MATHEMATICS SOFTWARE

The organizational approach described in the preceding section has to be supplemented by a well defined technical approach in the method for cooperative development and production of mathematics software. Both approaches have to be in harmony with each other. In practice it appears that the most important requirement for the realization of cooperation is sharing of information in a ready-for-use form. This means that a method for cooperative engineering of mathematics software shall technically enforce a standard way of structuring, storing and retrieving of information.
This concerns documents, data and software.

Already in the eighties in the Netherlands attention was paid to development of a method for engineering and storing of software based on a decomposition suitable for information sharing.

The software engineering approach that is applied now is based on contemporary techniques. In these two types of elements are distinguished, modules and objects:

- a module is a software component that performs an activity.
- an object is information, and modules encapsulating that information.

Modules are defined by: module name, description of what information is transformed by the module into what other information (it's interface) and what the transformation consist of (its function).
Objects are defined by: object name, description of what information is present in the object, and the definition of all modules encapsulating that information.

An information system suitable for information sharing is created for computational fluid dynamics. A scheme of this is presented in figure 1. It is the result of a cooperation of the National Aerospace Laboratory (NLR), Delft Hydraulics, the University of Twente and Delft University of Technology. As such it forms the technical bases of the Expertise Centre for Computational Fluid Mechanics in the Netherlands [4]. This forms part of the national infrastructure of services and facilities in the area of information technology.

In the system presented in figure 1 an executive function is indicated in addition to the functions mentioned so far. The executive supports execution of computer programs on a network of different computers that are required for specific types of processing [4]. The executive supports file handling in the network and enables the user to control processes in the network by means of assigning windows on the screen to specific processes or to specific computers in the network. In figure 2 the present computer network of NLR is given. The supercomputer is applied for computation processes that can be vectorized and that need the maximum processing power that can be made available to day via general purpose computers. The mainframe is used for the module management and the data management systems.
Data management is applied for exchange of information between installed software programs to enable easy transfer of programs to other environments (portability).

Finally workstations are used for specific commercially available software such as CAD/CAM software for geometry handling. The workstations also are applied for graphic aspects of pre-processing and post-processing and of graphic control of processing based on mathematics such as grid generation and solving a set of equations.

The system given in figure 1 is called ISNaS (Information System for flow simulation based on the Navier Stokes equations).

Reconsidering the requirements that originate from efficient cooperative development and production of mathematics software the following can be observed.

- Re-usability of software is improved by introduction of modules and objects.
- A transparent relation between functions in the Functional Design and modules in the Technical Design has a positive effect on the maintainability of the system.

Application of decomposition in modules and objects to development of mathematics software for computational fluid mechanics is described in [5].

5. CONCLUDING REMARKS

- A method is presented for development and production of mathematics software. The aim of the method is to produce software that can be applied for simulation purposes in industry.
- Mathematics software for applications in industry can be realized only if various specialists contribute to the development and production process. For this reason the method is designed in such a way that it supports cooperative engineering processes.
- The method has organizational and technical aspects. Both aspects need equal attention for the method to be effective.
- The quality of the result of application of the method depends on the skills and the attitude of the specialists that are involved in the development and production process.

6. REFERENCES

1. W. Loeve, Life-cycle-oriented methods for development and production of large-scale industrial mathematics software, Computers in Industry 18,Elsevier Science Publishers B.V.,(1992).
2. Henry Mintzberg, Structure in fives, Prentice Hall International Editors, (1983).
3. G. DeMichelis, Computer support for cooperative work, Butler Cox Foundation, (1990).
4. M.E.S. Vogels and W. Loeve, Development of ISNaS: An information system for flow simulation in design, Proceedings of CAPE89 Tokyo, North Holland, Amsterdam, (1989).
5. M.E.S. Vogels, Top down engineering of a flow solver, Design Documents and software, Internal Reports, National Aerospace Laboratory NLR, NL, (1990-1992).

7. FIGURES

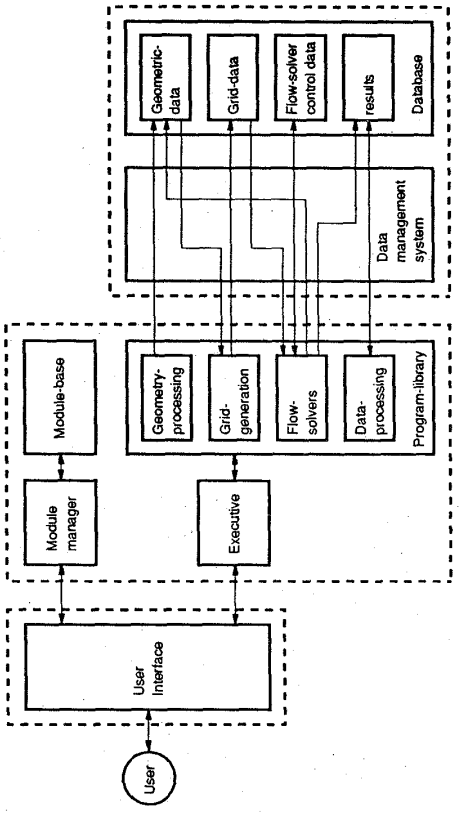

Fig. 1 Toplevel description of an open system for development, production and application of flow simulation software (ISNaS).

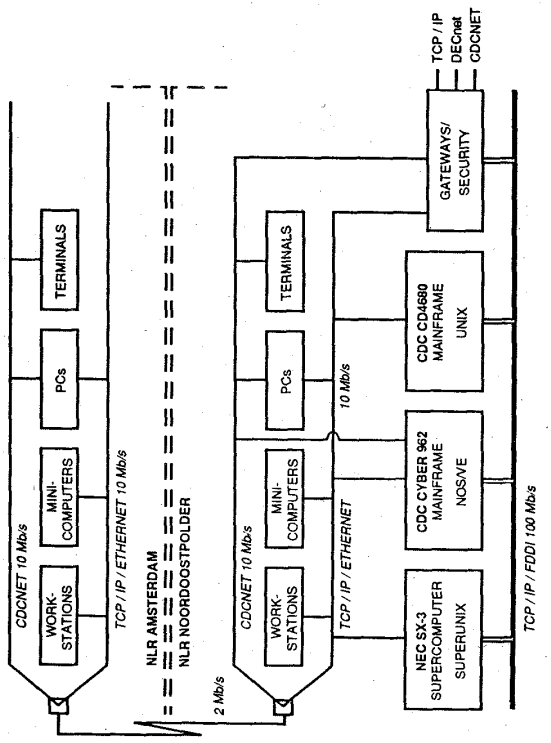

Fig. 2 The general purpose computernetwork for development of mathematics software at NLR