Quaestiones Informaticae

An official publication of the Computer Society of South Africa
'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika

Editor: Prof G. Wiechers
Department of Computer Science and Information Systems
University of South Africa
P.O. Box 392, Pretoria 0001

Editorial Advisory Board

PROFESSOR D. W. BARRON
Department of Mathematics
The University
Southampton S09 5NH
England

PROFESSOR K. GREGGOR
Computer Centre
University of Port Elizabeth
Port Elizabeth 6001
South Africa

PROFESSOR K. MACGREGOR
Department of Computer Science
University of Cape Town
Private Bag
Rondebosch 7700
South Africa

PROFESSOR M. H. WILLIAMS
Department of Computer Science
Herriot-Watt University
Edinburgh
Scotland

MR P. P. ROETS
NRIMS
CSIR
P.O. Box 395
Pretoria 0001
South Africa

PROFESSOR S. H. VON SOLMS
Department of Computer Science
Rand Afrikaans University
Auckland Park
Johannesburg 2001
South Africa

DR H. MESSERSCHMIDT
IBM South Africa
P.O. Box 1419
Johannesburg 2000

MR P. C. PIROW
Graduate School of Business Administration
University of the Witwatersrand
P.O. Box 31170
Braamfontein 2017
South Africa

Subscriptions

Annual subscriptions are as follows:

<table>
<thead>
<tr>
<th></th>
<th>SA</th>
<th>US</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individuals</td>
<td>R6</td>
<td>$7</td>
<td>£3.0</td>
</tr>
<tr>
<td>Institutions</td>
<td>R12</td>
<td>$14</td>
<td>£6.0</td>
</tr>
</tbody>
</table>

Quaestiones Informaticae is prepared for publication by Thomson Publications South Africa (Pty) Ltd for the Computer Society of South Africa.
NOTE FROM THE EDITOR

Three points must be made by way of introduction to the second issue of Volume 2 of Quaestiones Informaticae.

Firstly, an apology is in order for the mistake in the date (November 1983 instead of 1982) at the foot of my note introducing the preceding issue. Lacking the services of a professional proof reader, printing errors are bound to show up from time to time, but it is hoped that their number will be kept to a minimum!

Secondly, it is a pleasure to announce that this journal will not only serve to publish papers of a scientific or technical nature on computing matters under the auspices of the Computer Society of South Africa. An agreement has been reached to share the facilities of Quaestiones Informaticae between the CSSA and SAICS, the South African Institute of Computer Scientists. Henceforth this journal will also be used to publish the Transactions of this Institute. This implies certain changes to the cover pages which will be implemented in future issues. I shall continue to serve as editor, but on behalf of SAICS Prof R. J. van den Heever will share some of my duties and act as co-editor.

Finally Mr Edwin Anderssen, of Rand Afrikaanse Universiteit, has agreed to serve as circulation manager for Quaestiones Informaticae. I am grateful indeed that he is willing to serve the journal in this capacity, and look forward to a long period of fruitful cooperation.

G WIECHERS

May, 1983
Managing and Documenting 10-20 Man Year Projects

P. Visser
Software Management Systems Pretoria

Abstract
This paper presents a summary of tutorial material on accepted management techniques and philosophies as applied to the development of large software systems, based upon the documentation for such systems.

INTRODUCTION
Techniques and facilities to increase production can only be effective if they form part of a total management effort. This applies especially in the development of large software systems. Management of software development is not different from normal management, yet it is known to be difficult. The reasons can be found in the characteristics of software development which resembles research work in most management aspects. The method of implementation of management techniques must therefore be adapted to these characteristics:
- Not visible in the normal sense.
- Production facilities not obvious.
- Type of personnel — highly skilled.
- "Peoples" systems.
- End product defined as part of production — direct control is not possible as for manufacturing.
- Measurement dependent on judgement.

PRODUCTION MANAGEMENT FOR SOFTWARE PROJECTS
The purpose of production management is to accomplish the efficient use of company resources which constitute mainly personnel in the case of software development. Planning and control of the following aspects are therefore essential.
- Personnel — appointments, development, training and scheduling.
- Realistic progress estimates.
- Quality of completed items.
- Running cost per project.

Planning should establish realistic objectives to be used as criteria in the control process. The methods for establishing these objectives must however be adapted to the characteristics of software development. The planning and control systems must be based on self-balancing principles. Systems based on direct control principles can only destroy self-motivation and thereby the basis for a democratic process which is essential in the development of systems by a team. (Democratic process is used here in the sense that sound creative ideas from any team member can be incorporated into the design in a controlled way).

MANAGEMENT INPUTS
Facilities
Word processing facilities are so essential to the design process as hardware, compilers and other utilities to the development of executable programs. Other essential facilities (services) are typing/editing and a program library.

Software Standards
Standards should prescribe work procedures suitable for the proper planning and control of progress, cost and quality. Standards should also set criteria for the evaluation of the quality of development work.

Development of Personnel
Personnel should be encouraged to use the various techniques and methods for software development skilfully. This can only be achieved through proper guidance in a competitive environment. Training in techniques and specific items (e.g. software packages) is also essential in the specialization of personnel in order to provide them with a proper background for executing their allocated tasks effectively.

Organization
The organization of the team is essential for the division of work. It must provide for the democratic process needed during design but without endangering the integrity of the design. An efficient project management system is also essential but must be an integral part of the organization and work procedures.

PLANNING AND CONTROL
Principles
Three activities must be pursued on an ongoing basis:
- Setting of criteria (planning)
- Measurement of performance
- Evaluation of performance and corrective actions

These activities must be directed towards:
- Quality of work
- Progress of work
- Cost of work

Criteria for evaluation (Objectives)
In order to use a criterion for objective evaluation it must be measurable and have short term attainability (2-4 weeks). To set criteria for progress therefore requires that a large project be broken into small, executable tasks, each with a defined physical output. Coding has a natural output in the form of program listings, but constitute the smallest portion of work. The output for all other phases should be prescribed in the form of documents to be generated. The completion of such a document then marks the completion of a task and should be achieved in a short period of time.

If the completion of each small task is measurable, the progress for the total project can be calculated. It is assumed in this type of calculation that all tasks needed for project completion are included. Software standards should therefore include a standard framework for the planning of a project. This framework must be consistent with the team's work procedures which must therefore also be prescribed. The planned activities must then be scheduled against the available personnel taking sequence, skills, deliveries and other factors into account. Using the relative weight of each task, an expected progress curve for the project can be established.

High quality of work can be attained only through the combined effect of the following:
— Setting of standards for design, coding, testing and documentation.
— Correct use, development and training of personnel.
— Prescription of work procedures which will result in work of high quality.

Setting cost criteria is merely a calculation process, using the progress criteria as a basis.

Performance measurement
Progress estimates for small tasks individually contribute very little to the total progress. It is however assumed that no rework will be necessary. Quality control on the product of each small task is therefore essential as well as the total coordination of tasks. Especially during the design phase, coordinated performance to attain design integrity is important, and methods to attain this should be build into the work procedures and standards for documentation. Cost measurement can again be calculated from reliable progress estimates.

Evaluation and corrective action
If techniques are established for setting criteria as well as performance measurement, evaluation can be done regularly and corrective actions can be taken timely. Measurement of the effect of delays is also possible both in terms of development time and development cost.

SOFTWARE STANDARDS

Implementation of Software Standards
Standards should be the documentation of sound practice developed from theory and positive experience. It should assist the organisation by providing a structure for the efficient operation of a department or team. Software standards should also describe a desired result rather than techniques open for misinterpretation. As shown in previous paragraphs the adoption of proper standards can create an environment for the development of software which is suitable for the execution of normal management practices. As such, software standards and their proper implementation are a prerequisite for planning and control of software projects. Software standards should be adhered to by competent software engineers, else it would only be a rulebook on the shelf of the manager.

Responsibilities and work procedures
A standard team structure is used in all organisations which should be adapted to the type of systems developed by the organization. Responsibilities and authority of the various positions should be set out clearly. Work procedures for the major activities of the team should be planned such that the desired end results will be attained. These activities include the different types of design, reviews and other.

Configuration Management and QA
Quality of design and coding can only be built into a system during development and must therefore be controlled within the project team. The only management input is through sound work procedures as discussed above and the training of competent personnel. Poorly designed systems normally result in delays caused by rework with an adverse effect on cost. The quality and training level of personnel must therefore always match the requirements of the system.

Product quality can be maintained by software quality assurance procedures. These procedures must be controlled from outside the development team. Configuration Management practices are geared towards safeguarding completed copies of software and documentation, and maintaining the integrity of the completed portion. The program library is the basic vehicle needed for this task. Configuration Management during development is just as essential as for a fully operational system. It must however be adapted for this work phase and its corresponding work procedures.

Configuration Management and Quality Assurance procedures must form an integrated system and must be adapted to the size of the organization and other characteristics of the system being developed. Change control procedures form a natural part of Configuration Control and should be incorporated.

Documentation Standards
Documents form the physical output of any type of design process. It is therefore necessary to define a set of documents suitable for the following processes and to cater for the specific characteristics of the type of systems developed by the organization.
— Customer and user information
— Further design and development processes
— Quality and progress control
— Coordination in the development process
— Maintenance

For each of these documents a standard layout and contents definition must be given. The documents should also be based on the same principles underlying the work procedures in order to be the natural outcome of such procedures.

Design and coding standards
As for documentation standards, design standards must also be correlated with the prescribed work procedures. These standards should be viewed as the main criteria for making design decisions. Standards should therefore be specified for each design level independently and should be applicable to methods, control structures and data structures.

Where a standard framework for program or system structures are used in an organization, these should also be included as part of these standards. Design standards should not be open for misinterpretation. Techniques should therefore not be prescribed as a standard but rather the desired characteristics of the end result.

UNDERLYING PRINCIPLES FOR THE DEVELOPMENT OF LARGE DYNAMIC SYSTEMS

Characteristics of development
In a dynamic computer system coordinated actions are needed. The interactions and interfaces between programs result in a complex structure which influences the correctness and efficiency of the system. Mastering this global complexity in a system is the key to successful systems.

A totally integrated approach to system design and development can lead to the mastering of complexity and prevent mistakes at a lower level rather than correcting them. The design process should therefore be an ordered sequence of problem definition and decision making. The criteria for choosing between alternatives depend on the design stage. Such an ordered approach with specified decision criteria can maintain the integrity of the design throughout the design and implementation stages. Program development has a physical output in the form of compiler listings and demonstrable software. Too much attention is therefore focussed on this stage and software personnel are motivated to ignore the preceding design decisions, thereby ignoring the global complexity of a system.

The only solution to this dilemma is to enforce prescribed documentation in such a way as to provide the design phases with physical output. This facilitates progress, cost and quality control of the design phase and therefore the application of normal management techniques. More important however, it also enforces the ordered sequence of decisions by the designers, which is an essential prerequisite for the development of efficient software systems.

Global approach to design
The techniques of top-down design, development and testing focus on the program as total entity. The same approach may however be used to create an ordered approach to the design of dynamic software systems. This global design approach consists of various design levels, each of which focus on a particular entity, viewed as a black box at that stage, and its interactions with other similar entities and its environment. These levels of design determine the decision making criteria at that level, and are discussed below. In the same way the techniques of top-
down development and testing may be implemented.

This approach then allows one to focus attention on the global structure of a dynamic system during design, development and testing. The biggest problem encountered with the global design approach is that a great deal of insight with regard to the later implementation of each concept is needed. The only way to overcome this problem is knowledge, experience and preliminary investigation of implementation techniques.

**External design**

All systems are built to satisfy a specific need. The first task is to formalize this need in order to define a goal for system design and development. With this in mind, and taking resources and circumstances into account, a formal set of objectives for the system can be defined. The environment includes operators, recipients of output and equipment.

To describe all actions, reactions and other functions and characteristics in a consistent way, a standardized approach must be used. External design has as output documents describing the objectives and functional specification of a system. The criteria to guide design decisions during this phase must be stated explicitly in both company standards and in the objectives defined for each system.

**System design**

The system design level focuses on the task as entity. Tasks must therefore be viewed as black boxes performing specified functions, interacting with one another and using specified interfaces to achieve this. The major activities during system design is therefore the following:

- Define independent subsystems i.e. subsystems which can perform their normal functions independently within their functional environment.
- Define the tasks in each subsystem.
- Define the interactions between tasks explicitly.
- Design interfaces between tasks as well as with the external environment.

The criteria for making design decisions during this stage, must be included in the company standards and must always be within the restrictions of the functional specification and system objectives.

A standardized form for documenting a system design must be included in the company standards. In these documents both the logic and the various data must be described. The logic of the system is described in terms of the functions and of the characteristics of tasks and their interactions. All types of data are described in the interface design. The various documents describing the system design therefore contain a complete specification for each task in the form of a description of all its functions, interactions and interfaces.

**Program design**

A task as defined during system design is a dynamic entity in the system, capable of communicating with other entities and reacting in a prescribed way on impulses depending on various operating conditions. Equivalent programs must therefore be designed to fulfill these specifications. The logic of each program and the corresponding datastructures are designed and documented during the program design phase.

This phase focuses on the module as entity. A program's logic are described through design methods and module definitions, as well as their dynamic and static structures. Data structures form the interfaces between modules and are as such also included in this design phase. A module is viewed as a separately compilable collection of statements and may contain several procedures. Various techniques for program design are documented in textbooks. The same applies for design criteria on which to base decisions during this phase. These criteria must however be incorporated into the company standards, as well as the approach to the documentation of this work phase.

**Module design**

Modules are the smallest units specified in design documentation. Module design forms part of program development. It is primarily concerned with the explicit planning of language code. Its outcome forms a natural part of module listings in the form of structured comments and the resultant code statements.

**TEAM OPERATION**

**Team organization**

Larger software projects can only be developed by teams with the accompanying problems of coordination between personnel and the maintenance of integrity of design. Specialization on the other hand can only be economical in large teams (more than 25) or in large software organizations. In the normal team for SA (5-20) a combination of ideas for team organization can be used.

The Project Manager is the coordinator with the customer and responsible for system analysis and specification. His counterpart is a senior designer responsible for the complete design and its implementation. Technically he is assisted by one or more designers and programmers.

**Design coordination**

The identification of subsystems and their interrelation is done by the senior designer following the system specification. During the design phase however each designer is allocated the sole responsibility for a subsystem with the most difficult or most important one to the senior designer. The design is then developed by the responsible person, assisted during meetings by all other designers. In this way other designers and especially the senior designer can present ideas, coordinate interfaces and define potential problem areas. The original ideas of the senior designer can in this way be maintained or replaced by better ideas without the danger of losing cohesiveness between parts of the system. The designer responsible for each subsystem should then be able to implement this design with the aid of one or more programmers which function in the same way as described above.

**Coordination of development and testing**

The workplan for development (coding and debugging) should follow the principles of top-down development which is a coordination tool in itself. On the other hand the standard for documents should be such that every aspect of interactions and interfaces is described fully.

Testing is validating that the development process generated the product expected by the user. Design documents should include a testplan and test procedures which describes the actions and reactions of the system. This will identify differences between user expectations and system specifications at a very early stage.

**Customer/user coordination**

Reviews during design and implementation of each sub-system are essential. After the design of each subsystem this design should be presented, highlighting the consequences of specifications and the restrictions of design ideas. Before acceptance testing a review should again be held to highlight implementation problems encountered and the solution implemented.
Notes for Contributors

The purpose of this Journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to: Prof. G. Wiechers at:

Department of Computer Science
University of South Africa
P.O. Box 392
Pretoria 0001
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely. The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by a summary of less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate keyword list and a list of relevant Computing Review categories.

Tables and figures

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled. Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings, etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged thus:


Proofs and reprints

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged against the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Fifty reprints of each article will be supplied free of charge. Additional copies may be purchased on a reprint order form which will accompany the proofs.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.
Quaestiones Informaticae

Contents/Inhoud

Die Operasionele Enkelbedienermodel* ..................................3
  J C van Niekerk

Detecting Errors in Computer Programs* ................................7
  Bill Hetzel, Peter Calingaert

Restructuring of the Conceptual Schema to produce DBMS
Schemata*.............................................................................11
  S Wulf

Managing and Documenting 10-20 Man Year Projects* ...........15
  P Visser

Data Structure Traces*...........................................................19
  S R Schach

Case-Grammar Representation of Programming Languages* ......25
  Judy Mallino Popelas, Peter Calingaert

Die Definisie en Implementasie van die taal Scrap*...............29
  Martha H van Rooyen

*Presented at the second South African Computer Symposium held on 28th and 29th October, 1981.