T. BARON
I. IVAN
Academy of Economic Studies

A. BALOG
I.C.I.

The Characteristics System of Software Products Quality

The paper presents modalities of identifying and classifying the characteristics and attributes for software products quality. Similarly, the main features are presented for each characteristic or attribute, as well as some considerations on measuring the levels and their interaction with the other characteristics or attributes. Aspects of the management of the software products quality are approached in conclusion.

1. INTRODUCTION

The passing to a new quality in the programming activity implies the use of advanced programming techniques, the carrying out of applications in solving the major problems of enterprises, the construction of high performance software products and the constant improvement of the way of using the facilities provided by the operating systems.

In this context, an extremely important part is incumbent on the analysis of the software products quality. It resides in outlining the characteristics of the software products, in establishing to what extent they meet with the requirements generated by the electronic computer-aided solving of complex problems, in establishing the grade of utility of the program systems and in determining the economic efficiency resulting from the use of software products in a considered time interval.

The high costs characteristic of the activity of constructing software products call for analysing the correlation between the performances of the software products and the effects of their current use in the economic units.

The performances are established by taking into consideration the time interval in which the final solution is obtained and the consumption of resources necessary for a complete solution of the concrete problems.

The performance level is influenced by the way in which the conception-design capacity, the execution capacity, the concordance capacity, the current use capacity, as well as the maintenance capacity have been ensured to the software product, at an adequate level.

The scheduling of the performance level is aimed to ensure the efficient use of each software product. The creation of characteristics leading to the obtaining of the planned performance level takes place in each stage of the construction of the software products determining the quality of products with the performer.

The outlining of the features along with the current use of the software products determines their quality with the user.

An important part in constructing the software products is incumbent on the management of quality which is leading to an optimal correlation between the standard of quality, the design/programming techniques used and the economic and financial effects implied by this level. Likewise, an equilibrium is obtained between the complex character of the processings performed by the programs and the consumptions of necessary resources.

2. IDENTIFYING AND CLASSIFYING THE QUALITY CHARACTERISTICS

Values for the quality characteristics can be assigned, to each software product. These values show high or, on the contrary, low levels for certain quality characteristics, according to the way of interpreting the results obtained by using the calculus formulas associated with the levels.

In designing the software products, stress can be laid, at a given moment, on the achievement of a quality characteristic at a superior level. Resulting for the other characteristics will be levels which are influenced by the priority characteristic but, which should be maintained within limits providing the operable character to the program in conditions of high economic efficiency. For some characteristics of the software products quality, exact measurements can be carried out, for others indices are constructed and values associated with them by convention are estimated. There are also characteristics for which no measurings can be made; they can be emphasized through the general behaviour of the software product from a utilization to another.

The programmes are implemented so as to meet with all the requirements put forward by the end-users. If, upon design, certain qualitative features have a priority character and others are secondary, with the passage to new variants or with the extended use by other beneficiaries, changes can be operated which make the other characteristics become priority ones. The dynamic character of the hierarchization of the components of the quality characteristics system ensures the maximum utility of the software product.

The characteristics and attributes of the software products quality form a complex dynamic system. The increase in the level of a characteristic may automatically lead both to the growth of the levels for some characteristics and to diminished levels for others. Programmers must know the way in which the quality characteristics and attributes interact in order

to ensure the obtaining of the balance necessary for the software products to fall within the admissible performance limits.

The characteristics of software products quality can be classified according to some criteria, such as for example:

— the person making evident the quality characteristic; there are characteristics pointed out by the software products' users as concrete problems are solved; there are quality characteristics made evident by the programmers intervening in various stages of the life cycle of the software product to make changes into the components of the product;

— the specificity of the software product makes one or several quality characteristics have priority, aiming mainly to attain the scheduled levels for them; the other characteristics are considered secondary and their levels will be included within the admissible limits;

— the moment when the quality characteristic is outlined; there are characteristics rendered evident in the stage of the initial (current) use of the software product; other characteristics are emphasized in the stages in which the software product is revised; there are characteristics pointed out in the transition stages of the software product.

The qualitative balance of the software product is established by the aim for which the product is constructed and by the forecast structure of its life cycle. A change in the structure of the life cycle or aim determines the achievement of another qualitative equilibrium for the respective software product.

The quality characteristics and attributes form a hierarchized system; at a higher level are the elements determining the quality (characteristics) and at the next level there is a set of attributes which can be combined in various ways influencing the share of the elements within the system.

3. QUALITY CHARACTERISTICS IN THE CURRENT USE OF SOFTWARE PRODUCTS

3.1. Reliability

A programme has the characteristic of reliability if it fulfils the processing functions required by the end-user, in a given time interval, without errors. The reliability of the software products is outlined by examining the tolerance to errors, a fact which lends robustness to the respective product or the intolerance to errors. In the first case, the aim of the programming activity is to achieve programme components which should further provide results even in case that some errors are signaled out. The techniques used are based on a growing degree of protection redundance in the programme both in the form of instruction sequences and within the structure of the initial data. In the second case, the designing of the programme components aims to discontinue the execution in case errors are found and their explicite signaling out. The techniques used in designing and constructing the programmes intolerant of errors aim to assure that the execution of the programmes satisfies the apriori reliability criteria. The achievement of the scheduled reliability level is an iterative process. The techniques leading to the carrying

out of reliable software products are the structured design and programming; the programmes testing and debugging will be carried out automatically.

The evaluation of the reliability of the software products is based on models taking into consideration their structure and behaviour. The forecast of the reliability of the software products is made by using the data gathered in the stage of their design. Data gathered in the stage of testing the programme are used in the calculus of experimental reliability. Operational reliability is calculated with the aid of data recorded as concrete problems are solved with the respective programme. The precision of the results implies the taking of longer time intervals into account.

The reliability level is influenced by the way in which the modularity, the structurability, the correctness and consistency of the software product have been ensured. Having a high level of reliability, the programme modules can be resued in solving other problems too, positively influencing the reliability level of the software products they are integrated into.

The obtaining of the scheduled reliability level depends on the intensity with which the designing and programming techniques are used, as well as on the experience gained by analysts and programmers in implementing complex software products.

3.2. Correctness

A software product is correct if the changes operated are leading to the obtaining of results which correspond, qualitatively and quantitatively, to the programming specifications. Correctness ensures the performance by the software product of the processing functions requested by the end-user. This characteristic can be rendered evident by successive tests and by a formal check up by using the source text of the software product.

In the case of testing, the conclusion can be drawn that for a set of tests, the programme is performing the processes required through the specifications. The correctness check up aims to prove, with the aid of a system of axioms, that starting from initial assertions, by operating some changes, final assertions can be reached; the programmes is performing the processes required through the specifications. The limits of the constraints within which the correctness is demonstrated establish whether the program is totally or partly correct or whether the programme is incorrect. In case the correctness is proved, the finishing of the programmes is taken into consideration, given by the finite number of steps covered by the operating programme; the final results are also dependent on the initial data of the problem which must be solved. To demonstrate the correctness, both the informal and formal methods are used. The informal methods have developed with the use of assertions and with the establishment of the ways of their generation to enable the automation of proving the correctness. The formal methods are based on the construction of well-defined formulas; the satisfaction or insatisfaction of the conditions for the programme to be correct is demonstrated with the aid of first order predicate calculus. All methods are accompanied by the example containing the ALGOL programme for calculating

the highest common factor of two numbers. Considering that the programmes currently elaborated in solving practical problems are worked out in the COBOL and PASCAL languages and are of big dimensions or are structured in modules, the methods for demonstrating the correctness will prove efficient only as far as programmes are constructed for ensuring the automation of the demonstration in conditions of high efficiency.

The correctness of the programmes is influenced by all the characteristics bringing about changes in the programmes structure.

3.3. Efficiency

A programme is efficacious when there is an optimal correlation between the consumptions of resources (execution time, basic internal memory, types and number of peripherals) and the complexity of the problem to be solved. The efficaciousness of the software product results from the analysis of the time during which the results are obtained with the desired precision, correlated with the basic internal memory, the volume of the intermediate results and the files used. This quality characteristic is inversely proportional to the effort required by the current use of the software product and for maintaining it into operation by updatings.

A programme is more efficacious than other programmes having the same complexity level if more reduced consumptions of resources are necessary for obtaining the final results. The comparability of the programmes is possible by using indices expressing the number of items processed in the time unit, the number of iterations performed in the time unit, etc.

The efficiency of a software product can be expressed by using also the notion of the programme output; the programme output is defined as a relationship between the complexity of the initial data and of the final results.

The efficaciousness of the software products is influenced by their complexity; the increase in the reliability and maintainability of the programmes is leading to the diminishing of the efficacy level of the programme. The measuring of efficacy is possible by recording the data defining the dimensions of the problems solved and the resources consumed for solving them. The efficiency index permits to estimate the necessary resources for solving the problems whose dimensions are known.

3.4. Safety in current use

Safety in current use establishes the extent to which a software product does not permit the operation of unauthorized or undesired changes into volumes of data, as well as the partial or total destruction of the volumes of data. The situations which may determine an accidental change in the volumes of data are generated by the quality of the input data and of the parameters activating various processing functions. These situations can be eliminated or reduced in frequency by using protection mechanisms, programme sequences containing complex tests checking the input data and

the quality of the parameters with which the processing functions are activated. Likewise, volumes of data can be copied and according to the users' decisions, the content of the copies of volumes is handled dynamically.

The evaluation of the safety degree in current use depends on the quality and volume of instruction sequences which are ensuring protection, as well as on the address labeling and generating system for the items composing the volumes of data. Safety in operation depends on the level of reliability and on the complexity of the volumes saving routines, as well as on the capability of the software product of handling the reiteration points. This characteristic is directly correlated with the efficaciousness of the product; a software product is the more efficacious as its safety in current use is greater, the economy of the consumed resources resulting from the reduction to minimum of the reiteration operation due to accidents in the handling of the volumes of data.

Safety in operation is also given by the extent to which the software product does not permit the unauthorized access of other programmes to its components and to the volumes of data it operates with, ensuring the integrity of the respective product. The evaluation of the safety index resides in estimating the probability of the event occurance which corresponds to the unauthorized change in the volume of data and in the programme components. Therefore, it is necessary to analyse the activation frequences of the processing functions and the situations which may engender unauthorized interventions.

3.5. Stability

Stability is the quality characteristic indicating "the resistance" of the software product to the effects generated by a change both in the initial data and in the instruction sequences making up the software product.

Stability at an instruction sequence level, stability at a module level and stability at a software product level are established according to the level at which the change is made. Stability can also be seen from the point of view of the performances of the software product and also from that of the types of processings which can be operated. The more stable the programme the smaller the effect of the propagation of the changes or of the errors in its components. The stability level of the software product is scheduled in the designing stage and it may serve in estimating the necessary effort for the changes to be operated for ensuring the maintainability and adaptability of the software product. The experimental evaluation of the stability of the product in current use requires a big volume of data and a long interval for its creation. The stability of the product influences the level of reliability, maintainability and adaptability. Modularity and structurability influence in a directly proportional way the level of stability and complexity negatively influences the stability level.

4. QUALITY CHARACTERISTICS CORRESPONDING TO THE REVISION OF THE SOFTWARE PRODUCTS

4.1. Maintainability

A software product has the characteristic of maintainability at the extent to which it permits a rapid and easy updating so that its utilization in good conditions should be continued. Updating represents a totality of the operations performed at the level of the software product, at the module level, at the instruction sequence and at the instructions level, respectively. These operations are paddings, eliminations or modifications of modules, sequences, instructions, operands, respectively. Updating aims to carry out the processing functions in best conditions (maintainability) and also to achieve new processing functions (adaptability). In the first case, changes are operated into the existent structure in order to increase the performances of the software product or into the structure of data to facilitate their gathering and storing. Changes can also be operated into the existent processing functions for harmonizing them with the new processing demands put forward by the end-users. In the second case, the software product is endowed with new processing functions or changes are operated permitting the use of the product on a different type of computation system as well.

Maintainability, adaptability and portability determine the flexibility of the software products.

The maintainability level is established on the basis of data obtained in the designing stage (control flows, data flows), in the testing phase, in the current use stage (intervals for locating the errors, intervals for making corrections, intervals for operating changes).

The evaluation of the maintainability level is based on data obtained indirectly by examining the influence of the changes on other quality characteristics or attributes, such as for instance, the complexity, linearity, structurability and testability of the product. The use of experimental methods requires the recording of data on nearly the whole life cycle of the software product. In case there are programme generators or languages oriented on the design of software products are used, the maintainability level is calculated automatically, with the obtaining of each updated version of software product.

The maintainability characteristic is influenced by the quality of the reference material accompanying the product, the quality of the source text, by the stability, testability and extensibility of the respective product. A growing maintainability is obtained by constructing, from the very beginning, selfdocumented, modulated programmes, using the structured text and by ensuring a complexity level strictly correlated with the processing function fulfilled by the software product.

4.2. Adaptability

A programme is adaptable when it permits the integration of new processing functions, on the one hand and, when it permits the operation of changes in activating the sequences leading to the growth of the product

performance, on the other hand. Whereas modifiability is the quality characteristic correlating the volume of changes with the effort required by these changes, adaptability is the capability of the software product of exchanging instructions in order to increase the performances of the product in current utilizations and of integrating new processing functions which correspond to the demands which have appeared after the product entered current use.

Adaptability is directly influencing the reliability of the software products; it is influenced by the stability level and by the complexity level of the software products.

The evaluation of adaptability is based on including into the calculus the number of opening points existing in the programs and the combinations of activating the processing functions. Experimentally, adaptability is estimated by using series of data which are recorded on the greatest part of the life cycle of the software product.

4.3. Clarity

A programme is considered to be impure when its sequences contain instruction which are not necessary, without influencing the quality and quantity of the final results.

A programme is considered clear when the effort made by programmers for identifying the significance of its instructions and sequences is minimum. The clarity of the programmes is obtained as the source text is being elaborated instruction by instruction. Clarity is influenced negatively by the volume of impurities appearing in the programme. Clarity is evaluated either as an effort to decipher the programme or as a modality of conceiving the structure of the instruction sequences.

This characteristic may be outlined as changes are operated into the components of the software product or as new versions are being elaborated. Clarity influences positively the level of the maintainability, adaptability and portability, of the programmes and, it is influenced by the level of the legibility characteristic. The complex problems must be characterized by an increased clarity level.

4.4. Legibility

Although the programmes are elaborated for compilation or assemblage, the text resources represent the raw material for operating changes into the programmes. Legibility is the quality feature permitting the identification of the syntactic units of the programming languages of each software product. Legibility renders the aesthetic aspect of the programme and it is one of the concrete forms of achieving the design of a programme. Legibility depends on the constraints of the programming languages to be used. For each type of programming language it is necessary to establish the quantification of legibility. Legibility differs from clarity by the fact that it takes into consideration the necessary effort for identifying the programme's syntactic units, whereas clarity takes into account the necessary effort for understanding

the significance of each sequence in the programme from the point of view of data processing. Legibility and clarity are two characteristics whose grades must have high levels in the case of complex programmes. The estimation of the legibility level requires systematic measurements whenever the source text is used by other programmers.

4.5. Linearity

A structure associated with an instruction sequence is considered linear if the component instructions are operated one after another in the order of their appearance in the structure without any conditional or unconditional transfer instructions should exist.

A programme is absolutely linear if the sequences it is composed of by concatenation are leading to a linear sequence. The linearity degree is influenced negatively by the share held by the transfer instructions. The linearity feature is influenced by the programming techniques which are used; one of the objectives of structured programming is the implementation of programmes with a high linearity level. The evaluation of linearity is based on the source text which permits the automation of the calculus for the level of this characteristic. The increase of linearity results in extending the dimensions of the programme and in its growing clarity; the maximizing of linearity is diminishing the performances of the software product. By establishing conventions as to the meaning of the unstipulated transfers, the linearity level can be estimated differentiatedly from a programming technique to another.

4.6. Modifiability

A software programme has the modifiability capacity if the modules it is composed of have the capability of accepting changes in the conditions of maintaining the performances of the software product within the limits of an efficient use. Modifiability appears as a possibility of propagating the changes into the programmes so as programmes that had been adapted or transformed in view of maintaining them in operation but whose use is no longer efficient should become efficient. The changes into the programmes have a profound character aiming to attain the goal pursued (maintainability, adaptability) but, also to maintain their utilization at a high efficiency level.

The modifiability degree depends on the programming technique used in working out the software product and, moreover, on the diversity of the means at the programmer's disposal for making the changes in the programme, in the programme points and in the necessary way. Modifiability influences directly the maintainability and adaptability of the software products. To evaluate the modifiability degree data recordings are necessary especially in the testing stage of the software product, data to be completed as changes are operated into the structure of the software product.

The inclusion into the software products of the reusable components entails an increased level of modifiability by reducing the share of the components which can be changed; the reusable components accept in most of the cases the changes in the form of input data.

5. QUALITY CHARACTERISTICS OF THE TRANSITION PERIODS FOR THE SOFTWARE PRODUCTS

5.1. Portability

The constructive differences of the computing equipment determine particularities in implementing the programming languages; the operating systems have commands specific to each type of computer.

Portability is the qualitative characteristic of the programs to be run on several types of computers. The programmes are considered portable either if they can be implemented on several types of computers, directly, without any changes being operated or, if the execution on other types of computers requires changes, characterized by a diminished programming effort. The programming effort for ensuring portability must be much smaller than the effort required by the rewriting of the programmes, for attaining the same objective.

The portability of the software products is influenced by the implemented programming languages, the types of files one can operate with, the constructive particularities of the computing systems, the types of constants which can be represented, etc.

The portability level is given by the ratio between the complexity of the modified sequences and the complexity of the programme which has become portable. The portability of the software products can be achieved by the separate elaboration of the components which are greatly influenced by the passage to another type of computer, the construction of programme version generators which have as input data the software products whose portability must be ensured and the parameter whose value indicates the type of computer on which the software programme to be generated will be run. The portability level influences the performances of the software products as well as their maintainability.

5.2. Integrability

The software products are constructed for fulfilling, initially, distinct processing functions. In case the initial data are proper and the final results are used in decision-making, the software product is considered independent. In case the software product is seen as part of a complex system of automatic data processing, the problem of the integration of the component into the system and the analysis of its integration capability is appearing. The integrability characteristic reveals, on the one hand, the capability of the soft-

ware product of using as input data files obtained by other software products and, on the other hand, of providing other software products with the files created by it.

The integrability of the software products is evaluated by the effort required by the changes in the instructions sequences for accepting as input data the files provided by other software products. The effort determined by the change in sequences for the creation of files meant for processing by other software products is also taken into consideration.

Integrability can be evaluated by analysing the adaptation capacity of the software product by introducing, as new processing functions, components which change the input and output of data for the software product. Integrability is influenced by the levels of modifiability, adaptability and modularity. The high complexity software products must be endowed with a high integrability level, too.

6. ATTRIBUTES OF THE SOFTWARE PRODUCTS QUALITY

6.1. Testability

Testability is the quality attribute permitting the users to underline as many variants of problems as possible which can be solved with the aid of the respective software product. Testability is emphasized by using the control example existing in the reference material accompanying the software product with the user, mainly by using that part of the reference material which describes the using facilities of the product. Testability implies the controlled discontinuance of operation in case the problem data activate inadequate processing functions or they are surpassed by limit values of the parameters defining the type of the software product. The evaluation of the testability level is achieved either on the basis of the structure of the software product or by analysing the behaviour of the product to the tests made by various end-users. Having an increased robustness, a software product also shows a high level of testability. Sometimes, users may reveal combinations of processings which had not focused the designers' attention and, they can also reveal situations in which the product is not performing correctly or completely data processings described according to the documentation. By a permanent dialogue between the users and designers of the software product, changes are operated which are leading to the increased level of testability of the software products.

6.2. Completeness

A software product is complete to the extent at which all parts are potentially activable and each of the parts is fulfilling processing functions given in the programming specifications. Completeness is established by analysing the source text in parallel with the reference material and with the results obtained by running a set of initial data. The absence of completeness is signaled out by the users as they are incapable of solving problems

for which the reference material does not mention any constraints. A programme which has proved its correctness is, at the same time, a complete programme. The completeness degree is expressed in per centage; completeness influences the levels of reliability and portability of the software products. In case the informatic systems are implemented in stages, the software products are constructed in the form of variants; the passing to a new version means to add processing functions existent in the initial reference material, in front of which mention is made that they were not yet implemented. In contradistinction to adaptability, the functions forming the object of completeness are not new; they are envisaged upon designing the software product but their implementation is spaced out according to the putting of the software product into operation.

6.3. Interoperability

The interoperability of a software product represents its capacity of coupling with other software products. Interoperability determines the effort to be made by programmers for making a software product become, by coupling, part of another software product or the software product should include as a component part another software product. This attribute permits the complete reuse of a software product for the carrying out of another complex application. The interoperability level is influenced by the modularity degree with which the software product is conceived. The coupling of the software products is achieved directly or by using interfaces. In both cases, the transmission of the parameters observes the requirements in connection with the parameter standards.

6.4. Generality

Generality is the attribute outlining the extent of the processing functions, the variants of the problem which can be solved, the particular cases and the maximum dimensions which can be taken into consideration. The generality of the software product is presented to the users through the multitude of the facilities explicitely contained in the reference material accompanying the software product. The increase in the generality level of the software product has a negative influence on the efficiency and reliability of the respective software product. Having a high level of generality, the software products must have a high stability level; the increase in the generality level entails an increased level of complexity for the considered software product.

6.5. Consistency

A software product has the attribute of consistency to the extent at which its components are performing uncontradictory processing functions and are based on uniformization in using the symbols and rules of constructing the identifiers, the labels and generally, of the homogeneous sequences.

The internal consistency of the software product is revealed through the analysis of the source text and its evaluation is made automatically. The external consistency is given by the uncontradictory behaviour of the product. The evaluation of the external consistency is made after the test is finished or after the correctness of the programme is checked. The programme for which the correctness characteristic has been established is at the same time a consistent programme. Consistency determines influences on the reliability and maintainability levels. Consistency may be increased by introducing standards into the construction of the modules and by using developed programming techniques.

6.6. Complexity

A graph can be associated with each programme, a graph in which the nodes correspond to the instructions and the arces show their execution order. This graph determines the structure of the programme. Complexity is an attribute allowing both for establishing the differences among the programmes structures and for hierarchizing the programmes according to the structure of each programme. The complexity of the programmes can be determined if the frequency of the instructions appearing in the programmes is analysed. The evaluation of the software product complexity is possible either by starting from the source text of the product or by starting from the graph associated with it. The estimation of the software product complexity allows for scheduling the necessary resources for the implementation of its components by stimulating the achievement of complex software products, equilibrated in point of the dimensions and processing functions. The complexity level influences negatively the reliability, maintainability and stability of the software products.

6.7. Flexibility

A programme is flexible if it implies an ever more reduced number of constraints with the users. These constraints refer to the description of data for the problems to be solved and to the way in which the final results will appear. Flexibility is dependent on the volume of options at the users' disposal, on the multitude of combinations which can be generated by using parameters. Flexibility is also dependent on the possibility of obtaining complete results even if they are incorrect, in case the initial data contain errors. The programs with a high reliability level are at the same time flexible programs. Flexibility influences the level of the programmes complexity; the increase of flexibility is accompanied by a steadier growth of the complexity of the respective programme.

The programmes with a high modularity level also have a high flexibility level. Flexibility can be seen from the programmers' view as the capacity of the programmes of taking over new processing demand which had not been envisaged. The flexible programmes require minimum efforts for adaptation; the new processing functions correspond to new values for certain parameters of developments which belong to the same classes of orders for which programme components have already been elaborated.

6.8. Structurability

The homogeneous parts or those which fulfil a distinct processing function are classified into sequences, subprograms, sections, modules, etc. The structurability of the software program is an attribute by which a distinct character is ensured to the entities of the respective product. The number of levels on which decomposition is fulfilled must be optimal. Structurability permits to identify the algorithm steps as well as the fundamental structures within the programmes and also between the programmes. The programme will appear only as an assemblage of fundamental structures; the software product will also appear as an assemblage of distinct components-modules. Structurability implies the establishment of the position of each component making up the software product and of its connexions with the other components. The quantification of structurability may be based on the number of levels on which the components are nested and the number of connexions between the components. Structurability influences the modularity of the software products.

6.9. Modularity

The module is the component that fulfils a complete processing function. The modularity attribute of the software product is given by the capacity of the product of being decomposed into modules and of establishing their interconnections. Structurability is the attribute rendering the order within the module; modularity is the attribute describing the order within the product made up of modules. A software programme has a high modularity level if a minimum effort is required for identifying the processing functions according to standard rules. The modularity level is established already in the designing stage of the software product. The evaluation of this level is based on the source text of the software product. Modularity influences nearly all the quality characteristics; it is influenced by the complexity of the programmes.

7. MANAGEMENT OF SOFTWARE PRODUCTS QUALITY

The knowing of the quality characteristics and of the techniques permitting the implementation of software products with the quality standard close to the scheduled level represents the basic target of the programming activity.

The management of the software products quality includes all aspects pertaining to scheduling implementation, optimizing, checking and measuring the levels of the quality characteristics and attributes of the respective products.

The establishment of evaluation methods for the level of each characteristic or each attribute creates possibilities of introducing standards and of analysing the software products which are constructed. By gaining experience and by analysing the interactions between the quality characteristics, instruments can be created which permit to construct software products with a still higher qualitative level.

The ensurance of the scheduled level of the software products quality by correlation with the costs implied by the necessary resources to be used calls for the achievement of an equilibrium between the technical performance of the product and its cost, influencing positively the economic efficiency of the current use of the software products.

For a unitary qualitative analysis of the software products, the modality of data gathering and the calculus algorithm of the level, as well as the limits permitting the inclusion of each software product into a certain quality class are established for each characteristic or attribute. The indices used in the calculus of the levels for the quality characteristics are organized in *a system just as the quality characteristics and attributes represent a system. The hierarchization of the system components* depends on the priority character of the objectives to be implemented for each software product, at a given moment.

CONCLUSIONS

The analysis of the qualitative sides of the programming activity is reflected in the implementation of software products with high performances. The possibilities of controlling the way in which the software products are implemented, qualitatively, depend on the automatic gathering of data on the respective products, on their behaviour throughout all the phases composing the life cycle and the estimation, automatically too, of the indices for the levels of the quality characteristics and attributes.

REFERENCES

BAUER F. L. (publisher): *Software Engineering*, Springer Verlag, Berlin, 1977.

BOEHM B., W., and others: *Characteristics of Software Quality*, North Holland Publ. Co., Amsterdam, 1980.

COWELL W. (publisher): *Portability of Numerical Software*, Springer Verlag, Berlin, 1977.

FITZSIMMONS A., LOVE T.: *A Review and Evaluation of Software Science*, Computer Surveys, vol. 10, no. 1, 1978.

GHEORGHIU AL., BARON T., BARON C.: *Măsurarea, analiza şi optimizarea calităţii produselor industriale* (Measuring, Analysing and Optimizing the Quality of Industrial Products), under print.

GORDON R. D.: *Measuring Improvements in Program Clarity*. IEEE Transaction on Software Engineering, vol. SE-5, no. 2, 1979.

HARRISON W., A., MAGEL K., I.: *A Complexity Measure Based on Nesting Level*. SIGPLAN NOTICES, vol. 16, no. 3, 1981.

REES R., L., D. (publisher): *Software Reliability*, INFOTECH., 1977.

WESTLEY A., E., (publisher): *Software Testing*. INFOTECH, 1979.

* * * *Normativ privind acceptarea, testarea, experimentarea și omologarea produselor program de uz general.* (Norms on Accepting, Testing, Experimenting and Homologating Software Products of General Use). ICI, 1981.

BARON T., BALOG AL., IVAN I., TRANDAFIR R.: *Un cadru conceptual pentru evaluarea calității produselor informatice.* (Conceptual Framework for Evaluating Informatic Products Quality), article under print.

LE SYSTEME DES CARACTERISTIQUES DE QUALITE DES PRODUITS PROGRAMME

Les auteurs présentent les modalités d'identifier et de classifier les caractéristiques et les attributs de qualité pour les produits programme. Aussi, pour chaque caractéristique ou attribut on présente les traits principaux, certaines considérations concernant la commensuration des niveaux et le mode d'interconditionnement avec les autres caractéristiques ou attributs. Dans a partie finale on approche certains aspects de la gestion de la qualité des produits programme.

СИСТЕМ КАЧЕСТВЕННЫХ ХАРАКТЕРИСТИХ ПРОГРАММ

В работе представлены способы идентификации к классификации качественных характеристик программных изделий. Для каждой характеристики приведены основные черты, некоторые соображения, относящиеся к измерению уровней, и способ взаимодействия с остальным характеристиками. В заключении, авторы подходят к некоторым аспектам управления качеством программных изделий.

5.01.1982