QUALITY AND PRODUCTIVITY: CONTROL OF SOFTWARE SYSTEM ELEMENTS

João Bosco Schumam Cunha e Tatuo Nakanishi
INPE - National Institute for Space Research
12225, São José dos Campos São Paulo, SP

1 INTRODUTION

Quality and productivity require constant care during software development, especially when large systems are involved. Since the last decade, several quality attributes, such as alterability, usability, reusability and efficiency of development, became additional requirements to correctness and system performance [1]. The evolution of techniques, methods and tools for software development have improved the quality and productivity. However, current conditions point to a set of problems that are yet to be solved [3,5]. Some problems, as the difficulty to control multiple versions of a program module, would not appear in the development of a small system, but for large systems they constitute a decisive factor concerned with software quality and productivity.

The size of large software systems, such as in an artificial satellite control system, could reach hundreds of thousands of code lines, thousands of program modules, and the development team could be composed of tens of people. In this type of development a module written by a group could be reused by other groups at other parts of the system; this module could be modified during the development phase and consequently several other modules need to suffer modifications too. Items, such as a queue message used by several modules, could be modified too, and in this case it is necessary to recognise all involved modules, and if necessary, to modify them. In the same way, the documents and several items could be modified in order to undergo modifications when other elements are altered [6].

Besides the necessity to control the software units and documentation, this article postulates that several items such as parameters of procedure, logical names, event flags, etc, need to be verified and controlled to guarantee their quality (correctness, consistency, etc). Only after analysis and approval, modification and deletion may be performed on a controlled element in order to have its quality maintained.

The necessity of control extends over the operation phase so that the quality level

will be preserved during maintenance activity. On the other hand, the relaxation of the control activities would permit early degeneration of quality and drastic reduction of system life. Hence, the investiments with the implementation of this control, essential during development phase, could give positive and continued return during a system's life because the maintenance activity can be executed in efficient and reliable way.

In order to implement a control scheme, a set of measures to set the system development process are necessary. Among them we can cite: To define the elements to be controlled; To organize the development team and the development environment; To create a directory structure and to discipline its use; To adopt a disciplined utilization of computational resources; To define procedures to control the elements of the system; To define resources and procedures to system integration; To utilize automated tools to perform the control of the elements; To use defined procedures for maintenance activities.

One such a scheme was used in INPE's Satellite Control Software System.

2 THE CONTROLLED ELEMENTS

The elements produced during the system development are checked in respect to their quality before becoming controlled elements. As controlled elements they are ready to be read or consulted, but they may not be modified without previous authorizations. In most papers [2,4,6,7], the referred elements to be controlled are products (units of software and documents) and not parts of them. In this work other types of elements are considered to be controlled, i.e. items and relationships. An example of a set of controlled elements for a large system whith concurrent tasks is shown in Table 1.

It is worth observing that, in addition to traditional programs, routines and documents, for present purposes the product type includes data files, command procedures and include files. Items are considered either as elementary parts of products, or as groups of elements. Process of Data Flow Diagrams, Event flags and Global variables are examples of items, and in this case, they are parts of documents and programs. Event Flag Clusters and Subsystems are items that examplify, respectively, the Event Flags group and the Tasks group. Relationships are those elements that associate products to products, products to items and items to items. A program call (CAl) to a subroutine, or a program write (WRI) in a mailbox are examples of relationships. Items are important to maintain coherence of products, and relationships are important to maintain coherence among all items and products. The definition of which elements should be controlled depends on the system characteristics, in addition to the characteristics of the process used for its development.

After verified and approved, products are stored in Controlled Libraries. Registers of all elements, products, items and relationships are maintained in a database called

Database of Controlled Elements - BDEC. In this way, the control of the BDEC and the Controlled Libraries are an important part of elements control.

Products	Itens	Relationships
A - Data files	DF - Data flows	CAL - Call
F - Functions	EE - External entity	DEL - Delete
I - Include files	DP - DFD process	WRI - Write
O - Command procedures	DS - Data store	WAI - Wait
P - Programs	ER - Entity (ER)	USE - Use
S - Subroutines	RR - Relationships (ER)	CRI - Create
SSD - Specification and analysis	AR - Attribute (ER)	SOF - Switch off
SDD - System design	SU - Subsystem	GEN - Generate
ITP - Integration and test plans	TA - Task	REA - Read
	MQ - Message queues	SON - Switch on
UMD - User manual	SA - Shared area	MAP - Map
IMD - Installation manual	SI - Signal	CMP - Compose
ITD - Integration and test report	MS - Syncronism message	SIM - Simulate
	MD - Module	TST - Test
	CB - Common blocks	
	EF - Event flags	
	CE - Cluster of event flags	
	GS - Global sections	
	JB - Jobs	
	CS - Chain subsystem	
	KB - Keyboards	
	NL - Logical names	
	RP - Routine parameters	
	GV - Global variables	

Table 1: Example of Controlled Elements

3 DEVELOPMENT ENVIRONMENTS

A set of different activities need to be performed during development and maintenance of a software system. The large quantity and the variety of these activities for large systems must be executed with discipline and control. Environment definition for system development is one feature for the organization of activities, and if performed suitably, will form the basis for the elements control scheme. Each of these environ-

ments groups activities that are related by objectives, or that belong to a part of the development/maintenance process.

SYSTEM ENVIRONMENT - The following activities are performed in this environment: System requirements specification; Definition of environments and development process; Creation of norms and standards; Functional specification; Data specification; Split of the system into subsystems; Specification of logical interfaces between subsystems; Approval of subsystems specifications; Definition of the acceptance tests of the system; Approval of test plans; Approval of execution of modified controlled elements.

CONFIGURATION CONTROL ENVIRONMENT - The following activities are performed in this environment: Generation and maintenance of System versions; Maintenance of the Controled Libraries; Transference of elements between environments; Control of the Database for Controlled Elements (BDEC).

QUALITY CONTROL ENVIRONMENT - The following activities are performed in this environment: Verification of produced elements in accordance with specifications, norms and standards adopted; Verification of consistency and completeness of the elements' registers that are written in the BDEC.

IMPLEMENTATION AND DESIGN ENVIRONMENT - Design and implementation activities of a large system would be performed in distributed manner at more than one integration and design environment. In each of these environments, one subsystem is developed in order to facilitate the element control. The following activities are performed in this environment: Subsystem specification; Subsystem design; Planning tests of subsystems, tasks and modules; Implementation and test of software modules; Documents elaboration; Modification of software and document during the development and integration phases; Modification of software and document during maintenance activities.

SUBSYSTEM INTEGRATION ENVIRONMENT - In order to test and to integrate one subsystem, the other subsystems and external entities are substituted by simulators. These simulators are, essentialy, units of software produced for this purpose, and have similar interfaces to the ones used by the subsystems and external entities simulated. In this environment each subsystem is tested under circumstances wich are similar to the real operation conditions. Simulators are necessary because subsystems and external entities are developed at their own rates, so that one subsystem may be ready earlier than subsystems and/or external entities the concluded subsystem interfaces with.

SYSTEM INTEGRATION ENVIRONMENT - The integration and tests of the system are performed in this environment based on their subsystems and external entities, individually tested in advance. The integration and test steps are performed by the System Integration Team according to a plan previously approved by the System Team. One

System version is created for each integration step by the Configuration Control Team.

PROTOTYPE ENVIRONMENT - Software prototypes may be developed for different purposes. One of them is to complete the specification of the interface between the user and the System. Another is the viability verification of operational and performance requirements. The advantage of prototyping activities in one closed environment is to avoid undesired interference to other activities, as the ones related to design/implementation.

4 COMPUTATIONAL RESOURCES

System development activities involve using different resources such as: secondary memory files, communication mechanisms, interruption mechanism, process control resources, utilities, etc. The use of these resources should be disciplined and should obey norms in order to allow the correct functioning of the control schemes. On the other side, the control activities use other resources that include: A software utility for files library control; A software utility for system versions control; A data base for controlled elements; A suport system for elements control. The automatization level of control activities reached with the use of these software utilities must consider cost-benefit aspect, system characteristics and particular conditions and constraints of the development.

The facilities required for a software utility for file library control include: To allow a file to be modified only if it was previously reserved; To inform which users are currently using which library file; To keep log transactions of library use; To keep versions of library files.

The facilities required for a software utility for system versions control include: To keep the relation of version of products that compose each system version; To define and generate system versions; To guarantee that any further alteration of product be reflected in its owners system version.

The main objective of a data base for controlled elements (BDEC) is to store registers of controlled elements, in addition to provide different types of reports and cross-references about these elements. The BDEC facilities support several activities such as: Quality verification of controlled elements; Consistency verification among products, items and relationships; Generation of system versions; Evaluation of impact over the system when a modification of one controlled element is required.

A suport system for elements control (SSCE) would have facilities such as: Generation and transmition of standard forms and other messages; Registration of forms flow and their status; Controlled transference of files and products; Partial verification of the product regarding its completeness and consistency.

5 PROCEDURES FOR CONTROLLING PRODUCTS

5.1 PROCEDURES TO BRING NEW PRODUCTS UNDER CONTROL

PRODUCTS DELIVERY - As soon as the development of a product is finished, it is delivered to the Configuration Controller by its producer. This is done using a SSCE form named Form for Product Delivery (FEP) whose content is a list of products and other pieces of information for control purposes. The product itself must be ready in the proper library of the producer before delivery. For example: the software products of type I,F,S,P and O are stored in the Software Products Library, the documents are stored in the Documents Library and the data files are stored on the Data Files Library. The following information must be stored at the BDEC before delivery: The products registers themselves; The registers of all other products and items referred to; The registers of all relationships with other elements.

PRODUCTS COLLECTION - At the Configuration Control Environment, the received FEP is registered in the FEP Control Map of the SSCE in order for its management to be initiated. The listed products are reserved at producer libraries and transferred to a working directory. While reserved, the products may not be altered by their producers.

PRODUCTS VERIFICATION - At the Quality Control Environment, the collected products are checked in respect to their quality such as: Consistency with corresponding specification; Internal completeness (procedure headers, comments, etc); External documentation completeness; Coherence with other controlled elements; Obedience to standards and norms. The SSCE and the BDEC are used to obtain lists of the elements related to each product, to compile the software products (type: F, P, S) and to register verification results.

MANIPULATION OF VERIFIED PRODUCTS - According their types, the Configuration Controller transfers those approved products to the proper Controlled Libraries and to the proper directory of the Integration Environment by using the SSCE resource. The products reserved at producer libraries are unlocked and marked as approved or not, according to verification results. One message about final results of the products that were delivered is passed on to the producer.

PRODUCER'S ACTIONS - The producer deletes the approved products from his libraries. Those products that were not approved are modified according to verification results and delivered again as new products.

5.2 PROCEDURES TO CONTROLLED PRODUCTS MODIFICATION

COMMUNICATION OF PROBLEMS/MODIFICATION - The teams communicate problems and modification necessities of controlled products to the Configuration Controller through the Problem Communication Form (FCP) of the SSCE. At the Configuration Control Environment one map of the SSCE is used for FCP management purposes.

FCP EVALUATION - The System Team analyses the problem/modification, its context, the extention of required modifications, the involved products, and determines the cost and impact of modification. The result of this evaluation determines whether the problem/modification must be resolved, postponed or refused. When a product needs to be modified, an SSCE form named Authorization for Product Modification (AMP) is filled out and the number of the AMP is registered at the respective FCP.

AMPs MANAGEMENT - The AMPs are registered and managed through a proper control map of the SSCE. Each AMP is sent to the team that will execute the modification and one message about the corresponding FCP is sent to the its original site. The product. object of the AMP is marked as ready for modificated and its status register is updated in the BDEC.

PRODUCTS MODIFICATION - The designated team would copy the products from the Controlled Library to his working area before executing the modification. As soon as the modifications are finished, the products are sent again to the Configuration Controller.

CONTROL OF MODIFIED PRODUCTS - One modified product is controlled in a similar way as a new product. In this case the control procedure contains additional actions such as: To register the AMP number on the corresponding FEP; To verify if the modifications denoted on the AMP were executed; To update the AMPs control map.

6 CONCLUSION

This paper points out the importance of controlling the elements of a large software system when quality and productivity are desired. In addition it is remarked that only to control products is not enough: it is also necessary to control several types of items and relationships. Moreover, in order to implement the contml scheme it is necessary to take a set of precautions, so that the control becomes an essential part of the development process, and that it be applied during the remainder of system life cicle.

In the case of INPE's Satellite Control Software System development, the control scheme produced very good results. The guarantee for quality of end products and the productivity of quality control were incremented by a mechanism for products reserva-

tion and by automated preliminary verification of elements consistency. The test and integration phase were finished in the predicted time and the automated generation of system versions, as well as the consistency control of interface elements contributed to this fact. The automated accessing of the elements' registers simplified the impact analysis of modifications. The organization of environments contributed to elements control and to achieve general quality and productivity of development; for example, it was very important to correctly distribute the work, the responsability, and to discipline the activities. The scheme implementation included two DEC software packages, one for the libraries control and other for the versions control. Two other packages developed by INPE's team integrated this scheme too, the Data Base for Control Elements (BDEC) and the Suport System to Control Elements (SSCE). The utilization of these computer resources yielded a satisfatory level of activity automatization, although some activities that were manualy performed, such as the comunication between environments, could have been automated as an SSCE subsystem.

The design and implementation of a control scheme for distributed development environment that includes a set of workstations is being investigated by the authors. In this endeavour, besides adjusting the scheme according to the new resources, the increase of the automatization level is a primary target.

References

[1] ACM/SIGMETRICS; ACM Workshop/Symposium on Measurement and Evaluation of Software Quality, Proceedings; *ACM/SIGMETRICS* 10(1): New York, 1981.

[2] E.H. Bersoff; Elements of software configuration *IEEE Transactions on Software Engineering*, 10(1):79-87, Jan. 1984.

[3] B.W. Boehm; Software risk management: Principles and practices, IEEE Software, Vol. 8, no. 1, pp. 32-41, Jan 1991.

[4] S.L. Feldman; Make - A program for maintaining computer programs *Software - Practice and Experience*, 9:255-265, 1979.

[5] M. Kumar and J. Wong; Transaction models for design environments, J. Systems Software, Vol. 16, no. 3, pp. 219-228, Nov. 1991.

[6] N.H. Madhavji; Environment Evolntion: The prism model of changes, IEEE Trans. on Software Engineering, Vol. 18, no. 5, pp. 380-392, May 1992.

[7] S.H. Zucker and K.B. Christian; Automated configuration management on a DOD Satellite Ground System, IEEE AES Magazine Software, pp. 10-15, november 1986.