iipp publishing

Science of Maintenance

Journal homepage **www.m-sci.rs**

ISSN 2787-3072

2 (2022) 1

Software Maintenance Packages

Goran Vorotović, Miloš Januzović

Belgrade University Faculty of Mechanical engineering, Belgrade, Serbia

Abstract

The software life cycle as a concept, vague, and by no means understood regardless of its applicability in modern society remains as the "eternal child" of modern technology.

Taking into account the possibilities of modern software engineering and its effect and adhering to the basic principles of programming, user requests to improve and optimize software packages become legitimate and, above all, usable.

Practice has evidently shown that in the life process of software, its maintenance is a key component in the business process, interpersonal relationships and the functioning of the entire system as a whole.

The practical knowledge of the authors of the paper and their decades of work in the field of software maintenance have shown that maintaining something that works is incomparably more important than something new that does not have a clearly defined limit.

The experiences of the authors in the field of software maintenance in the past few decades are shown in the paper and as such, according to the humble opinion of the authors, they must be a significant link in the software life cycle.

Key words

Software engineering
Software maintenance
Software life cycle

Corresponding author

gvorotovic@mas.bg.ac.rs

Article info

Original Scientific Paper
Received 27.05.2022.
Accepted 29.07.2022
Online available 15.09.2022.
Open access article CC BY license

1. Introduction section

Systems can be natural or artificial. Natural systems arise along natural processes, while artificial or technical systems are treated as systems in which humans intervene in a natural order by applying pervasive technologies through components, attributes and connections between systems. The types and diversity of artificial systems are numerous and go beyond the areas of communication, defense, education, health, manufacturing, transport and others. The most common modern approach to system engineering that permeates most standards is presented in the following figure.

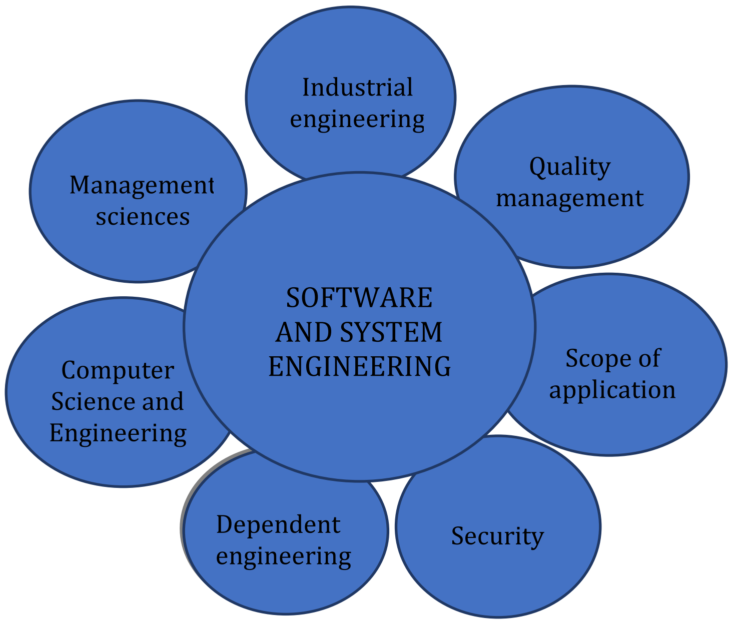

Figure 1. System engineering

The time in which we live and the time that is to come imposes on us the indisputable fact that non-application of engineering in all phases of the system is expensive and very disruptive.

It is indicative that comparisons and permeation to the level of identity of system engineering and software engineering are becoming more frequent. The transfer of the physicality of the phenomenon into the virtual environment of information technology accelerates (time is, as we will see in the definitions that follow, a key factor in increasing the effectiveness of systems and a necessary factor of primarily technical systems) to unimaginable limits of the whole system in terms of its functionality. The standard approach to system and system engineering within this categorization takes shape:

"The engineering process consists of processes that directly specify, implement or maintain a product, its relations to the system and user documentation. In cases where the system consists only of software, this process applies only to the construction and ***maintenance*** *of such software. The development of systems and software for the wider market or users can be presented in the form of the contractor's marketing function, from which the requirements are defined and determined."*

The final transfer of requirements defined by a certain method to the heart of the system is done through certain engineering processes such as analysis, design, testing, implementation, post-processing and release of the system by defining final reports. This process is performed by applying some of the existing methodologies such as PSP (Personal Software Process).

This way of developing the system fits perfectly into the basic definitions of the physical and conceptual system, which say that the physical system is the one that manifests itself in physical form. It consists of real components and it is in contrast to the conceptual system in which symbols represent the attributes of the properties of the components. Ideas, plans, concepts and hypotheses are examples of conceptual systems.

The physical system takes up physical space whereas the conceptual systems are organization of ideas. One type of conceptual system is a set of plans and specifications for a physical system before it is created. The proposed physical system can be simulated in abstract form through a mathematical or some other conceptual model. Conceptual systems often play an essential role in the functioning of physical systems in the real world.

Therefore, through the previous definitions, a comfortable approach to the integration of system engineering and the increasingly present software engineering as a way to a modern and above all functional and efficient system is provided.

2. Information System as a Subsystem of Data Transfer to a Higher Level

The transformation of information from one such model to a higher level of information is also performed using a large number of appropriate methodologies. However, the basic preparation of the system is done by defining the need and elaborated requirements through a methodology similar to that described in the previous chapter.

In order to monitor the condition of components and systems completely, it is necessary to transform the obtained data into a higher level of information using the methodology of defining requirements and through appropriate documents (such as measures to be taken to improve software, processes, etc.).

This process of "transformation" of information to a higher level in modern systems engineering is performed using information systems.

The generation and binding of data is in direct connection with the definition of the requirements of the entire observed system, as well as in relation to the existing data of the manufacturer, the existing exploitation and the like.

The data collected in this way, together with all other data, enters the centralized database, from where, with the help of computational algorithms, defined requirements and goal functions, they are processed and they give reports adequate for the observed domain as a final result.

This data processing implies the existence of specialized software that connects the central database with the database of data collected from other locations.

Such processing results in data on the current state of the observed component, as well as on the measures that need to be taken to raise the situation to a higher level. On the other hand, it results in a certain financial assessment of the operation of the system and components, as well as the primitives of the expert system.

3. PSP

Software support is most often provided in the form of developed user applications designed to work in the network due to the complete connection of all segments of the organization that owns special motor vehicles.

The most expedient way to an efficient information system is to develop the system itself within the organization itself. Such a solution requires some of the universal software development and implementtation processes. One of the most used is the so-called PSP (Personal Software Process). The basic concept of the PSP process structure is shown in Figure 2.

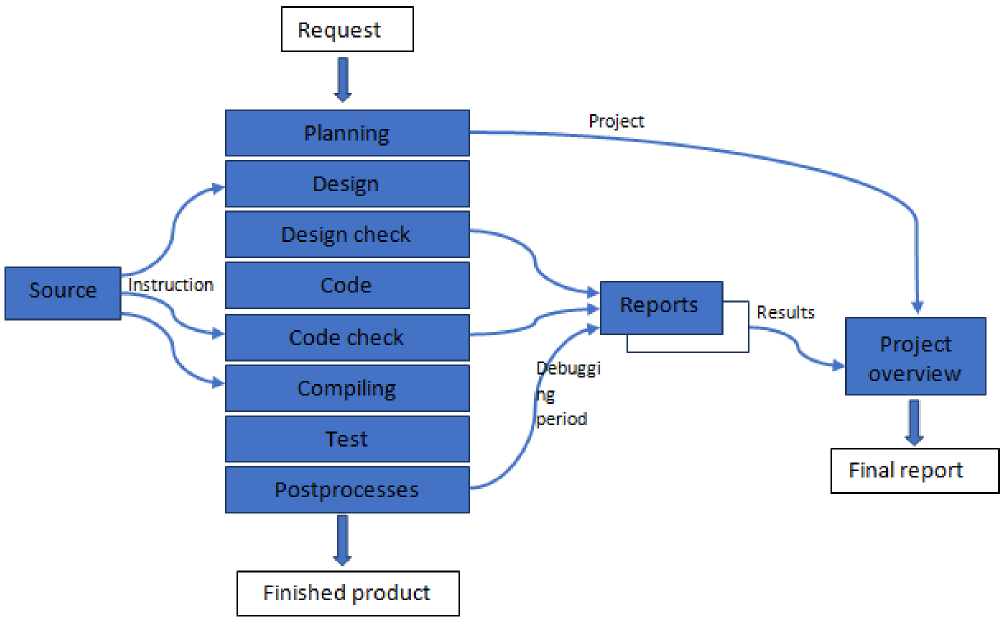

Figure 2. Personal Software Process

Table 1. An example of a simple PSP source

Phase number	Purpose	Leading towards the improvement of modular-level programs
	Input criteria	•Problem description •PSP1 Final report form •Template size estimate •Experimental estimation and current data size •Errors and reports •Standard error types •Interruption of observation (optional)
1	Planning	•Generating or use of request description •Use of methods to evaluate the required new or changed software modules •Completing the template size estimate •Estimate of required development time •Entering the planned data in the project final report •Completing the report over time
2	Development	•Program designing •Design implementation •Program compiling and detected errors repairing •Program testing and repairing the detected errors •Time reports completion
3	Post processing	Completing the final report with current times, errors and data sizes
	Output criteria	•Complete program testing •Complete final report with estimated and final data •Complete size of estimated templates •Complete test of template report •Report on all errors and time singularities

The beginning of the process is defining the request description. This source represents a "guide" for the described process as well as for the final project report in terms of recording and presenting project data. The engineering process consists of strict traceability of sources primarily to detect errors in a unit of time and then due to the necessary generation of error reports. At the end of the project, the results are summarized through the post-processing phase, errors in the unit of time are determined as well as the size of the software package, etc. after which these data are entered in the final project report. In the end, the final product is placed on the market released by the final project report.

Since there is no standard on the number of PSP processes, due to subjectivity that depends solely on the system engineer, it is recommended to use the PSP method in a series of seven consecutive versions of the process. These versions are numbered from PSP0 to PSP3 and are similar in terms of forms, reports, sources and standards as shown in Figure 3.

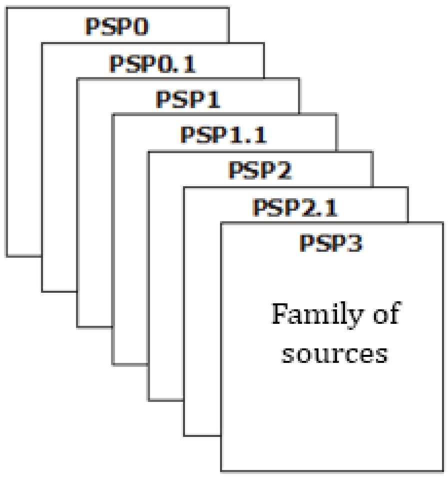

Figure 3.

Process resources define the steps for each part of the process, reports and forms necessary to record the situation and standards that guide engineers for further work. An example of a simple PSP source is given in the following table. It may be concluded that the purpose of these sources is to lead the system engineer to a consistent design of the system and its full understanding, which once again emphasizes the importance of requirements as input parameters of modern systems.

4. Maintenance process

The maintenance process contains the activities and tasks of the maintenance designer. The process is activated when the software product undergoes code and appropriate documentation modifications due to a problem or the need to improve or adapt.

The goal is to modify the existing software product while preserving its integrity. The process involves migrating and withdrawing the software product. The process ends with the withdrawal of the software product.

The activities contained in this section are specific to the maintenance process; however, the process may use other processes from JUS ISO / IEC 12207. If a development process is used (5.3 JUS ISO / IEC 12207), the term designer should be interpreted as maintenance designer.

The maintenance designer manages the maintenance process at the project level by adhering to the management process that is specified in this process;

- establishes infrastructure by adhering to the infrastructure process;
- adapts the process to the project using the adjustment process;
- and manages the process at the organizational level following the improvement process and the training process.

When the maintenance designer is a maintenance service provider, he performs the delivery process.

4.1. List of activities.

This process consists of the following activities:

- Process implementation;
- Problem analysis and modification;
- Implementation of modification;
- Maintenance inspection/acceptance;
- Migration;
- Software withdrawal.

4.1.1. Process implementation

The activity consists of the following tasks:

- The maintenance designer develops documents and executes plans and procedures for the implementation of activities and tasks from the maintenance process.
- The maintenance designer determines the procedures for receiving and monitoring reports on problems and requests for modification submitted by the user and establishes feedback with users. Whenever problems are encountered, they must be registered and entered into the problem-solving process (6.8JUS ISO/IEC 12207).
- The maintenance designer implements a configuration management process (6.2JUS ISO/IEC 12207) (or establishes an organizational interface with it) to manage modifications to the existing system.

4.1.2. Problem analysis and modification

The activity consists of the following tasks:

- The maintenance designer must analyze the problem report or request for modification in terms of its impact on the organization, the existing system, and the systems it faces in terms of the following:

  a) Type: for example, corrective, improving, preventing or adapting to a new environment;

  b) Scope: for example, modification size, cost, time required;

  c) Criticality: for example, impact on performance, safety or security.

- The maintenance designer disputes or confirms the existence of the problem.
- Based on the analysis, the maintenance designer develops options for implementing the modification.
- The maintenance designer must document the request for the problem / modification, the results of the analysis, and the implementation options.
- The maintenance designer must obtain approval for the selected modification option as specified in the contract.

4.1.3. Implementation of modification

The activity consists of the following tasks:

- The maintenance designer conducts an extensive analysis and determines which documentation, software units and versions should be modified. This must be documented.
- The maintenance designer enters the development process to implement the modification. The requirements of the development process are supplemented:

  a) by test and evaluation criteria that test and evaluate modified and unmodified parts of the system (software units, components, and configuration elements) and which must be defined and documented.

  b) by complete and correct implementation of new and modified requirements. In addition, it is checked whether the original, unmodified requirements remain unchanged. Test results must be documented.

4.1.4. Maintenance inspection/acceptance

The activity consists of the following tasks:

- The maintenance designer shall perform the inspection (s) with the organization that approved the modification in order to determine the integrity of the modified system.
- The maintenance designer shall obtain approval for satisfactory completion of the modification as specified in the contract.

4.1.5. Migration

The activity consists of the following tasks:

- If a system or software product (including data) is transferred from an old to a new work environment, it should be verified that the software product or data modified during the migration still complies with this standard.
- The migration plan is developed, documented and executed. Users are also involved in planning activities. The plan contains:

  a) Requirement analysis and migration definition;

  b) Development of migration tools;

  c) Software and data conversion;

  d) Execution of migration;

  e) Migration check;

  f) Support for the old environment in the future.

- Users are informed about migration plans and migration activities. Notices contain:

  a) A statement of why the old environment is no longer supported;

  b) Description of the new environment and when it will be available;

  c) A description of other available support options, if any, when support for the old environment is withdrawn.

- Parallel work of the old and new environment can be carried out to facilitate the transition to the new environment. During this period, the necessary training should be conducted, as stipulated in the contract.
- When the planned migration approaches, a notification must be sent to all interested parties. All relevant documentation on the old environment, registers and code should be archived.
- After use, an examination is performed to assess the impact of moving to a new

environment. The results of the inspection must be sent to the authorities for information, management and action.

- Data used or related to the old environment must be available in accordance with the requirements of the data protection contract and the checks applicable to the data.

4.1.6. Software withdrawal

The activity consists of the following tasks:

NOTE - The software product is withdrawn at the request of the owner.

- A software withdrawal plan that denies active support to organizations that develop and maintain software must be developed and documented. Plan activities must include users. The plan must contain the elements listed below. The plan must be executed.

  a) Termination of full or partial support after some time;

  b) Software product and related documentation archiving;

  c) Responsibility for any future support;

  d) Transition to a new software product, if it is applicable;

  e) Possibility to access archival copies of data.

- Users are notified of the software withdrawal plan and activities. Notices include:

  a) Description of replacement or upgrade with indication of availability date;

  b) A statement as to why the software product cannot be further supported;

  c) A description of other support options available when support ceases.

- The parallel operation of the retreating software product and the new software product is carried out to facilitate the transition to the new system. During this period, training of users must be provided, as stipulated in the contract.
- When the time comes for the planned withdrawal, a notice is sent to all those interested. All relevant development documentation, registers and code should be archived when appropriate.
- Data used or associated with the withdrawn software product must remain accessible in accordance with the requirements of the data protection contract and the checks applicable to the data.

5. Conclusion

The practical problems that the authors have had in their work on software maintenance so far have indicated that the components of migration and endless "withdrawal" of software from use are dominant aspects of the entire maintenance process. Experience has shown that in the software maintenance process the human component is most emphasized through requirements and optimization as input process parameters. In this sense, the opinion of the authors is that the software maintenance process must be given importance in the level of its development where the requirements and obligations in each phase of the software life cycle would be divided "into equal parts".

6. References:

[1] Dr Gradimir Ivanović, Dr Dragutin Stanivuković, Pouzdanost analiza i projektovanje TEHNIČKA UPRAVA SSNO, Beograd 1988
[2] Blanchard B.S., Fabrycky W.J. Systems Engineering and Analysis, Prentice Hall Inc., New Jersey, 1998.
[3] Todorović J., Inženjerstvo održavanja tehničkih sistema, JUMV, Beograd, 1993.
[4] Craig Larman, Applying UML and Patterns,
[5] Prentice Hall PTR, New Jersey, 1998.
[6] Watts S. Humphrey, The Personal Software Process, Carnegie Mellon University, 2000
[7] Č. Mitrović, B. Vasić, Informatika u mašinstvu, Mašinski fakultet Univerziteta u Beogradu, 2000
[8] Jugoslovenski standard, JUS ISO/IEC 12207, Službeni list SRJ, 1997
[9] ISO/IEC Standard 12207