Естественные и технические науки. 2021. №9 (160). с. 98-103
https://stepanovd.com/science/article/113-2021-4-requirements

ANALYSIS OF FUNCTIONAL REQUIREMENTS RELEVANT FOR DEVELOPING CORPORATE INFORMATION SYSTEMS

Dmitry Yu. Stepanov[1], Mikhail N. Statsenko[2]

[1] RTU MIREA, department of Corporate Information Systems, PhD, professor assistant MIREA

[2] RTU MIREA, department of Corporate Information Systems, 1st year MSc

Abstract: *implementation of complex information systems requires development a lot of software programs. Development might take long time because program must be analyzed, designed, built and tested. There are many different requirements applied to any program, however functional requirements typically are the same. Knowing and understanding such requirements will help in developing proper application and contribute project delivery in time.*

Keywords: *V-model approach,. software programs, program development, ERP-projects, control theory, system analysis, programming, feedback loop, evolution principal, combination of algorithms, organizational levels.*

I. INTRODUCTION

Corporate information system or so-called ERP-system is a complex application containing a set of program software. Hundreds of interconnected programs are to be developed from the scratch during ERP-system implementation project [1]. Content of each tool is problem domain oriented. Usually ERP-system is been implemented using V-model approach, mapping requirements and testing types together [2]. Business, user and functional requirements are identified to build further information system. First two types of requirements are to be defined and described by end users, however third one is common. Thus, if you are familiar with typical functional requirements it's easy to design and develop scalable software program, moreover it will save efforts and might decrease project duration.

II. STATE OF THE ART

There are many papers devoted to design and development of software programs, however it's more about standalone applications or tools but not complex information systems [3]. The feature of information systems is that each developed application is interrelated with the others and, of course, has impact on them. Any changes in application can ruin the rest. Another weak point of classic and modern papers related to program development [4-6] is that they are oriented for building methods and techniques, rather than design principals. The last one is crucial as there is responsibility differentiation in ERP-projects: technical consultants are in charge on designing a tool, programmer – it's building. Moreover, any changes in program code can be done only by developer. Some papers [7-9] describe agent and model-based approaches to design integrated architecture for a set of applications, however there is no explanation core and support parts of each program.

III. PROBLEM STATEMENT

The purpose of this paper is gathering functional requirements which can be applicable for developing and modification any ERP-system. To cover this topic following tasks to be solved: analyzing principals relevant for software programs development, exploring typical requirements for complex programs and estimating theirs's usage in real ERP-systems implementation projects.

IV. OVERVIEW

The main rule in programming is flow chart for program must be prepared first, then build can start. Why is it suggested this way? When one is creating flow chart lots of facts must be taken into consideration, thus understanding of program architecture and its' algorithms might change. In fact, one is planning but not building program software at this stage. The same situation can happen while designing and developing ERP-system. One must think about program screens, algorithms for population these screens, tables where data is stored and to be selected from. All these should be analyzed before development is started. There are a few subjects can be applied for program design and development. Paper [6] describes three main disciplines:

- Control theory.
- System analysis.
- Programming.

Control theory is telling how to handle systems, processes and objects. Originally theory considers technical systems, the ways to control it by giving orders and checking obtained results. The objective of management is to make system achieve state it's required. There are a few principals considered in theory to control systems: programmable, by compensation, by feedback loop and combined. Nowadays feedback loop concept is widely used in most of popular desktop programs.

System analysis can be considered as a method to solve complex problems and support decision making. It allows defining root cause of issue, analyzing different solutions and providing recommendations. There are consistency, unity, connectivity, final goal, equifinality, functional, evolution and ambiguity universal principals described in this discipline. The last three can be adopted and used for software program development as well.

Programming describes processes of data management. It can be divided into programming theory, explaining structure of programs, commands and parameters; methodology, grouping programming languages together; technology, containing ways how to deliver software; engineering, improving delivery process and program quality; tools, supporting development process. Principals of modularity as well as reliability, functional selectivity and default values are relevant for building complex applications.

A. Control theory and feedback loop

Let's consider principals related to program development within each subject. Feedback loop principal from control theory can be considered as a combination of object, person and feedback loop, linking together first two (fig. 1). The principal specifies following: if person wants object to achieve the goal, then input commands must be given. Object, taking into account the input from person, should process it and provide final results after execution to user by means of feedback loop. Having the results, user will take a decision what to do next.

Applying the principal to software development, object and person will be replaced by program and user correspondingly, feedback loop will remain the same. The concept make one think about issues can happen while program running and inform user about it. Thus, user knows if program has success or failure. For instance, if you did amendments in Word document and want to close it, Microsoft application will ask if save the changes before closing, besides most Android applications ask nothing at all.

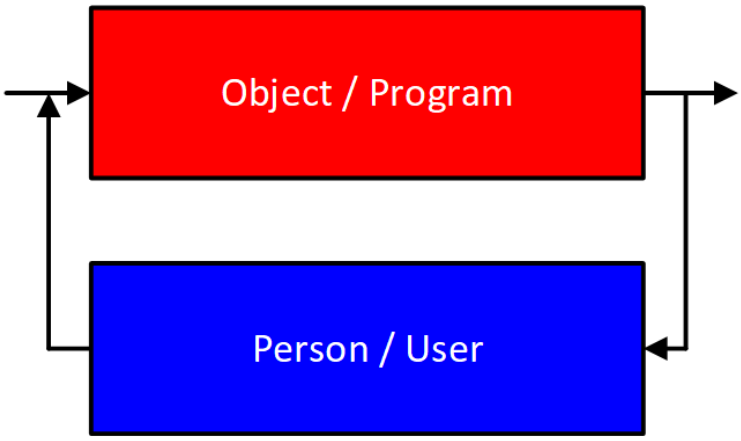

Fig. 1. *Illustration of feedback loop principal*

B. System analysis and evolution

During program designing we should consider functionality of software might be evolved soon. For example, application was originally built for one time run by two or three users, however later one decided to use it on a regular basis all over the company. Thus, while planning program architecture, you must design it the way to obtain easy changeability, extendibility and scalability. Evolution concept means if there is program developed for current customer, then it must be possible to utilize it for any other clients without changes or with small modifications (fig. 2).

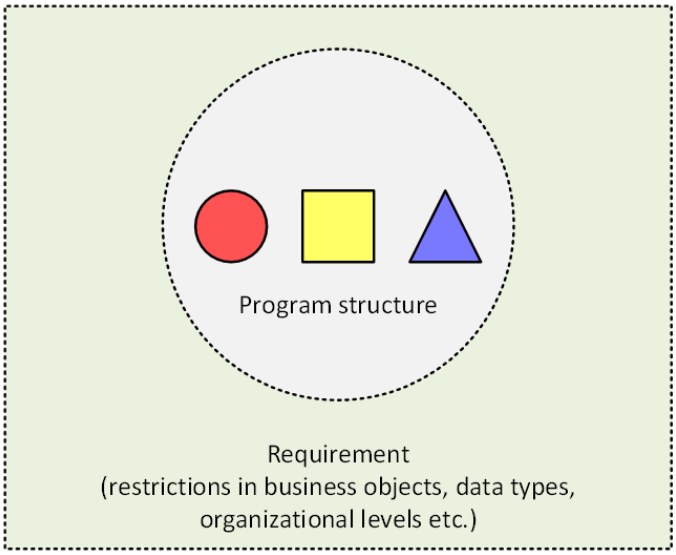

Fig. 2. *Graphical presentation of evolution principal*

C. Programming and reliability

In short, program can be considered as a combination of algorithms and structured data. Typically, application processes data, which can be stored in database tables, flat files or in program itself. The result of program work is updated data. Reliability concept means if tool processed data somehow, there should be functionality to revert changes. For example, vehicle can move straight, but it does not cover all possible directions, thus there is a wheel to go on the right, left and back. The principal is saying, if application, you have developed, is about saving data, there must be possibility to change and delete it, even if there are no requirements from user. Otherwise, if program performed and caused data inconsistency, it would be impossible to fix issues.

V. IMPLEMENTATION

We introduced subjects and its principals: feedback loop, evolution and reliability, which can be adopted for software development. Let's go into details and shape functional requirements per each principal. Later, we will prioritize them as per criticality for ERP-system implementation projects.

A. Evolution and naming convention, constants, organizational levels, authority checks

As we have discussed, program evolution is about forecasting software growing and making corresponding changes in application design before build. Naming convention is a set of rules how to name each developed object: program, function, parameter, table etc. Formula (1) represents example of random naming convention applied for program names in SAP ERP system:

$$ZAABBCCC, \quad (1)$$

where Z – constant, identifying it's customer but not standard program; AA – country code, for example, 'RU' for Russia; BB – application area code in SAP, for instance sales has 'SD' code, CCC – 10 chars to be used for unique program name.

Each application has constant values in program code. These values must be stored in the system centralized way. It's relevant for ERP-system implementation projects as there are hundreds of programs created from the scratch or modified. Usually the task is solved by creating single program or custom database table where constants per different pair of parameters will be maintained. SAP ERP system has standard table TVARV where constants can be created and updated.

If program is in line with naming convention, has centralized storage for constant parameters, then it's time to speak about user interface. Typical software program has three main screens in ERP-systems: selection screen, screens for selected and processed data [10]. Once program run, end user will see selection screen, where initial parameters can be specified (fig. 3).

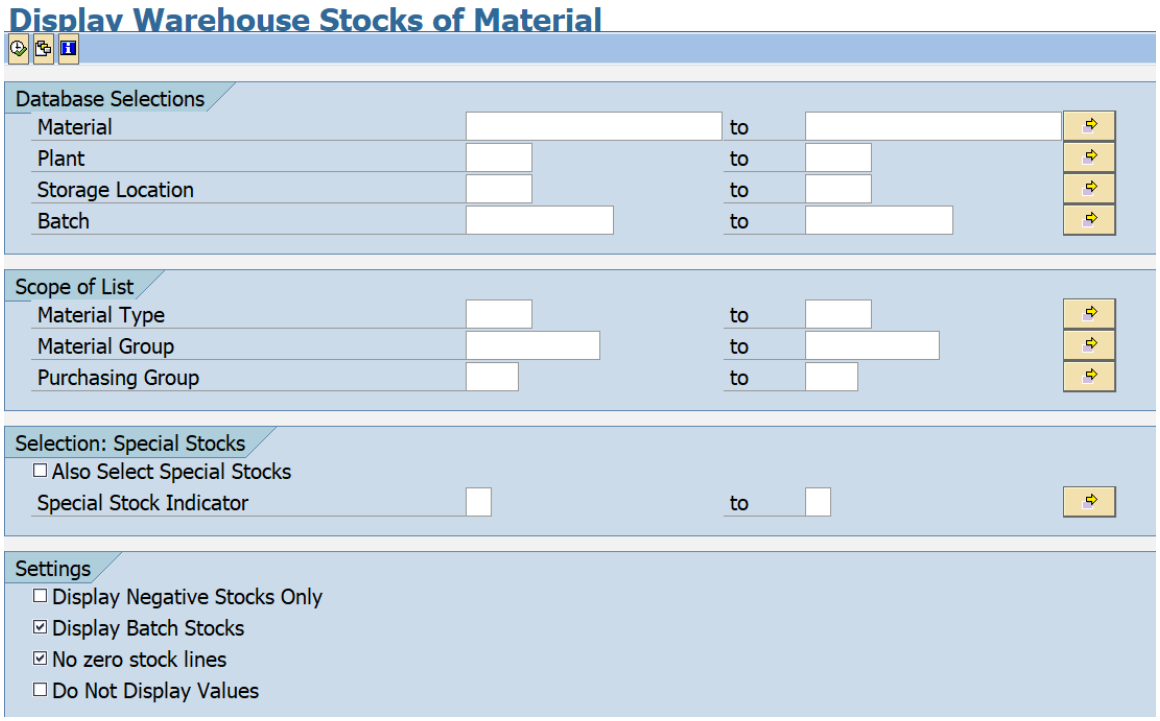

Fig. 3. *Example of program selection screen containing organizational levels in SAP ERP system*

After population selection screen data, next screens will appear. Putting fields for organizational levels on selection screen, will allow you to use the program in different clients and companies. At the same time, information must be stored in database tables per organizational level as well. Examples of such levels are company, plant, storage location names and any other business entities.

Organizational levels placed on program selection screen allows restricting data volume to be selected from database tables and displayed on the screens. Otherwise, program tries to select all information from database, which will cause time out error. Data selection must be aligned with user authorization. Modern ERP-systems has two kind of authorization checks: if user has access to run program and specify selection screen data per organizational level. Hence, end user will never see data relevant, for example, for the plant he does not belong to.

B. Feedback loop

The main idea of handling the program is that user knows what exactly happens now. If program catches error, then user must be notified about for further corrections (fig. 4). Otherwise, error will be identified only during detective analysis, but it's too late. Each program has impact on business data, thus each missed and unresolved issue induces system inconsistency. The earlier problem is identified, the better. Feedback loop in a form of notification is to be developed for both dialogue and background program modes.

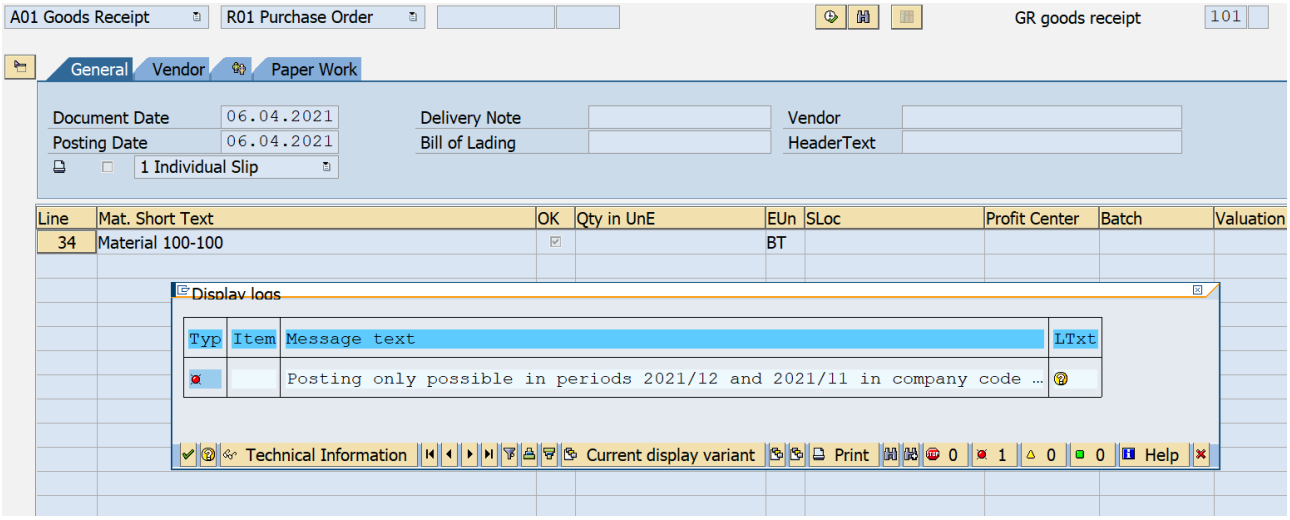

Fig. 4. *Screenshot from SAP ERP system displaying error message if something went wrong*

C. Reliability and results corrections

If you develop program, for example, to insert records to database table, you should also realize possibility to edit and delete these data. Extra functionality can be a part of developed program or reasonable workaround [11]. The matter is that, no one can predict issues can happen, thus there must be opportunity to discard made changes. For instance, SAP ERP system has completely different programs to add and edit business data, display and delete it.

VI. EVALUATION

There is three-layer landscape configured for ERP-system: development, quality and productive. Originally program is built in development layer, then to be transferred to quality system for user acceptance testing and to productive environment for real-time use. Software can be moved to productive layer only if testing performed successfully and there are no remarks to program code from customer development team.

Table 1. Usage of functional requirements in ERP projects

Subject	Principal	Functional requirement	Usage, %
System analysis	Evolution	Naming convention for all program elements	88
System analysis	Evolution	Constants to be stored centrally	75
System analysis	Evolution	Organizational levels on selection screen	100
System analysis	Evolution	Authority checks and data restrictions	100
Control theory	Feedback loop	Feedback loop via user notification	25
Programming	Reliability	Possibility to correct program results	0

Usage of considered above functional requirements was estimated empirically under eight SAP ERP implementation projects (tab. 1). Evolution based requirements are extensively used. In most of the projects customer development team will not allow transferring program to productive environment, if this principal is not realized in application. Feedback loop and reliability principals looks like 'Nice to have' functionality, software program can have no such requirements realized and be in productive layer. However, if issue happens, it will take too much more time to identity root cause and make corrections. As you can see from the tab. 1, most of utilized requirements are related to system analysis but not programming subject. It's specific of ERP-system implementation projects.

CONCLUSIONS

In this paper we have considered functional requirements applied for developing ERP-systems. Three subjects were analyzed to identify common requirements: control theory, system analysis and programming. Empiric estimation showed requirements under system analysis principals are commonly used in most of SAP ERP implementation projects. Next step for the research is investigating other subjects and its principals, estimating them in ERP implementation projects.

REFERENCES

1. Samara T. ERP and information systems. Integration or disintegration, John Wiley & Sons Limited, 2018, 145 p.
2. Sudakov V.A. Corporate information systems, MAI, 2016, 96 p. (in Russian).
3. Knuth D., The art of computer programming. Volumes 1-4A. Addison-Wesley Professional, 2011, 9998 p.
4. Popkov V.M., Sokol D.T., Chuguev M.V., Nedyalkov Yu.V. Complex program systems and problems while developing them. Scientific journal, 2017, № 1(14), p. 14-16 (in Russian).
5. Pfleeger S., Atlee J. Software engineering: theory and practice. Pearson, 2009, 756 p.
6. Richards M., Ford N. Fundamentals of software Architecture: an Engineering approach. O'Reilly Media, 2020, 432 p.
7. Petrie C. Agent-based software engineering. Proceedings of PAAM conference, Manchester, April, 2000, p. 185-202.
8. Jennings N.R. An agent-based approach for building complex software systems. Why agent-oriented approaches are well suited for developing complex, distributed systems. Communication of the ACM, 2001, vol. 44, № 4, p. 35-41.
9. France R., Rumpe B. Model-driven development of complex software: a research roadmap. IEEE Proceedings of international conference on software engineering, 2007, p. 83-101.
10. Stepanov D.Yu. Preparing functional specification documents for corporate information system development on SAP ERP example (part 1). Corporate information systems, 2019, №3(7), p. 29-52 (in Russian).

11. Stepanov D.Yu. Shaping universal requirements to ABAP-programs while preparing functional specification documents. Actual problems of modern science, 2014, Vol. 78, № 4, pp. 258-268 (in Russian).

PAPER DETAILS

Dmitry Yu. Stepanov, Mikhail N. Statsenko. Analysis of functional requirements relevant for developing corporate information systems // Естественные и технические науки. – 2021. – vol.160, №9. – p.98-103. – URL: https://stepanovd.com/science/article/113-2021-4-requirements.