INFORMATYKA EKONOMICZNA BUSINESS INFORMATICS 3(25) · 2012

ISSN 1507-3858

Jędrzej Wieczorkowski, Przemysław Polak
Warsaw School of Economics, Poland
e-mail: {jedrzej.wieczorkowski, ppolak}@sgh.waw.pl

AN APPROACH TO ANALYSIS AND IMPLEMENTATION. FROM THE WATERFALL MODEL TO THE TWO-SEGMENTAL MODEL OF INFORMATION SYSTEMS LIFECYCLE[1]

Abstract: From the point of view of the purchasers of software packages such as ERP systems, analysis and implementation phases involve most resources through the process of acquiring such software. The complexity of these phases is particularly notable in the case of corporate implementations in organizations with a distributed structure. Therefore, understanding the objectives of these phases and the activities involved in them is a key factor for the success of software package implementation projects. The authors created a two-segmental model aimed at the better representation of the life cycle of information systems. The classical waterfall model was used as a basis and reference point for the new model. The article demonstrates that the actual course of the two phases is better represented in the proposed two-segmental model.

Keywords: life cycle of information systems, two-segmental model of life cycle, analysis and implementation phases, software packages.

1. Introduction

Acquiring information systems is a difficult and complex process, burdened with a high risk of failure. Models of the lifecycle of information systems allow better understanding of that process and, consequently, lower the risks associated with the implementation projects. They are widely used in the content of textbooks and courses on software engineering and information systems development. In order to achieve that aim, those models must appropriately reflect the actual activities carried out during the process. However, the variety of information systems, software architecture and development methods have led to the creation of various, sometimes very different models.

[1] Selected parts of this article were published under non-exclusive copyright in *Proceedings of the Federated Conference on Computer Science and Information Systems FedCSIS 2012* (see [Wieczorkowski, Polak 2012])

2. The traditional models of information systems lifecycle

Early attempts to describe the full life cycle of information systems, in particular the production process, led to the development of the waterfall model of the software lifecycle. Originally the model was described by Royce in 1970 [Royce 1970]. In accordance with its initial form, each phase included feedback to previous phases. Later, the model was often limited to a strictly linear form, which was considered synonymous to the sequential approach with identified specific phases (see Figure 1) [Pressman 2001; Cadle, Yeates 2008].

The waterfall model and its practical usefulness receive sometimes radically different assessments. On the one hand, it is claimed that no real big project was carried out strictly in accordance with that model. On the other hand, when treated more flexibly, particularly allowing for reasonable iterations, most real-life information systems projects match up to that model [Jaszkiewicz 1997]. Generally, the waterfall model is useful to describe projects in which it is unlikely to return to work done in previous phases and the final products of those phases remain unchanged.

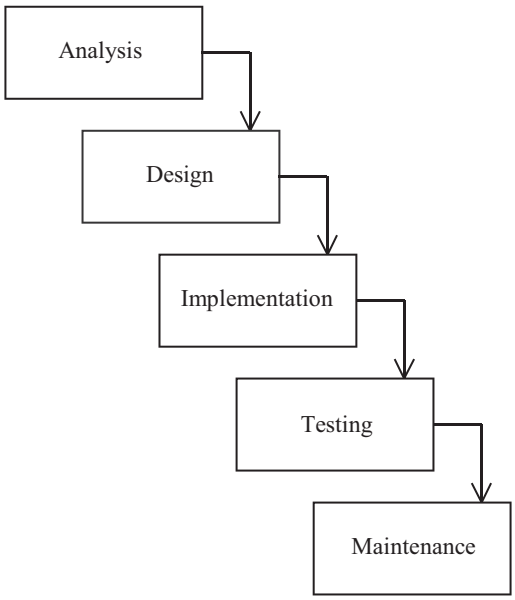

Figure 1. The basic form of the waterfall model

Source: own elaboration.

In order to eliminate the fundamental disadvantages of the waterfall model, such as the imposition of strict work sequence and the high cost of errors made in the initial stages or long breaks in direct relations between customers and producers, many other models were proposed, primarily using iterative connections (incremental

and spiral models) and prototyping. Also, more complex models were developed including [Pressman 2001; Cadle, Yeates 2008; Kobyliński 2004; Sommerville 2011; Chmielarz 2000; Maciaszek, Liong 2005]: "b" and "V" models, a parallel model, a database structure design model, a formal methods model, an extreme programming model, a formal transformation model, models for structured methods and object-oriented models. However, the waterfall model was the starting point for most of those concepts, and it became the basis for the development of structured analysis and design methods (for example SSAMD – Structured Systems Analysis and Design Method) [Pressman 2001; Cadle, Yeates 2008].

3. The life cycle of software packages

Life cycle models in software engineering literature mainly reflect the process of producing completely new software. However, the tendency to apply the extended usage of existing elements is observed in the practice of software development. That approach is a basis for the object-oriented methods of building systems, a reusable software and the component-based development model, as well as the usage of standard software packages.

Systems supporting the operational activities of companies and broadly defined management support (Enterprise Information Systems), including in particular ERP systems, are generally based on the concept of standard software packages. The same software products after adaptation to the needs of specific companies (in the process of customizing) are used in organizations from various industries characterized by their different activities. The typical software life cycle models proposed by software engineering are designed for systems dedicated to a particular customer (custom software). In terms of ERP systems, this approach is used very rarely. In the case of standard software packages, the classical approach towards the life cycle describes only the steps in the production of a system, aside from the issues of implementation. In the literature on software engineering, the acquisition of standard systems is only mentioned (for example the concept of packet-based solutions [Cadle, Yeates 2008]), but it cannot be recognized as a comprehensive model. Therefore, the literature describing the ERP systems contains several attempts to describe and model the life cycle of software packages:

- O'Leary [2005] distinguished processes which must be completed by a company implementing an ERP class system: deciding to go system, choosing system, designing, implementing, operation after going live, training.
- Similarly, Ross and Vitale [2000] differentiated sequenced steps: design, implementation, stabilization, continuous improvement, transformation.
- Flasiński [2006] distinguished three main sequential phases: pre-implementation analysis, system selection with signing a contact and proper implementation.
- Harwood [2003] proposed an evolutionary approach which includes five spirally linked phases: needs, vendor selection, implementation, go live and review, improvement.

Some topics concerning the work performed by a customer or an implementation partner during system acquisition were also discussed by Lenart [2010], Ray [2011] and Leon [2008].

There is a paradox – almost all of the above mentioned approaches are based on the classical linear approach when describing actions on the customer side. The description of the software packages life cycle in software engineering theory is significantly different from the practice of systems implementation, which reflects the implementation methods offered by the leading vendors of ERP systems.

In an attempt to build a comprehensive model of the life cycle, it is important to solve the dilemma of which approach to use. The model should include both the activities carried out during the software development and during the implementation. It would be theoretically possible to employ different approaches in the various phases of the cycle. However, according to the authors, the model, which should clearly distinguish the group of activities carried out in relation to the software development and its implementation at customer sites, would be too complicated. Therefore, in this article, the classical waterfall model is adopted as a reference point to discuss the characteristics of the implementation of software packages.

The model presented below is not an innovative proposal, it is only an attempt to present a model approach applied in practice. Its purpose is to provide systematic characteristics of individual phases, in particular the implementation analysis and the implementation.

4. The two-segmental model of information systems lifecycle

The two-segmental model of information systems lifecycle, developed by the authors [Wieczorkowski, Polak 2011], represents a typical lifecycle of a highly parameterized software package. The main feature of the model is its division into two segments (see Figure 2):

– Segment I, which covers activities occurring on the part of a system vendor,
– Segment II, which covers activities occurring on the part of a customer.

The first segment, in its principal run, is completed only once. Re-runs are possible in cases of the development of new versions. The second segment is run multiple times. It is repeated independently for each customer.

The article focuses only on two phases of the customer segment: the implementation analysis and the proper implementation. The combination of these phases is justified by the characteristics of the ERP systems implementation methodologies, which often consider these phases together as one phase.

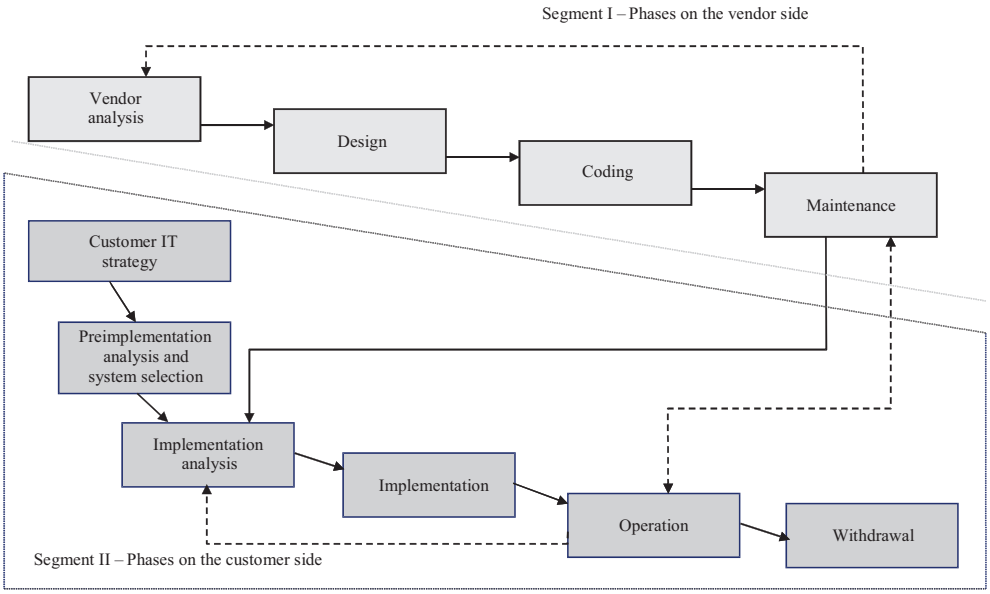

Figure 2. The two-segmental model of software packages lifecycle

Source: own elaboration.

5. Implementation analysis

The implementation analysis is performed just after selecting a system. Any earlier analytical work carried out in an organization in which the software is implemented, is aimed at choosing the system appropriate for the needs of the organization.

The implementation analysis phase and implementation phase can be performed only when the result of the system selection and the concept of its acquisition are known, as the ways to carry it out can vary, for example, depending on the implementation methodology. The analysis should lead to the development of a detailed specification defining the functions of the system, including the characteristics of its components, the principles of communication within and outside the system, as well as the conditions of use.

The analysis may be done in the traditional manner, i.e. through the specification of an IT strategic plan and the preimplementation analysis. In this case, the implementation analysis is produced by studying the existing state of information systems and management procedures in an organization. Some methodologies (e.g. ASAP) focused on the speed of action, limit the importance of the implementation analysis and instead propose predefined solutions addressing specific business sectors and regions. This approach can be described as an accelerated analysis methodology. Such solutions are supported by the reference models of business

processes, which are one of the customization methods. They are based on the recommendations of organizational settings, designed for the most efficient use of the system in the specific type of business. In this case, the company should implement the reference model using, for example, the ready-made maps of business processes. Some proprietary methodologies refer to such activities as implementation modeling, stressing that the system model is formed primarily by the location and parameterization of system solutions, and not through the identification and detailed analysis of customer processes. The use of business process reference models in many cases seems to be a reasonable compromise between ensuring the flexibility of solutions and maintaining the internal integrity of the software package.

The use of the accelerated method of analysis, particularly the use of the reference models of key business processes, can incur a risk of the partial loss of competitive advantage. Such an approach enables to use best practices in an industry, but in cases of implementation in market leaders, this argument is questionable at best. In such cases, customers should take into account the possibility of the strong customization of such activities, often in spite of recommendations from implementation partners, which, by simplifying the implementation, work in their own interest. On the other hand, excessive process customization can lead to the disproportionate individualization of the system, which is associated with an increase in the complexity of the project and, in the long run, the difficulty of software maintenance. These problems usually do not occur in the case of supportive or typical processes, which are implemented throughout the industry in a very similar way.

In real life, the details of the procedure used during the implementation analysis can vary greatly depending on the chosen solution, which may be based, for example, on the customization of the basic version of a system or on obtaining a predefined industry solution. Regardless of whether the two phases – the implementation analysis and the proper implementation – are executed separately or are merged in one phase, they are in practice a single project. The analysis is the basis for keeping the costs of the project at the intended level. Thanks to it, the difficulties in the functioning of organizations linked to the acquisition system are minimized. For the project's success, the fundamental aspects of the organizational setup of each project are particularly important. They include: the establishment of the organizational structure of the project, defining the objectives of the project (if possible, measurable and verifiable), defining the scope of the project, developing the project schedule with the definition and valuation of resources and a timetable for the deployment of funds, the application of the chosen methodology, along with the rules of the supervision and acceptance of the project and with risk, quality and change management.

In order to make the comparison of the implementation analysis phase from the two-segmental model of information systems lifecycle with the waterfall model of software lifecycle, it is necessary to refer to a phase usually called the requirements analysis, and partly to the design phase. There are many common elements in the corresponding phases, particularly in the case of the traditional approach to the

implementation analysis. But even in such cases, the implementation analysis does not distinguish between structured and object-oriented approaches. Moreover, the implementation analysis is dominated by the process-oriented approach, particularly in cases of the accelerated analysis.

6. System implementation

The implementation phase of software packages can be divided generally into two main concurrently performed sets of activities:

– the adaptation of a system to the specific needs of an organization (technology customization),
– the adaptation of business procedures and processes to opportunities arising from system capabilities.

Both these processes appear to be equally important. On the one hand, management solutions are primary to IT solutions [Skwarnik 2002]. On the other hand, the implementation of a computer system may become a stimulus to carry out the optimization of management solutions. The customization of a system can be carried out using various methods, depending on adjustment mechanisms available in a system. Every system contains a limited set of available programming and non-programming methods [Wieczorkowski, Polak 2010].

The extended usage of programming methods can justify the formal separation of a new subphase – the design on the customer side. That subphase is very similar to the design phase in the classical model. However, using programming methods is an option; therefore, in the two-segmental model, that subphase is not specified and any design work on the customer side belongs to the implementation phase.

The second group of activities – the adaptation of organizational procedures and processes – may be associated with business processes reference models, with the implementations of changes resulting from the thorough reorganization of business processes in accordance with BPR concept, or with continuous process reorganization following the implementation of BPM (Business Process Management) strategy. Regardless of the notion of changes, the process approach dominates the implementation phase, in the same way as in the implementation analysis phase. The process approach is also typical for contemporary implementations of ERP software packages.

In the case of the implementation phase formed on the basis of the waterfall model, the group of activities related to organizational changes is also included. However, the formal software production models do not devote enough attention to those changes.

In addition to the two main group of activities performed during the implementation phase, this phase includes, *inter alia*, such important activities as:

– the adaptation of system technical infrastructure, the management of operating environments and software installation;

– system integration with other applications;
– defining and implementing the principles of system administration and management;
– system testing;
– training for future users and the preparation of user instructions;
– the migration of data from existing systems.

The adaptation of system technical infrastructure includes, apart from the hardware setup, the configuration of software such as operating systems, database management systems, etc. The management of operating environments involves building technical infrastructure and software installation. The separation of environments is associated with ensuring the safety and efficiency of the system by separating the functions of individual servers, software or data areas. Systems operate in several different roles, such as ongoing operation, development and training. Hence, most software vendors call for the separation, in the form of individual installations or at least the areas of data, of operational system environments such as: production, training, development, testing and quality assurance, and also a spare security environment.

It is most important to separate the production environment as an independent installation, usually on a separate server. After implementation, it will serve as an environment for the proper operation of the system including the registration of data related to real business operations. Any further development activities or users' training should not influence that environment. Programming or configuration errors cannot result in the incorrect operation of the production environment. Other environments, depending on the size of the organization, may also be isolated as separate systems (independent software installations) or as separate data domains within a single installation. That second method, offered by some software packages, provides the separation of data between the environments. However, it does not solve performance problems resulting from overlapping different roles of a system carried out at one time. Furthermore, different data areas can use common configuration data, making it difficult to separate the various roles of a system. The structure of clients (*mandants*) used in the mySAP ERP system is an example of the implementation of separated data areas [*SAP*… 1998]. As part of the proposal of separating environments, all activities related to configuration and software development should take place in the configuration and development environment. After preliminary unit and modular tests carried out as part of the development, the changes are transferred to the test and quality assurance environment. There, further system tests take place, including checking whether the configuration of other modules' work is not infringed. Only after formal approval can the configuration be transferred to the production environment. Changing the management system, including transport mechanisms responsible for transferring data between the environments, is an important element in ensuring the safety of the system, both at the stage of implementation and operation.

Special features characterize the backup security environment. In this case, the choice of solution is influenced by the expected level of system availability in the event of failure. It is possible to opt out of a separate backup environment and to employ traditional backup copies and emergency procedures. On the other hand, it is also possible, for example, to apply real-time mirroring of all transactions in the production environment. The issues of separating software operating environments may also be complicated by the use of cluster technology.

Activities related to the integration (merging) with other systems in the organization can be very labor consuming. They usually involve programming and include, in particular, the construction of interfaces for data and messages exchanged between applications. They are particularly important in the case of solutions based on components and service-oriented architecture (SOA). The process of implementing the integration platform, in most cases, should be treated as a separate IT project, which does not fall into the software package life cycle.

The implementation of software, as in the case of software development, requires defining and implementing the principles of system administration and management. These principles have a huge impact on system security. Within this group of tasks, it is particularly important to develop and implement the concept of access rights.

The system tests performed during the implementation phase may also include the tests of new programs and the test of software package modules: function tests, integration tests, stress tests and acceptance tests.

Training related to the project usually begins before the implementation phase. Such training involves management staff (in order to demonstrate the general capabilities of the system) as well as the staff assigned to the project. It can continue during the implementation phase. However, the essential training at this stage is addressed to future end users and system administrators. In the case of the large integrated systems, it can be very labor intensive because of the number of participants. The training involves also preparing user manuals.

The migration of data often requires the development of tools dedicated to this purpose. As with any software project, small systems lifecycles can be identified. They include their own analysis, design and coding phases.

The above-mentioned additional groups of activities are very similar to the actions associated with the implementation phase in the traditional waterfall model of a software life cycle. Hence, both the two segmental model and the waterfall model similarly describe the technical activities of the implementation phase, whereas differences are observed in the approaches towards business-related activities.

The final outcome of the implementation phase is the conversion of information systems and the start of the production system. In the implementation of integrated systems, different approaches to the sequence of implementing modules can be used: sequential (step by step), overall (big bang), mixed – initially highest priority modules (middle size big bang). The overall approach is supposedly the most effective. It gives the possibility of the parallelization of some work, but may be

problematic and risky in the case of large projects. The practice of implementation proves the popularity of mixed methods in which the selection of the first group of modules results from the efforts to maximize the synergy benefits from the first step of the conversion. Usually the first step includes highly integrated modules, crucial for implementing the main business processes, for example: financial accounting, inventory management, procurement and sales. The following steps involve more modules, e.g. human resource management, controlling, etc. The sequential method can be ineffective due to the long duration of the project and, as a consequence, a long wait for its benefits.

The unique features characterize corporate implementations in multi-branch organizations. In the case of choosing the architecture of separate branch installations, the implementation of a distributed system is usually preceded by the pilot installations in one or more branches. The pilot implementation gives the opportunity to test the system (or at least parts of it) in real conditions. The functional requirements for the system are usually very similar in all branches. It gives, in the case of a decentralized system, the opportunity of using the initial replication of operational parameters from the pilot system in the installations in other locations. This method, called roll-out, employs the implementation based on a pre-configured business model, which describes the standard activities of the corporation. The reference solution is transferred to other branches. According to practical experience, the standardized business model used in the roll-out should include [Kunze 2007]:

– documentation and configuration of business processes;
– organizational structure;
– standard reports, forms, interfaces and other extensions;
– standard roles (users rights profiles);
– project documents' templates.

Advanced systems have a built-in mechanism to save the current configuration of the software along with information about nonstandard modifications made to the code. On this basis, it is possible to make copies of the configuration and the activation in a different system. The adoption of such an organization of the implementation significantly reduces the time and cost of a project. Regardless of the duplication of the solutions used in the pilot system, especially in the case of international organizations, there may be a need for the further customization of the system to adapt it to the needs of branches that operate in specific conditions of the country.

7. Conclusion

The two-segmental model of the information systems' life cycle is mainly characterized by the separation of vendor and customer segments, which reflects the main feature of software packages. Moreover, particular phases distinguished in the two-segmental model, and their sequence, differ from classical life cycle models. The

traditional sequence of phases (analysis, design, coding, implementation) was broken and the phases were divided between both segments. In the customer segment, two phases, the implementation analysis and implementation, were distinguished. They correspond to the implementation methods developed by the vendors of software packages and, as those methods are widely used, to the practice of the IT market.

Software packages implementation projects are characterized by the strong impact of the process approach on the phases of implementation analysis and implementation. But the process approach is also used in classical software engineering, e.g. data flow diagrams, which have long been used in the structural analysis, and are in fact process models. However, the modern process approach is not limited to the process transformations and the flows of data, but attempts to map business processes in organizations. Therefore, business processes play an important role in the implementation analysis, and the reference models are widely used in the customization of software packages.

References

Cadle J., Yeates D., *Project Management for Information Systems*, Pearson, Harlow 2008.

Chmielarz W., *Zagadnienia analizy i projektowania informatycznych systemów wspomagających zarządzanie*, Wydawnictwa Naukowe Wydziału Zarządzania Uniwersytetu Warszawskiego, Warszawa 2000.

Flasiński M., *Zarządzanie projektami informatycznymi*, ch. 3–6, Wydawnictwo Naukowe PWN, Warszawa 2006,

Harwood S., *ERP – The Implementation Cycle*, Butterworth-Heinemann, Oxford 2003.

Jaszkiewicz A., *Inżynieria oprogramowania*, Helion, Gliwice 1997, pp. 15–28.

Kobyliński A., Inżynieria oprogramowania. Modele cyklu życia oprogramowania, [in:] J. Płodzień (Ed.), *Wybrane zagadnienia informatyki gospodarczej*, Szkoła Główna Handlowa w Warszawie, Warszawa 2004, pp. 13–38.

Kunze M., Roll-outy SAP: szanse i zagrożenia. Specyfika wdrożeń w firmach międzynarodowych, *Lepszy Biznes* 2007, nr 1.

Lenart A., Wdrażanie i użytkowanie systemów ERP, [in:] S. Wrycza (Ed.), *Informatyka Ekonomiczna. Podręcznik akademicki*, Polskie Wydawnictwo Ekonomiczne, Warszawa 2010, pp. 343–372.

Leon A., *Enterprise Resource Planning*, Tata McGraw-Hill, New Delhi 2008, pp. 124–138.

Maciaszek L., Liong B.L., *Practical Software Engineering: A Case-Study Approach*, Pearson/Addison Wesley, Harlow 2005.

O'Leary D., *Enterprise Resource Planning Systems – Systems, Life Cycle, Electronic Commerce and Risk*, Cambridge University Press, Cambridge 2005.

Pressman R.S., *Software Engineering: A Practitioners Approach*, ch. 2, McGraw-Hill, New York 2001.

Ray R., *Enterprise Resource Planning. Text & Cases*, Tata McGraw-Hill, New Delhi 2011, pp. 27–55.

Ross J., Vitale M., The ERP revolution: Surviving vs. thriving, *Information Systems Frontiers* 2000, Vol. 2, Issue 2, pp. 233–241.

Royce W., Managing the development of large software systems, [in:] *Proceedings of IEEE WESCON*, Vol. 26, August 1970.

SAP Basis Technology, SAP 1998.

Skwarnik M., Elastyczne informacyjne systemy zarządzania. Mechanizmy realizacji, [in:] E. Niedzielska, H. Dudycz, M. Dyczkowski (Eds.), *Prace Naukowe Akademii Ekonomicznej we Wrocławiu nr 955*, Wrocław 2002, pp. 444–454.

Sommerville I., *Software Engineering*, ch. 2, Addison-Wesley, New York 2011.

Wieczorkowski J., Polak P., Customization of software packages – technology and business process perspectives, [in:] M.B. Nunes, P. Isaías, P. Powell (Eds.), *Proceedings of the IADIS International Conference Information Systems 2010*, International Association for Development of the Information Society Press, Porto 2010, pp. 549–552.

Wieczorkowski J., Polak P., The two-segmental model of software packages lifecycle, [in:] P. Powell, M.B. Nunes, P. Isaías (Eds.), *Proceedings of the IADIS International Conference Information Systems 2011*, International Association for Development of the Information Society Press, Avila 2011, pp. 291–295.

Wieczorkowski J., Polak P., Analysis and implementation phases in the two-segmental model of information systems lifecycle, [in:] M. Ganzha, L. Maciaszek, M. Paprzycki (Eds.), *Proceedings of the Federated Conference on Computer Science and Information Systems FedCSIS 2012*, Polskie Towarzystwo Informatyczne, IEEE Computer Society Press, Warsaw, Los Alamitos, CA 2012, pp. 1041–1046.

UJĘCIE ANALIZY I WDROŻENIA. OD MODELU KASKADOWEGO DO DWUSEGMENTOWEGO MODELU CYKLU ŻYCIA SYSTEMÓW INFORMATYCZNYCH

Streszczenie: Fazy analizy i wdrożenia w cyklu życia pakietów programistycznych klasy ERP angażują najwięcej zasobów i są najistotniejsze z punktu widzenia nabywcy tego rodzaju oprogramowania. Problem jest szczególnie istotny w przypadku wdrożeń korporacyjnych w organizacjach o rozproszonej strukturze. Ważne jest więc dla nabywców dobre zrozumienie celów tych faz i czynności realizowanych w ich trakcie. Model zaproponowany przez autorów ma na celu lepsze odwzorowanie cyklu życia systemów informatycznych. Jako odniesienie przyjęto klasyczny kaskadowy model cyklu życia systemów informatycznych. W artykule wykazano, że rzeczywisty przebieg tych dwóch faz jest lepiej odwzorowany w zaproponowanym modelu dwusegmentowym.

Słowa kluczowe: cykl życia systemu informacyjnego, dwusegmentowy model cyklu życia, fazy analizy i wdrożenia, pakiety oprogramowania.