Determinants of ERP Systems as a Large-Scale Reuse Approach

Marko Mijac¹, Ruben Picek^{1,*}, and Darko Andročec¹

¹ Faculty of Organization and Informatics, Department of Information System Development, University of Zagreb, Pavlinska 2, Croatia

Abstract. From the very beginnings of software era, enterprise environments have been one of the greatest generators of demand for complex software systems. Attempts to satisfy these ever growing needs for enterprise software systems have had a mixed success. Software reuse has been one of the leverage mechanisms software producers had at their disposal, and through the year's different reuse approaches emerged. One of the most successful large scale reuse approaches in enterprise environment are Enterprise Resource Planning (ERP) systems, which intend to reuse domain analysis and best practices in doing business, as well as software design and implementation. However, in literature, ERP systems are seldom viewed and described as one of the reuse approaches. Therefore, in this paper we aim at systematically analyzing ERP systems from the software reuse perspective in order to gain better insight into their characteristics, constituent elements, and relationships.

1 Introduction

On the famous NATO conference on software engineering in 1968, it became clear that building software had to be approached in an engineering way, in order to tackle the problem of continuously increasing demand for high quality complex software systems. As it might have been expected, one of the greatest generators of demand for such software systems were enterprise environments. At first, these were large corporations and institutions, such as financial institutions, telecommunication companies, industrial corporations etc. Later, when involved technologies became more available and more affordable, software producers had to cater the needs of broader clientele, which differed in both scale and target domain. Now, in addition to large corporations, small and medium-sized companies and even individuals became generators of demand for enterprise and business software. Building such software systems tailored to fit specific business processes of each separate organization, was simply not feasible within available resources of both producers and consumers of software. Huge risks and development expenses had to be tamed by dispersing them to as many organizations as possible, and increasing development productivity in general.

As in many other areas of human activity, software producers turned to one of the most powerful mechanisms to increase productivity and quality – reuse. In the context of software, Frakes and Kang [1] define reuse as the use of existing software or software knowledge to construct new software. In the very beginnings of software era, software reuse was predominantly opportunistic, which meant it was not planned in advance, but rather applied on an ad-hoc basis. Later, reuse became more and more systematic,

which meant that software artefacts were intentionally created to be reused and then reused multiple times. Over the time, it became clear that a lot of the artefacts that arise in different stages of software development process can be reused, so different reuse approaches emerged. Because of its fundamental importance, reuse permeated most of the software development, including programming paradigms, development processes, and even business models.

As much as the needs of individual business clients differed between each other, their shared goal is to support business processes through the use of information technology and software systems. Fortunately, these business processes, especially if constrained within particular business domain, are common or similar enough across many organizations. One of the most successful large-scale reuse approaches in enterprise environment that utilize this fact, are Enterprise Resource Planning (ERP) systems. ERP systems are modular software systems with centralized database which integrate and support distinct but related business processes in organization. In terms of software reuse, Sommerville [2] similarly describes ERP systems as large-scale systems which encapsulate generic business functionality, with business rules being configured for particular organization. By being some of the largest and most complex software systems, they are hard, expensive and time consuming to build. However, like other reuse approaches and mechanisms, ERP systems justify and amortize involved effort and resources by being reused in many organizations within business domain.

Although widely used and accepted, ERP systems are seldom viewed and described as one of the reuse approaches. We believe, however, that systematic depiction of ERP systems as reuse approach can be

* Corresponding author: ruben.picek@foi.hr

beneficial to their better understanding and positioning in fairly broad reuse landscape. Therefore, in this paper we will analyze ERP systems as an approach to software reuse in order to gain better insight into their characteristics, constituent elements, and relationships. The remainder of the paper is structured as follows. Next section brings brief review of relevant related work on ERP systems in the context of reuse. The third section explains methodology of the analysis. Fourth section presents extensive results of the conducted analysis, followed by discussion and conclusion sections.

2 Related work

In the current literature, ERP system are rarely studied in the context of reuse approaches. However, there are some works focused on reuse approaches for ERP requirements understanding and engineering. For example, Daneva [3] addressed the reuse aspects in the ERP requirement engineering process by incorporating reuse metrics planning as part of the implementation of metrics on SAP R/3 implementation projects. Daneva developed a practical requirements reuse measurement plan consisting of the following components: relevant stakeholders, a requirements engineering process model, a process integration model, counting rules, tools and reuse data usage tables. In her later paper, Daneva [4] showed the application on a function points based reuse measurement model in three ERP projects for telecommunication sector. She has found that reuse is possible up to 80% at best.

Wang et al. [5] proposed the architecture of component-based ERP systems in which the high efficiency for reconfiguration is supported by multiple grained components. Their paper also describes the process and artifices of ERP agile reconfiguration based on component reuse. Baig and Fadel [6] concluded that the use of reusable artefacts during requirement engineering cycle of ERP implementation can increase the chances of success. Those artefacts include business process diagrams, use cases and mapping sheets for configuration and migration. Salinesi et al. [7] presented a reuse-based requirements elicitation method for ERP integration. The method was developed in an action research approach in a project held in a charity association that wanted to integrate Microsoft's Dynamics NAV.

Demirors and Kucukates Omural [8] tried to establish an approach to calculate reuse reflective size of ERP projects. They applied their approach in an SAP implementation project to evaluate it in a real-life setting. Orosz and Orosz [9] introduced a new method of identifying the parts of software that can be later reused on the example of cloud based ERP solutions. They have chosen a model driven approach, and used Microsoft Dynamics 365 in cloud version to model and test their approach.

3 Methodology

In order to analyse reuse aspects of ERP systems in a systematic manner, we used a taxonomy proposed by Krueger [10] in his seminal paper on software reuse. Proposed taxonomy describes reuse approach from the perspective of its reusable artefacts and how they are abstracted, selected, specialized and integrated. These four taxonomy dimensions are described in the table below and further elaborated in the remainder of this section.

Table 1. Taxonomy dimensions

Abstraction	Describes the type of software artefacts being reused and the exact abstractions used to describe them.
Selection	Describes how reusable artefacts are being selected for reuse.
Specialization	Describes how generalized artefacts are specialized for reuse.
Integration	Describes how reusable artefacts are integrated into a complete software system.

Abstraction is a fundamental technique for handling complexity in software engineering, essential in modelling to reduce and simplify original concept being modelled. Abstraction always suppresses some of the details of the original concept. Through the history of software development, we can witness continuous process of raising the abstraction level, from the raw hardware representation of a program, to machine code, assembly, higher-level programming languages, domain-specific languages and others.

Each abstraction has two levels: specification and realization, with specification being associated with multiple realizations. Abstraction specification is a higher, more general level, which exposes parts of the abstraction that the abstraction creator wanted to be visible to abstraction users. These parts can be either variable, i.e. expressing variant characteristics, or fixed, i.e. expressing invariant characteristics of abstraction realizations.

Fig. 1. Two levels of abstraction.

In addition to variable and fixed part, abstraction realizations also contain hidden part, i.e. the part that abstraction creator wanted to suppress from specification and eliminate from concern of the users. To put it more plainly, we can say that abstraction specification describes “what” the abstraction does or “what” it

represents, and abstraction realization describes “how” the abstraction does it.

In order to express effectiveness of abstractions, Krueger [10] introduces the notion of cognitive distance, as a measure of amount of intellectual effort required by software developers to take software system from one stage of development to another. Effectiveness of abstractions is inversely proportional to cognitive distance, so the goal of reuse approaches is to minimize cognitive distance. Krueger [10] discusses two ways this can be done: (1) reduce intellectual effort to go from initial conceptualization of system to system specification, (2) reduce intellectual effort required to produce executable system from the system specification.

Selection dimension determines how particular reuse approach helps developers find appropriate reusable software artefact and compare it with potential alternative artefacts. This is especially important when there is a large number of software artefacts which are reuse candidates, and it is not obvious which ones are fit for our needs. If in such cases there is no support for artefact selection, finding and assessing appropriate artefact may prove to be more challenging than building the artefact.

In order to be reusable, artefacts are generalized and made incomplete through variable part of abstraction.

Specialization dimension determines how are these reusable software artefacts customized and configured in order to satisfy concrete needs. This primarily refers to implementing variable parts of abstraction.

Integration dimension deals with the way reusable artefacts are integrated into complete software system. While in the cases of simple reusable artefacts integration can be done implicitly and ad hoc, in the cases of more complex reusable artefacts integration may require explicitly defined integration frameworks and procedures.

4 Results

In this section we will presents results of defined four taxonomy dimensions: abstraction, selection, specialization and integration.

4.1 Abstraction

When we consider purchasing ERP system, what we have in mind is reuse of already existing software implementation (ready to be used modules and components) and design (overall architecture, applied styles and patterns) to support business processes in particular domain. Design and implementation are primary reusable artefacts that ERP as a reusable approach offers as an alternative to building software systems from scratch. These artefacts are best described by abstractions coming from the very business domain they target, such as domain concepts, business processes and business rules. That being said, abstraction

specification in the context of ERP systems would refer to informal or formal descriptions of what domain concepts, business processes and rules can be supported by particular ERP system. On the other hand, abstraction realization would refer to how this is done, i.e. it refers to executable software system adjusted to support business processes of particular company.

At fundamental level, in order for ERP systems to be reusable, there has to be something to be reused - something common to all abstraction realizations – a fixed part of the abstraction. In ERP systems this includes overall system architecture, components and features that the vendor envisioned to be used “as is”, without modification. In a mature, and legally well-regulated business domain, such commonalities in how business is done should allow creation of fairly large fixed part of the ERP system. However, even in the same domain different companies operate differently. What is directly offered by ERP system, in a larger or lesser extent, always differs from what individual company needs and expects. So in order for ERP systems to be reusable, there also has to be a way to cover the variabilities in business processes – a variable part of the abstraction. This is done by allowing users to configure and customize ERP systems.

In his report, Kimberling [11] describes configuration as a normal set-up of the software system, something that is a usual part of any ERP implementation as a prebuilt option, and does not involve changes to source code. According to same author, unlike configuration, customization necessarily involves changes to source code. Configuration is a preferable method of achieving variability for both ERP vendors and users. At one hand it allows vendors to be in charge of what is and can be modified in ERP system, and on the other, provides users with simpler way to adjust ERP system to their needs. However, this requires ERP vendors to anticipate every possible variation that could arise in customers’ needs, which is of course not possible. By allowing customers to customize ERP system, i.e. to make code modifications, vendors increase the flexibility of the ERP system, but also introduce certain risks. ERP vendors usually allow customization of only certain features and at a cer tain customization points. What exactly and how can be configured and customized is elaborated in ERP’s technical documentation. However, most implementation details are not visible to external users and make the hidden part of the abstraction.

In many cases, ERP systems do not allow sufficient level of variability, or the configuration and customization may be too expensive for the customer. In such cases, compromises are made, and customer may settle to customize only part of the ERP system, and to accept other parts of ERP system as they are, unmodified. In this way, customer does not only reuse implementation and design of the software system, but is also forced to reuse business practices built into ERP system.

ERP systems as a complete or semi-complete executable systems can be characterized as a reuse approach with a small cognitive distance. Arguably, most of the intellectual effort is spent to drive the system from initial conceptualization, to system specification, where the customer's requirements and needs are expressed in terms of domain abstractions and aligned with ERP features. Producing an executable system from system specification is a matter of selecting required modules and components, and making necessary configurations and customizations. Design, implementation and testing activities are mostly avoided by the customer, except in cases of customization. We can identify following ways ERP systems reduce cognitive distance:

- Facilitates domain analysis by providing already identified common domain concepts, and by offering standards or best practices and workflows.
- Facilitates system design by already incorporating system architecture.
- Facilitates system implementation by providing already working system with a number of components and modules to use and customize.
- Facilitates system testing by providing components and modules thoroughly tested by ERP creators as well as ERP users.
- Facilitates system deployment and putting system to use by providing detailed procedures.

This, however, does not mean that the process of acquiring and setting up ERP system in a company is an easy endeavor. On the contrary, due to the sheer size and complexity of the domain and ERP software system, this falls into one of the most challenging and resource demanding investments in company's lifetime.

4.2 Selection

While there is an increasing number of ERP system offered for businesses worldwide, probably there are only a few ERP systems that could be suitable for particular company and their business processes. Although the choice of selecting ERP system is narrower, it is still very difficult and strategically important to select the right one, because it will support the overall business and failure to do so successfully can be extremely costly. These systems are advertised by their vendors and representative implementators, online or through different events. An overall, standard catalog of ERP solutions is not available. Buyers of ERP system (Company) need to identify ERP solutions which are available in their environment, and then assess and compare their characteristics in order to decide on solution which would be best suited for them. This could be a fairly complex and sophisticated process, which takes into account ERP system availability, compatibility with company's processes, prices and business model, features, localization, technology, support, quality, tradition etc.

This step is essential, since selecting wrong or unsuitable ERP system can result in total failure and bring into the question the very existence and survival of the company.

This segment can be viewed from two perspectives: selection on a level of organizational informatization (whole information system) and selection on a level of ERP modules.

4.2.1 Selection on a level of organization informatization (whole information system)

Everyone ask the same set of questions: How to find the best one? Is there unified best practice of selecting ERP systems? The first reusable element in this segment is to determine the Process of selection. Different proposals of selection process can be found in the literature. Stefanou [12] proposed a framework for the selection process of ERP systems, which consists of three phases: business vision, detailed examination and definition of business needs, and of the various constraints and selection of modules of the core system that support critical business processes. Another ERP selection process model is presented in [13] with similar segments: preparation, analysis, evaluation and selection. Next approach describes a formal description of relevant characteristics of ERP system followed by systematic methodology called SHERPA [14]. Further on, differences in characteristics of the ERP system selection process between small or medium and large organizations are also discussed in [15] and [16]. In close connection with the process of selections, Methods of selection are our second reusable element. Based on the approach and criteria, it is clear that organizations choose ERP system based on: detailed requirements, key requirements or proof of concept (demo), so they need the appropriate selection methods. These methods can be divided into several categories:

- Methods based on information gathering techniques (open and closed questions) - the most prominent is a funnel method [13].
- Methods which emphasizes the importance of testing an ERP system before making a final decision for its selection. Here, the purpose of the test is not to determine the functionality itself but also to determine the adaptability of the solution to the actual requirements. Example of this category is Clarkston-Potomac method [13]
- Methods for multi-criteria decision-making where the most popular are ANP (Analytic Network Process) and AHP (Analytic Hierarchy Process) - [17] and [18].
- Software tools for validation of ERP packages

Final reusable element for Selection are artefacts where the most valuable is document with detailed description of AS IS and TO BE business processes. Usually this document is called: functional requirement document or business blueprint.

4.2.2 Selection on level of ERP modules

Today ERP systems are aligned in modules which are organized in a core of ERP systems and in addition/advance segments. Once the organization select the ERP solution, remains question which business segments to cover. Although the idea of ERP systems is covering all aspects of business, because of existing legacy application, there is often a need to integrate these legacy apps with ERP core functionality in order to reduce the costs. Another issues are customer specific segments of particular business so many add-ons have to be developed for each customer. Because they are reusable (some other customer of particular ERP systems can also have need of that segment), ERP vendors establish mechanism for validating these specific add-ons and allow their resellers to sell it to others. Since in today's ERP systems market, digital transformation of business emergence and classic business models are changing, benefits of cloud technology and the Cloud ERP system are recognized, so new ideas of reuse are arising. The leading ERP system provider start to offer some applications that suits your business needs and can be easily integrated to Cloud ERP core functionality. So in future, we can expect thousands of apps on specialized ERP vendor stores that can be reused.

4.3 Specialization

Specialization corresponds to filling-in the variable parts of the abstraction, i.e. deciding on concrete abstraction realization that fits our needs. In the context of ERP system this means configuring and customizing the features of ERP system in accordance to requirement specification of the target company. These are the necessary means for companies using ERP system to differentiate themselves from their competitors. Usually, some of the configurations are mandatory during the ERP system set up, and may include: enabling or disabling certain modules, components and features, changing administrative settings and preferences, making minor adjustments to database model, reports and graphical user interface etc. Such configurations are most often well supported by the vendor, in a form of prebuilt options, existing modules and components to choose from, and guides and tools to aid making configurations. Customizations are usually not mandatory, and the base, standard ERP instances can be set up without it, leaving configuration approach to handle any variations. However, in many cases, differences between company's requirements and ERP's features bring the need for source code modification, either by replacing existing feature, modifying it, or even adding previously non-existent features. Kimberling [11], for example, reports that only 23 percent of companies manage to set up plain vanilla ERP software. Others are forced to make customizations in order to align ERP system to their business processes. Both configuring and customizing ERP systems is a complex task, which requires great expertise in both target

business domain and ERP technology. Individual companies interested into acquiring ERP solutions rarely have such experts. So, this gave rise to a large number of individual consultants and consulting companies which became specialized and certified for setting up particular ERP solutions.

4.4 Integration

The vision of software system that would integrate different business processes across organization and allow them to efficiently store and share information is the one that inspired the creation of ERP systems. Integration of components and modules within ERP system is determined by the overall ERP software architecture, prescribed programming interfaces, contracts and other structural, functional, and performance-wise constraints that are either purposely set by ERP vendors, or are simply inherent to the ERP system itself. Integration of predefined components and modules (either ERP vendor made or third-party) that are compatible with ERP's architecture is a matter of choosing which ones we want to include into ERP system, and possibly making some configurations. For example, one might choose to use modules such as Accounting, Finance, and Sales, and the underlying ERP architecture has to ensure that these modules coexist and adequately communicate. When making new modules and components, or making any modifications, it is necessary to keep them aligned with existing architecture, specified interfaces and constraints. Integration of ERP systems with other existing, possibly legacy systems in a host company or outside of it (e.g. business partners), is another complex issue. Depending on the mismatch in expected inputs and outputs between ERP system and an external system, and their respective openness to change, integration might require different solutions. This may involve customizing ERP system and/or legacy systems to bring them to the same terms. If it is not possible or desirable to change existing systems, a frequent alternative is to create a middleware system that will mediate their differences and make them compatible. Some of the widely accepted technologies for aiding integration effort include SOA (service-oriented architecture) and EAI (enterprise application integration) [19]. Integration between Cloud ERP and an external system is a question of interoperability. According to Boza et al. [20], interoperability in this context can be seen as interaction between ERP system and other, internal or external systems. They proposed two perspectives of interoperability, the first one considering technological aspects such as web services, SOA, Cloud computing, IoT etc., and the second one considering business aspects such as BPM, BPR; virtual enterprises, references models etc. For example, Andročec et al. [21] have proposed the ontologically based model for integration of the IoT and Cloud ERP systems by using Semantic Web services.

5 Discussion

Although they are rarely discussed as a software reuse approach, using proposed taxonomy we demonstrated that ERP system have an important place in overall landscape of reuse approaches. The primary artefacts intended to be reused through ERP systems are design and implementation artefacts of software which supports business processes. By being very large-scale reuse approach, ERP systems contain other, smaller-scale reuse approaches. Design is, for example, imposed by ERP's software architecture, which is defined through the use of several other design reuse approaches, such as architectural styles and design patterns. However, ERP systems are more concrete than such reuse approaches which offer solely design reuse. Indeed, by being executable system, they offer implementation reuse in a form of software modules and components. These, again, were not made from scratch, but rather built reusing software frameworks, libraries, and other components.

As with other reuse approaches, reusable artefacts are in most cases presented to end-users through well-define abstractions, namely domain concepts, business processes and rules. In terms of this, ERP systems can be characterized as a reuse approach with effective abstractions, i.e. with a small cognitive distance. High-level of reuse and good abstractions enable acquisition of ERP system to be easier, less resource consuming, and more available to broader range of companies. Unfortunately, this does not take away all the complexities and risks involved into acquiring ERP system. In addition, small cognitive distance also implies giving up most of the control over the system design and implementation, since lowering intellectual effort means bypassing a number of traditional software development activities. Thus, acquiring ERP system still remains an important and strategic decision.

Analyzing ERP systems through the dimensions of selection, specialization and integration, witnesses the size and complexity of the ERP systems' domain, with each pointing at a whole different research area. Selection, for example, deals with major questions of choosing appropriate ERP system for particular company, and choosing right components within the ERP system. Failing to do so can jeopardize the sheer existence of the company, therefore selection process has to be systematic and thorough, preferably guided by the proven methods. Specialization on the other hand, deals with introducing chosen ERP system into a company. Even if we have chosen adequate ERP system, a number of important issues may arise at this stage, which can have adverse effects on ERP acquisition, or even cause it to fail. In order to prevent this from happening, and to guide the process of introducing ERP system into a company, major vendors prescribe specialized methods, which are based on theory and past experiences. Number of issues arise at this stage also due to the need to configure and customize ERP systems to better suit

specific needs of individual companies. Sometimes, these customizations are not even possible, or are too hard and expensive to be made. Lastly, integration deals with how reusable artefacts are integrated into a whole, complete system. In terms of internal ERP components and extensions this is determined by the overall architecture and prescribed interfaces of the ERP system. However, since ERP systems are usually introduced into a company with other, already existing software systems, external integration also has to be considered.

5 Conclusions

In this paper we analyzed ERP systems in terms of software reuse using the four-dimension taxonomy. We showed that ERP systems have all what it takes to be considered large-scale reuse approach with effective abstractions. Selection, specialization and integration dimensions provided suitable framework to discuss characteristics and issues of ERP systems. We believe such analysis to be beneficial in terms of better understanding ERP systems in general and as a reuse approach. This enables further comparison of ERP systems with other comparable large-scale and also constituent smaller-scale reuse approaches, and positioning ERP systems in the very broad and tangled reuse landscape. We also believe that this analysis can be used to assort conducted and future research in the domain of ERP systems. In the future work, we intend to focus on the ERP specialization issues, namely configuration and customization.

References

1. W.B. Frakes, Kyo Kang, Software reuse research: status and future, *IEEE Trans. Softw. Eng.*, vol. **31**, No. 7, pp. 529–536, (2005)
2. I. Sommerville, Software engineering, *Tenth edition*. Boston: Pearson, (2016)
3. M. Daneva, Practical Reuse Measurement in ERP Requirements Engineering, *Advanced Information Systems Engineering*, pp. 309–324, (2000)
4. M. Daneva, Understanding Functional Reuse of ERP Requirements in the Telecommunication Sector: An Empirical Study, *Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, Rotterdam, Netherlands, pp. 216–221, (2014)
5. Z.-J. Wang, X.-F. Xu, D.-C. Zhan, Component reuse based agile reconfiguration for Enterprise Resource Planning (ERP) systems in manufacturing enterprises, *Int. J. Prod. Res.*, vol. **44**, No. 23, pp. 5107–5129, (2006)
6. J. J. A. Baig, M. A. A. Fadel, Measuring reusability during requirement engineering of an ERP implementation, *Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, Cairo, Egypt, pp. 258–262, (2017)

7. C. Salinesi, M. R. Bouzid, E. Elfassy, An Experience of Reuse Based Requirements Engineering in ERP Implementation Projects, *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, Annapolis, MD, USA, pp. 379–379 (2007)
8. O. Demirors, N. Kucukates Omural, Exploring Reuse Levels in ERP Projects in Search of an Effort Estimation Approach, *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Prague, Czech Republic, pp. 191–197 (2018)
9. I. Orosz, T. Orosz, Code reusability in cloud based ERP solutions, *21st International Conference on Intelligent Engineering Systems*, Larnaca, Cyprus, (2017)
10. C. W. Krueger, Software Reuse, *ACM Comput Surv*, vol. **24**, no. 2, pp. 131–183, (1992)
11. E. Kimberling, ERP Software Customization: The Ultimate Sin of Enterprise Software?, *Panorama Consulting Solutions*, [Online]. Available: <http://panorama-consulting.com/erp-software-customization-the-ultimate-sin-of-enterprise-software/>, (2009).
12. C. J. . Stefanou, The Selection Process of Enterprise Resource Planning (ERP) Systems, *AMCIS 2000 Proceedings*, Long Beach, pp. 988–991., (2000)
13. C. D. Avram, R. D. Zota, L. Ciovisa, Developing an ERP Strategy Based On the IT Solution Life Cycle, *Econ. Inform.*, vol. **12**, no. 1, pp. 26–41, (2012)
14. X. B. Illa, X. Franch, J. A. Pastor, Formalizing ERP Selection Criteria, *Proceedings of the 10th International Workshop on Software Specification and Design*, Washington, DC, USA, pp. 115, (2000)
15. E. Bernroider, S. Koch, Differences in Characteristics of the ERP System Selection Process Between Small or Medium and Large Organizations, *AMCIS 2000 Proceedings*, pp. 1022–1028, (2000)
16. E. Bernroider, S. Koch, ERP selection process in midsize and large organizations, *Bus. Process Manag. J.*, vol. **7**, no. 3, pp. 251–257, (2001)
17. S. Tasnawijitwong, T. Samanchuen, Open source ERP selection for small and medium enterprises by using analytic hierarchy process, *5th International Conference on Business and Industrial Research (ICBIR)*, Bangkok, pp. 382–386 (2018)
18. J. P. Silva, J. J. Gonçalves, J. A. Fernandes, M. M. Cunha, Criteria for ERP selection using an AHP approach, *8th Iberian Conference on Information Systems and Technologies (CISTI)*, Lisboa, Portugal, (2013)
19. T. Kähkönen, K. Smolander, ERP Integration – A Systematic Mapping Study, *14th International Conference on Enterprise Information Systems-ICEIS*, France, (2013)
20. A. Boza, L. Cuenca, R. Poler, Z. Michaelides, The interoperability force in the ERP field, *Enterp. Inf. Syst.*, vol. **9**, no. 3, pp. 257–278, (2015)
21. D. Andročec, R. Picek, M. Mijač, The ontologically based model for the integration of the IoT and Cloud ERP service, *Proceedings of the 8th International Conference on Cloud Computing and Services Science*, Funchal, Madeira, Portugal, pp. 481–488, (2018).