Aligning Software Design with Development Team Expertise

Jānis Grabis[1], Egils Meiers[2], Inese Šūpulniece[1], Solvita Bērziša[1], Edgars Ozoliņš[2] and Ansis Svaža[2]
[1]*Institute of Information Technology, Riga Technical University, Kalku 1, Riga, LV-1658, Latvia*
[2]*Visma Enterprise, Kronvalda bulv.3/5, Riga, LV-1010, Latvia*

Keywords: Enterprise Application, Refactoring, Development Reorganization, Team Expertise, Clustering.

Abstract: Large enterprise applications are developed by teams of developers specializing in particular functional or technical areas. An overall application architecture is used to guide allocation of development tasks to the development teams. However, quality of the architecture degrades over the application life-cycle and manual refactoring is challenging due to the size and complexity of enterprise applications. This paper proposes to use automated clustering of large enterprise applications, where clusters are built around application business centers, as a means for refactoring the software design with an objective to improve allocation of software modules to development teams. The paper outlines a module allocation process in the framework of the overall enterprise application development process and reports an illustration of the allocation process. The illustration is based on the case of refactoring of a large third tier ERP system.

1 INTRODUCTION

Development of large software applications such as Enterprise Resource Planning (ERP) systems is a complex task. These systems are constantly evolving and huge efforts are devoted towards maintenance of existing applications and developing new functionality. Expertise of development team is a crucial factor to ensure efficient maintenance and software evolution (Bennett and Rajlich, 2000). That is especially important for large multi-functional applications because for their wide scope and long life-cycles. Developers specialize in particular functional and technical areas to ensure development efficiency (Liang, 2010). This specialization is enabled by having a modular system design (Paulish, 2002). Unfortunately, the system design if initially present tends to deteriorate during the life-cycle for large complex applications (Cai et al., 2009).

This paper investigates a problem of refactoring the system design of long life-cycle packaged applications with an objective to support modularized development by dedicated teams. The refactoring is achieved by automated clustering of the system into self-contained modules. The automated clustering is considered because manual refactoring is prohibitive in the case of large systems. It is assumed that development of the modules requires specific development expertise and teams are formed and the modules are assigned to them to attain the best match between the required competencies and the team's expertise.

The objective of this paper is to propose a method for aligning software design and team's expertise. The method is geared towards development of packaged applications including ERP systems. ERP development is investigated from the vendor perspective (as opposed to the ERP implementation perspective). Application of the method is illustrated using an example of the third tier ERP system undergoing a system's redesign project. The further research is intended to focus on evaluation of actual benefits of redesigning of the ERP systems from the vendor's perspective what is an insufficiently exposed research and practical problem. The main expected contribution of the proposed research is to determine suitability of automated refactoring to guide development team assignment and to facilitate inter-team collaboration in the case of large-scale packaged applications.

The rest of the paper is organized as follows. Section 2 describes the ERP development process highlighting its modular nature and discusses the role of development team's composition. Section 3 introduces a process for allocating modules to development teams. Section 4 describes preliminary evaluation of the alignment process. Section 5 concludes.

Grabis, J., Meiers, E., Šūpulniece, I., Bērziša, S., Ozoliņš, E. and Svaža, A.
Aligning Software Design with Development Team Expertise.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 1*, pages 560-565
ISBN: 978-989-758-187-8

2 ERP DEVELOPMENT PROCESS

An ERP development process resembles the traditional software development process. Two distinguishing features of this process are specific aspects of requirements management and wide scope of the application resulting in functional and technological complexities. Monnerat et al. (2008) suggest to use enterprise modeling techniques to establish a comprehensive set of requirements covering all areas of application of ERP systems. The incremental approach (Sommerville, 2010) to evolving functionality of the ERP systems on the basis of key requirements and overall architecture is used to address the functional and technological complexities. Figure 1 shows an overall ERP development process.

An ERP system can be developed from scratch or by evolving existing software. The latter case is more common in practice since either the previous version of the ERP system is available or the ERP system development is a continuation of successful custom software development. In this research, we focus on maintenance and evolution of existing ERP systems. The development process is driven by feedback from customers, market trends, changes in regulatory requirements and other factors (Xu and Brinkkemper, 2007). The enterprise modeling activity concerns scoping of ERP development and identification of key requirements towards the ERP system. ERP systems consist of functional modules, which cover certain areas of enterprise activities. Modules can be developed relatively independently (modules from the development perspective are not necessarily the same as modules from the functional perspective). However, to ensure development and usage efficiency and consistency, the functional modules are developed following common principles determined according to the base requirements and operationalized in the overall architecture or systems design. The individual modules are integrated together in order to release a new version of the ERP systems to customers. The module development, integration and release are continuous processes, especially, if agile techniques are used in development (De Carvalho et al., 2010).

Enterprise modeling requires participation of process owners and key users (Sandkuhl et al., 2014). They specialize in different business areas of the enterprise and possess limited knowledge and understanding about specific aspects of other business areas. Moreover, the research suggests that cross-functional teams have negative impact on implementation of ERP systems (Lui and Chan 2008). Similarly, agile development practices suggest using vertical teams rather than horizontal teams (Ratner and Harvey, 2011). Carmel and Bird (1997) provide evidence that packaged systems are usually developed by teams of up to five developers. Therefore, it is often practical and advisable to distribute ERP development activities among teams specializing in particular business areas.

Software architecture plays a major role in dividing software into manageable modules assigned to individuals or small teams for development (Unphon and Dittrich, 2010). However, that might be hampered by intricacies of the ERP technical design (Rettig, 2007), i.e. ERP systems consist of a large number of components linked together in a complex web of associations, which has evolved during the life-cycle. The overall architecture can be improved by refactoring although manual refactoring of large enterprise applications is challenging. This paper explores automated decomposition of ERP systems as a part of software design refactoring to improve allocation of modules to development teams.

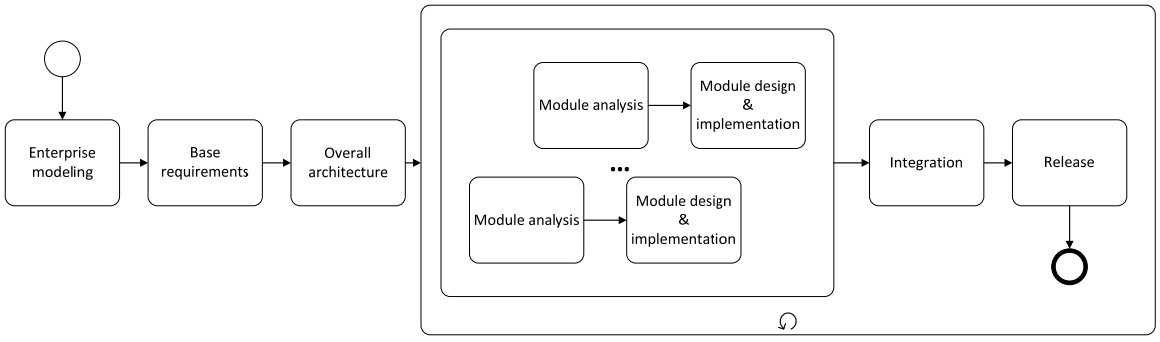

Figure 1: ERP systems development process.

3 ALIGNMENT APPROACH

The alignment approach is elaborated for an ERP system which requires major architecture refactoring. The refactored architecture will be used to guide future software evolution processes including project management and team assignment processes.

System redesign and identification of modules takes place at certain milestones of software evolution. Development teams change more frequently. However, it is assumed that once a module has been assigned to team knowledge is preserved in it even though team members change occasionally. Alignment between team expertise and design also needs to be periodically updated since newly developed components are assigned to modules and characteristics of modules might change.

The system is divided in modules using clusters built around business centers (Figure 2). The business centers are system's design components identified by a system architect as being central to providing desired functionality. The clustering is performed automatically and clusters consist of closely related components as measured by strength of associations among the components. There are components having only internal associations within a cluster and there are associations spanning boundaries of the clusters. The latter associations are particularly important to determine interfaces and to set contracts among development teams. The clustering addresses just some of the system's redesign concerns. It is used as an input to other refactoring activities (e.g., Riva 2004), which yield the final division of the systems into modules. Competency requirements are identified for every module. They concern knowledge of specific functional or technical areas associated with a particular module. For instance, ab absence management module requires knowledge of human resources management.

Development teams work continuously throughout the system's life-cycle and has certain functional and technical competencies. The available competencies concern knowledge possessed by team members. Experience in a specific functional or technical area plays a major importance in determining team competencies. The modules are assigned to the development teams by matching the available competencies and the competency requirements. Some changes in teams' composition can be introduced to achieve a better match.

4 PRELIMINARY RESULTS

Feasibility of the alignment approach is evaluated by analyzing a third tier ERP system. This system is a multi-module system developed by its vendor over 20 years using object-oriented development techniques. The systems has about 4 million source lines of code, 26,000 classes containing business logics and about 160K associations. IT is a three-tier client-server system though architectural principles, system design and styles of programming as well as functional requirements have experienced many changes and maintenance and development of new functionality have become increasingly complicated. The company has initiated a system's redesign project. In order to simplify the system's design it is attempted to improve decomposition of the system in modules. Given the size of the system, at least initial decomposition is performed using automated clustering techniques. The improved decomposition is envisioned to facilitate assignment of development teams to individual modules of the systems.

The company has about 10 teams working on system's evolution. A team usually includes a product owner, business expert, two to five developers and a couple of tester depending on workload. The business expert represents customer needs and specializes in a particular technical or

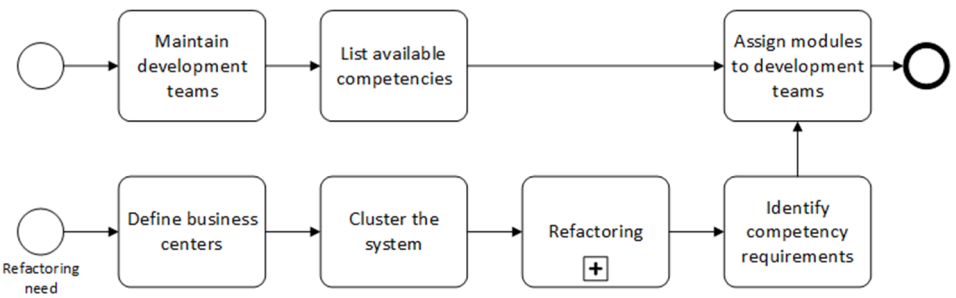

Figure 2: Alignment approach.

functional area. The product owner creates development tasks to implement the requirements. Successful product owners have intimate understanding of functional as well as technical aspects of her modules. Developers and testers have technical competencies and are more productive if they have sufficient understanding and experience about a given module.

45 to 50 tentative business centers are identified. For example, there is an industry specific solution for forest management, which has classes implementing functionality for forest clearances management, wood transportation and billing. Even though clusters are built around the business centers, new clusters also can emerge during the clustering process.

The ERP system is clustered using a hierarchical clustering algorithm (e.g., Cui and Chae 2011). The clustering is performed using a systems representation as a graph as an input. The graph's nodes are source code modules and classes. The graph's edges have several types including uses, extends, implements and other associations. Nodes are attached to clusters to maximize a similarity measure calculated as a weighted sum of edges connecting the node to candidate clusters. The technical description of the clustering algorithm is beyond the scope of this paper and additional details are provided in (Šūpulniece et al., 2015).

The clustering yields around 100 clusters though the right level of granularity is yet to be determined. Figure 3 shows a fragment of high level clustering results. Clusters are shown as bubbles and associations connect interrelated clusters. It can be observed that there is a relatively large number of inter-cluster associations even after the clustering and the clusters have varying degree of centrality.

Figure 4 zooms in on three clusters. The bubble size represents the number of intra-cluster components. Ovals surrounding a bubble and enclosed within a square indicate components having inter-cluster associations. These components are of particular interest because they will serve as interfaces among development teams. One of the clusters identified is a cluster for processing customer payments. This cluster has 229 intra-cluster components and 86 components interfacing with other clusters (there is more than a thousand intra-cluster associations).

The clustering results do not represent a ready-to-be-used new technical design of the system and are not directly transferable to development. It is possible that a single cluster might require different competences due to inefficiency in the current systems design. The clusters will be used by system architects and other stakeholders for discussions on redesigning the system. That will lead to a set of software modules, which could be assigned to individual development teams.

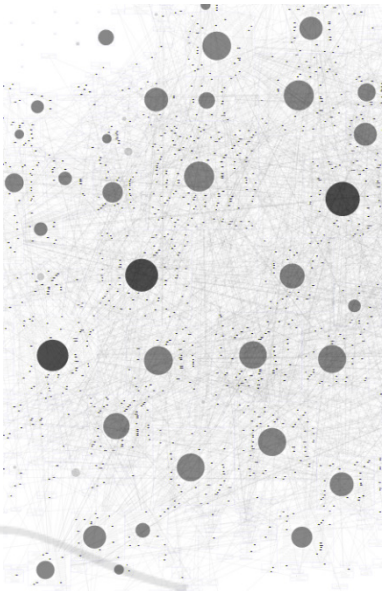

Figure 3: A fragment of clustering results.

Figure 5 illustrates allocation of modules to development teams. This illustration focuses on five tentative modules: 1) financial accounting (FA) billing; 2) sales and distribution (SD) sales order processing; 3) forest management (FM) billing; 4) FM clearance; and 5) FM transportation. The former modules are cross-sectional, while the latter three modules belong to a horizontal solution developed specifically for the forestry industry. The identified competency requirements are given in Table 1 (the knowledge of the base development technologies applies to all modules.

Table 1: Competency requirements for tentative modules.

Module	Required competencies
FA billing	FA
SD sales order processing	CRM
FM billing	FA
FM clearance	FM, GIS integration
FM transportation	FA, GIS integration

Among the development teams, there are teams FA, customer relationships management (CRM) and forest management, respectively. Team FA has expertise in functional aspects of financial accounting what matches to the FA Billing module. Similarly, Team CRM specializes in customer facing

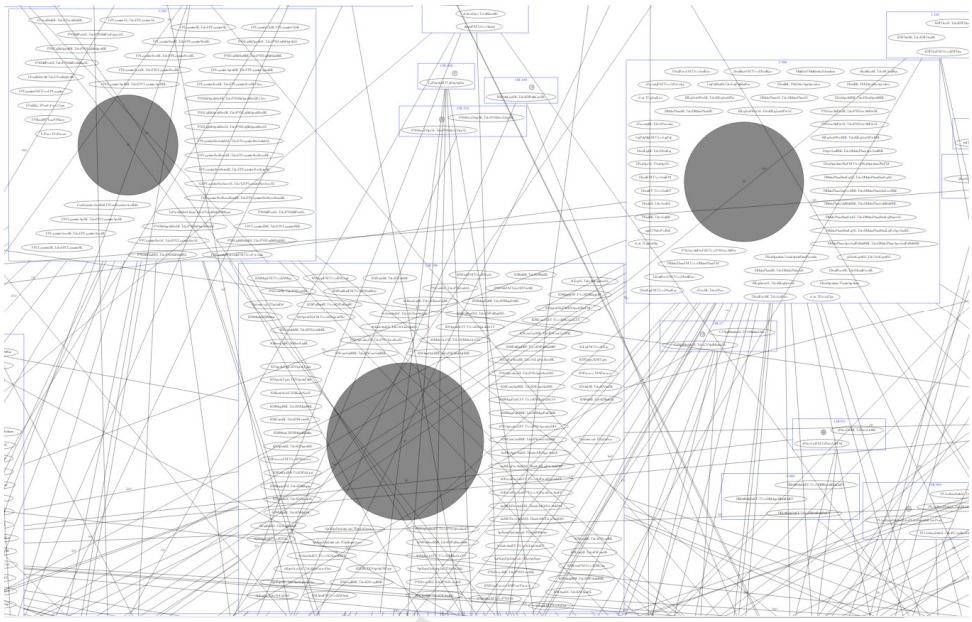

Figure 4: Sample clusters showing intra-cluster and interface components.

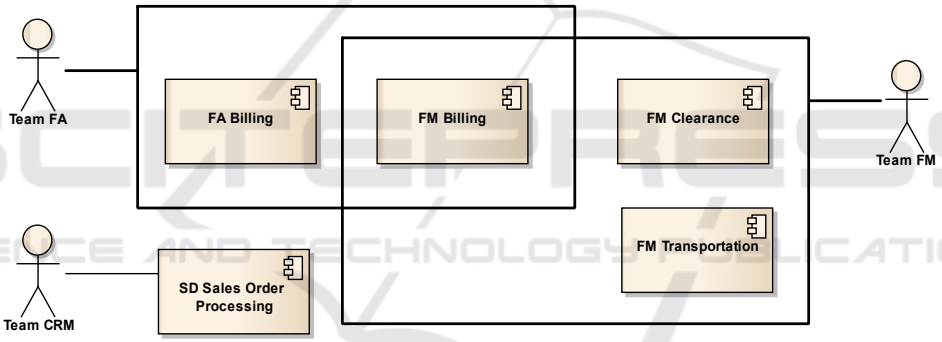

Figure 5: An illustrative matching between teams and modules.

processes what matches to the SD Sales order processing module. Team FM has experience in working with forest management related functionality. However, the FM Billing module also requires FA competencies and there is a decision to be made about allocating this module to one of the teams.

In many cases development teams can be rearranged to find the best fit between modules and teams. Otero et al. (2009) describes a formal approach for assigning teams according to their competencies. This method could be adopted for purposes of this investigation. It also accounts for varying degrees of competency and experience.

5 CONCLUSIONS

The paper proposes a method for automated clustering of enterprise applications as a means for allocating modules to development teams. It is argued that ERP systems are best developed by teams specializing in specific functional and technical areas. The overall architecture is used to allocate modules to these specialized development teams. Clustering is used for automated identification of the modules because manual refactoring is prohibitive. Business centers are used as a starting point of clustering to attain better alignment between software design and expertise of development teams.

The decomposition based allocation is expected to bring the following benefits: 1) teams can specialize in particular functional and technical areas of application development; 2) clear separation of responsibilities among the teams; and 3) faster integration testing (i.e., teams are responsible for

intra-module testing and integration testing focuses only on interface components). From the practical perspective, research results will be used to find the best allocation of modules to development teams and to manage collaboration among the teams. From the theoretical perspective, further research is expected to provide insights in ERP development from the vendor perspective and to evaluate actual benefits of software design refactoring.

There are several challenges to be addressed. The first challenge is finding the appropriate level of granularity or cluster size. The second challenge is definition of modules on the basis of clustering results. A special attention should be devoted to clusters mixing various expertise requirements and to identification of competency requirements for the modules. Finally, the module to team allocation method should be formalized. The granularity level will be determined in experimental studies and by receiving feedback from the development team. The modules will be developed by involving software architecting experts. The evaluation will be performed by means of the case study and comparative analysis of software development efficiency measures.

One of the main challenges is to convince development teams that automated refactoring suggests appropriate solutions for changing the long-established way of working and collaborating among the teams.

ACKNOWLEDGEMENTS

The research is funded by the ERDF project "Information and communication technologies competence center" Nr. KC/2.1.2.1.1/10/01/001 (Contract No. L-KC-11-0003, www.itkc.lv) activity 1.3. "The Method of Monolithic System Decomposition According to SOA Principles."

REFERENCES

Bennett, K.H., Rajlich, V.T., 2000. Software Maintenance and Evolution: a Roadmap. *Proceedings of the Conference on The Future of Software Engineering*, pp. 75-87.

Cai, Z., Yang, X., Wang, X., Wang, Y., 2009. A systematic approach for layered component identification. In 2009 2nd *IEEE International Conference on Computer Science and Information Technology*, pp. 98–103.

Carmel, E., Bird, B., 1997. Small is beautiful: a study of packaged software development teams, *Journal of High Technology Management Research*, 8(1), 129-148.

Cui, J.F., Chae, H.S., 2011. Applying Agglomerative Hierarchical Clustering Algorithms to Component Identification for Legacy Systems. Information and Software Technology, 53(6), 601-614.

De Carvalho, A., Johansson, B., Manhães, R.S., 2010. Agile software development for customizing ERPs. In *Enterprise Information Systems and Implementing IT Infrastructures: Challenges and Issues*, pp. 20-39.

Liang, T.P., Jiang, J., Klein, G.S., Liu, J.Y.C., 2010. Software Quality as Influenced by Informational Diversity, Task Conflict, and Learning in *Project Teams, IEEE Transactions on Engineering Management*, 57(3), 477-487.

Lui, K.M., Chan, K.C.C., 2008. Rescuing Troubled Software Projects by Team Transformation: A Case Study with an ERP Project. *IEEE Transactions on Engineering Management*, 55(1), 171-184.

Monnerat, R.M., De Carvalho, R.A., De Campos, R., 2008. Enterprise systems modeling: The ERP5 development process, *Proceedings of the ACM Symposium on Applied Computing*, pp. 1062.

Otero, L.D., Centeno, G., Ruiz-Torres, A.J., Otero C.E., 2009. A systematic approach for resource allocation in *software projects. Computers & Industrial Engineering 56*, 4, 1333–1339.

Paulish, D., 2002. *Architecture-Centric Software Project Management*. Addison-Wesley, Boston, MA, USA.

Ratner, I.M., Harvey, J., 2011. Vertical slicing: Smaller is better. *Proceedings - 2011 Agile Conference*, Agile 2011, pp. 240-245.

Rettig, C., 2007. The trouble with enterprise software. MIT Sloan Management Review 49(1), 21-27+90.

Riva, R., 2004. *View-based Software Architecture Reconstruction. PhD thesis*, Technical University of Vienna.

Sandkuhl, K., Stirna, J., Persson, A., Wisotzki, M., 2014. *Enterprise Modeling: Tackling Business Challenges with the 4EM Method*. Springer, Berlin.

Sommerville, I., 2010. *Software Engineering*. Person, 9th Edition.

Šūpulniece, I., Polaka, I., Bērziša, S., Ozoliņš, E., Palacis, E., Meiers, E., Grabis, J., 2015. Source Code Driven Enterprise Application Decomposition: Preliminary Evaluation. ICTE in Regional Development 2015 Valmiera, Latvia, Procedia Computer Science 77, pp. 167-175.

Xu, L., Brinkkemper, S., 2007. Concepts of product software. *European Journal of Information Systems. 16(5)*, pp. 531-541.

Unphon, H., Dittrich, Y., 2010. Software architecture awareness in long-term software product evolution. *Journal of Systems and Software 83(11)*, 2211-2226.