Evaluating Enterprise Resource Planning Analysis Patterns Using Normalized Systems Theory

Ornchanok Chongsombut[1] and Jan Verelst[2]

Department of Management Information Systems, University of Antwerp, Antwerp, Belgium
Ornchanok.chongsombut@uantwerp.be and jan.verelst@uantwerp.be

Abstract. Normalized Systems theory provides an important practical way of developing evolvable information systems, even huge application systems for organizations. The purpose of the paper is to present an analysis of the analysis patterns of the well-known Microsoft Dynamics CRM 2016 adhering to the design patterns of Normalized Systems theory.

Keywords: Normalized Systems; Evolvability; ERP; Analysis Patterns; Microsoft Dynamics CRM

1 Introduction

At present, one of the most challenging aspects of designing enterprise information systems is evolvability. ERP systems are costly and time-consuming to developed. Moreover, ERP systems have extremely complicated structures, and therefore, they are not easy to implement. Due to both the complexity of ERP Systems and organizations' requirement for customized solutions to serve their business objectives, ERP systems should be evolvable. Here, evolvability means software should be easy to change over time [1, 2].

To ensure the evolvability of information systems that adhere to Normalized Systems theory, it has been argued that that information systems should be developed without *combinatorial effects.* Combinatorial effects occur when the impact of a change depends on the size of the information systems. To increase the evolvability of information systems, these combinatorial effects should be minimized. To date, a number of studies have already been done on Normalized Systems theory and implemented in several software projects [2-4]. Nevertheless, the analysis pattern of ERP packages applying Normalized Systems theory has a number of limitations.

2 Normalized Systems Theory

Normalized Systems theory provides a practical way of developing evolvable information systems through the so-called pattern expansion of software elements [3].

2.1 Normalized Systems Theorems

To guarantee high evolvability of information systems, Normalized Systems theory proposes four theorems such as Separation of Concerns, Data Version Transparency, Action Version Transparency, and Separation of States [2]. Furthermore, how the four Normalized Systems theorems are manifested in a practical way is shown in Table 1.

Table 1. The Normalized Systems four theorems in practice [5]

Normalized Systems Theorems	The practical way of developing information systems
Separation of Concerns	• Multi-tier architectures separating presentation logic, application or business logic, database logic, etcetera
Data Version Transparency	• Polymorphism in object-orientation • Wrapper functions
Action Version Transparency	• XML-based technology (e.g., for web services) • Information hiding in object-orientation
Separation of States	• Asynchronous communication systems • Stateful workflow systems

2.2 Normalized Systems Elements

Five expandable elements were proposed to ensure the evolvability of Normalized Systems applications. The internal structure of these five elements is described by Normalized Systems design patterns such as data elements, action elements, workflow elements. connector elements, trigger elements [2-4].

3 Analysing the partial analysis patterns of the ERP package

In this section, we examined the Microsoft Dynamics CRM 2016 from an evolvability point of view and demonstrate both conformance with Normalized Systems theory and violations against it.

3.1 Indications towards of conformance with Normalized Systems principles

Here, our aim is to examine conformance between the model of Microsoft Dynamics CRM and Normalized Systems principles.

Based on the NS theorems, most of the model of Microsoft Dynamics CRM seem to be related to the NS principles. Firstly, the Microsoft Dynamics CRM architecture is the Multi-tier architectures. Moreover, the Microsoft Dynamics CRM implements cross-cutting concerns, for example, Reporting (Dashboards, Charts, Excel and SRS), Security model that focuses on access rights to the entities in the system [6-8]. For first and second points straightforwardly follow from *the Separation of Concerns theorem*.

According to *the Data version transparency theorem*, focusing on the interaction between data and action entities. Data Version Transparency implies that data entities

can be modified (insert, delete, update) without affecting the calling actions [9]. In Microsoft Dynamics CRM, the information hiding has been applied to develop the software. Properties cannot be directly accessed, but can be read or written by using provided-method. Additionally, Microsoft Dynamics CRM has been implemented using XML-based technology that leads to conformance with the Data Version Transparency theorem.

Following *the Action version transparency theorem*, this theorem implies an action can be modified without affecting the calling actions. First, Microsoft Dynamics CRM is usually implemented through wrapper functions through the use of polymorphism in C#.NET or VB.NET. Second, the Microsoft Dynamics CRM implements cross-cutting concerns as explained above. Therefore, the developing of Microsoft Dynamics CRM relates the Action version transparency theorem.

The Microsoft Dynamics CRM relies on asynchronous service to improve overall system performance and scalability [10]. Combinatorial effects can be avoided through asynchronous processing.

3.2 Indications towards violation of NS principles

When analysing the analysis patterns of Microsoft Dynamics CRM, some indications towards violation of the Normalized Systems principles might be noticed. Microsoft Dynamics CRM addresses challenge of customer management, therefore, this module was analysed in point of evolvability. We noticed that there is duplication of attributes about address details in many classes such as Account, Contact, Address, Lead, LeadAddress, Quote, Invoice, and Order. Attribute duplication seems contradictory to the Normalized Systems theorem, Separation of Concern. According to the assumption of unlimited systems evolution, software can be changed over time. Therefore, the eventual impact might become related to the overall system size and lead to a combinatorial effect.

4 Conclusion

In this paper, we analysed the analysis patterns of ERP package, Microsoft Dynamic CRM, to explore conformance with Normalized Systems theory and violations against it. While the interpretation of the analysis patterns of ERP package shows some conformance towards Normalized Systems theory, also some analysis patterns towards violations of Normalized Systems. The finding was found the developing well-known commercial ERP system seems to relate Normalized Systems theory both four theorems and five elements [5, 9]. On the other hand, a few points seem to contradict Normalized Systems principles. Similarly, the finding of Mendling et al [11], there are some error probabilities in enterprise models.

The limitations of our study need to be acknowledged. We only analysed partial analysis patterns of one ERP package. As part of future research, we will redesign and

rebuild the existing data model of existing ERP software packages based on Normalized Systems theory. In practice, we will rebuild existing ERP packages using the Normalized Systems expander to obtain high evolvability.

5 References

1. Verelst, J., et al., *Identifying Combinatorial Effects in Requirements Engineering*, in *Advances in Enterprise Engineering VII: Third Enterprise Engineering Working Conference, EEWC 2013, Luxembourg, May 13-14, 2013. Proceedings*, H.A. Proper, D. Aveiro, and K. Gaaloul, Editors. 2013, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 88-102.
2. De Bruyn, P., et al., *Towards Applying Normalized Systems Theory Implications to Enterprise Process Reference Models*, in *Advances in Enterprise Engineering VI: Second Enterprise Engineering Working Conference, EEWC 2012, Delft, The Netherlands, May 7-8, 2012. Proceedings*, A. Albani, D. Aveiro, and J. Barjis, Editors. 2012, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 31-45.
3. Oorts, G., et al. *Building evolvable software using normalized systems theory: A case study*. in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. 2014. IEEE.
4. Oorts, G., et al., *Easily evolving software using normalized system theory-a case study*. Proceedings of ICSEA, 2014: p. 322-327.
5. De Bruyn, P., D. Geert, and H. Mannaert, *Aligning the Normalized Systems Theorems with Existing Heuristic Software Engineering Knowledge*. he Seventh International Conference on Software Engineering Advances, 2012: p. 84-89.
6. Microsoft. *The extensibility model of Microsoft Dynamics CRM*. 2015 [cited 2017; Available from: https://msdn.microsoft.com/en-us/library/gg327974(v=crm.7).aspx#BKMK_architecture.
7. Microsoft. *The programming models for Microsoft Dynamics 365*. 2016 [cited 2017; Available from: https://msdn.microsoft.com/en-us/library/gg327971.aspx.
8. Microsoft. *Microsoft Dynamics architecture*. 2017 [cited 2017; Available from: https://msdn.microsoft.com/en-us/library/aa496912(v=ax.10).aspx.
9. Mannaert, H., J. Verelst, and K. Ven, *Towards evolvable software architectures based on systems theoretic stability*. Software: Practice and Experience, 2012. **42**(1): p. 89-116.
10. Microsoft. *Asynchronous service in Microsoft Dynamics CRM 2016*. 2016 [cited 2017; Available from: https://msdn.microsoft.com/th-th/library/gg328021(v=crm.8).aspx.
11. Mendling, J., et al., *Errors in the SAP reference model*. BPTrends, 2006. **4**(6): p. 1-5.