International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:15, No:1, 2021

Blueprinting of a Normalized Supply Chain Processes: Results in Implementing Normalized Software Systems

Bassam Istanbouli

***Abstract*—**With the technology evolving every day and with the increase in global competition, industries are always under the pressure to be the best. They need to provide good quality products at competitive prices, when and how the customer wants them. In order to achieve this level of service, products and their respective supply chain processes need to be flexible and evolvable; otherwise changes will be extremely expensive, slow and with many combinatorial effects. Those combinatorial effects impact the whole organizational structure, from a management, financial, documentation, logistics and specially the information system Enterprise Requirement Planning (ERP) perspective. By applying the normalized system concept/theory to segments of the supply chain, we believe minimal effects, especially at the time of launching an organization global software project. The purpose of this paper is to point out that if an organization wants to develop a software from scratch or implement an existing ERP software for their business needs and if their business processes are normalized and modular then most probably this will yield to a normalized and modular software system that can be easily modified when the business evolves. Another important goal of this paper is to increase the awareness regarding the design of the business processes in a software implementation project. If the blueprints created are normalized then the software developers and configurators will use those modular blueprints to map them into modular software. This paper only prepares the ground for further studies; the above concept will be supported by going through the steps of developing, configuring and/or implementing a software system for an organization by using two methods: The Software Development Lifecycle method (SDLC) and the Accelerated SAP implementation method (ASAP). Both methods start with the customer requirements, then blue printing of its business processes and finally mapping those processes into a software system. Since those requirements and processes are the starting point of the implementation process, then normalizing those processes will end up in a normalizing software.

***Keywords*—**Blueprint, ERP, SDLC, Modular.

I. Introduction

IN 2005 we were involved in a full scale implementation of an ERP SAP system to replace an existing legacy software. That implementation is one of the reasons why this paper is being written, because if knowing about normalized systems at the time of implementation, most of the requirements and the supply chain business processes would have been designed differently, especially as things changed later and the organization started growing fast; it was very costly to implement the changes and sometimes we were forced to customize because it was cheaper, easier and a faster solution but of course customization made the system more rigid and again any later changes became costlier and more difficult to implement. Hence one of the goals of this paper is to increase the awareness when designing business processes because they are the starting point of implementing new software.

In Fig. 1, we can see that a supply chain process and workflow is mapped into a software package, in my example the software package is SAP. That block workflow initiates the whole implementation process.

If the blueprints created are modularized then the software developers/implementers will use those modular blueprints to map them into modular software that is by normalizing the business processes they are helping the software creators in normalizing their software.

Approach and Explaining the Concept

To prove the above concept, we will go through the steps of developing, configuring and/or implementing a software system for an organization by using two methods: The SDLC and the ASAP method for an ERP software. Both methods start with the customer requirements, to the blueprinting of its business processes and finally mapping those processes into a software system. By showing that a software implementation always starts with a business requirements and what the "to be" processes should be, this will be proving that those processes are the launching pad of the software system developed. Those requirements, presented with work-flows and blueprints, drive the business analysts and developers to create or tailor the software system accordingly. They are the platform of the software system developed. Hence by normalizing the blueprints the created software will most probably be normalized.

II. Definitions

Following are some of the definitions that will help readers.

Normalized Systems and Modularity

"Normalized Systems (NS) are new modular structures with unique evolvability characteristics where combinatorial effects are systematically controlled or eliminated." [1]

Modularity is about breaking down large complex systems into smaller, loosely coupled blocks. It is a way to manage a complex system by dividing it into modules "—which can then communicate with one another only through standardized interfaces within a standardized architecture—one can

Bassam Istanbouli is with the University of Antwerp, Belgium (e-mail: bistanbouli@gmail.com).

eliminate what would otherwise be an unmanageable spaghetti tangle of systemic interconnections." [2, p.19].

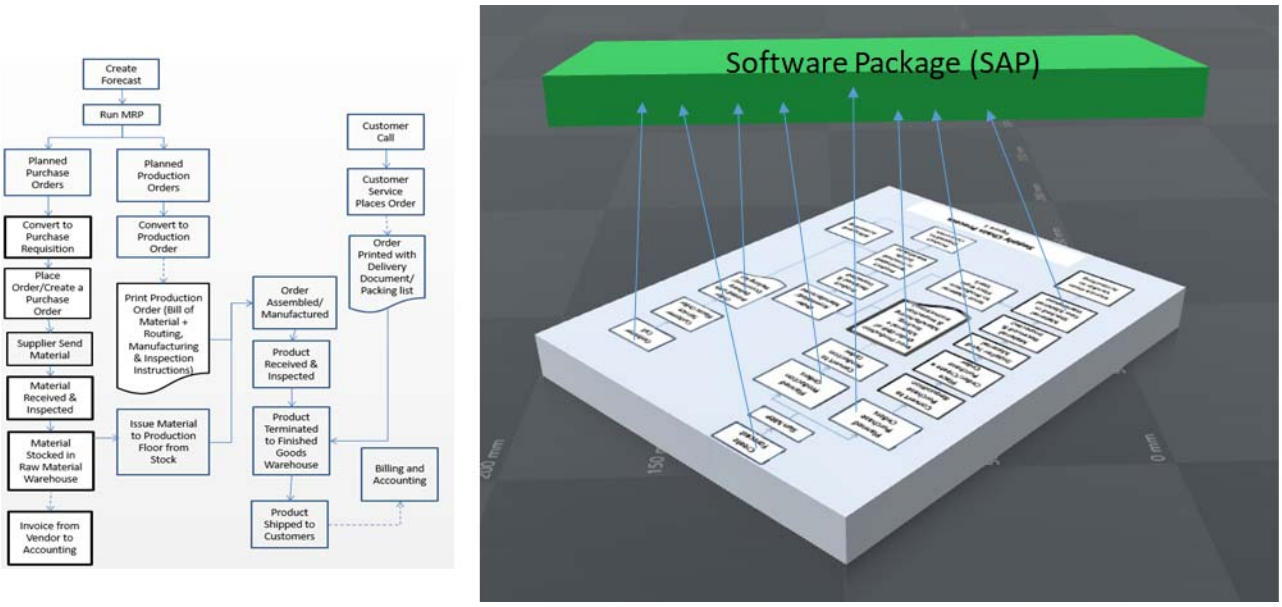
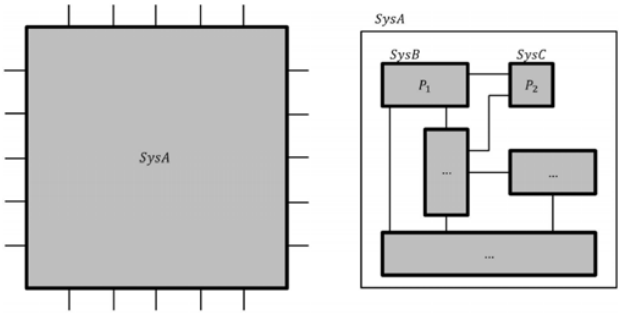


Fig. 1 A model of a workflow process mapped into a software package

Fig. 2 it represents a system A, part (a) as one block with various inputs and outputs, and part (b) is the same system but broken into smaller modules.



(a) (b)

Fig. 2 (a) System A as one black box, (b) System A broken into smaller modules [11, p.101]

"Modules are made of submodules which in turn can be divided into submodules" [11]; hence, a complex system can be divided into modules which in turn can or may be divided into modules and so forth until we reach an optimum solution for our analysis or get to lowest module which maybe at the component level.

When a system is broken down into smaller sub-systems, those sub-systems/modules should be simple and should be able to be changed without the knowledge of other modules or affecting them. Also the changes should not affect the interfaces. Hence major changes can be done to individual modules only [4].

Combinatorial Effects

It is the impact on a system proportional to the system's size, not to the size of the change: "Functional changes causing impacts that are dependent on the size of the system as well as the nature of the change correspond to instabilities of the information system." Those instabilities are called combinatorial effects [11, p.270].

Evolvability

Evolvability was mostly defined for software, but in our case it also can be related to processes. One of the good definitions related to this subject is: "we describe evolution as changes in a system's environment (domain), requirements (experience) and implementation technologies (process). Then we define evolvability as a system's ability to survive changes in its environment, requirements and implementation technologies." [5, p.2].

Enterprise Resource Planning

According to Shehab et al. "Enterprise resource planning (ERP) system is a business management system that comprises integrated sets of comprehensive software, which can be used, when successfully implemented, to manage and integrate all the business functions within an organization." [3, p.1].

"In the previous information systems, processes cannot communicate effectively with each other, owing to many databases and conversion of data from one system to another was expensive. ERP systems were found to be a solution to these problems as they are configurable information system packages that integrate several business functions into a single

system with a shared database." [6, p.1].

Supply Chain Management

A good definition of supply chain management by van der Vorst: "…we take a process view, which means we look at a supply chain as a sequence of (decision making and execution) processes and (material, information and money) flows that aim to meet final customer requirements and take place within and between different supply chain stages. The supply chain not only includes the manufacturer and its suppliers, but also (depending on the logistics flows) transporters, warehouses, retailers, and consumers themselves." [7, p.2].

SDLC

To design a software system for an organization we need to understand the objective, structure and the processes of that organization [8]

As seen in Fig. 3, the Software/System Development Life Cycle in an ongoing process, that keeps in repeating the cycle whenever there is a new requirement happening. This is one of the main reasons that whatever we design needs to be flexible and easy to change.

It is always a team's effort that is organized in a project form which includes management, system analysts, programmers and super users from the business. The team contains representatives from all groups/departments involved.

The goal of this paper is discussing the process, step by step, by which a software system or application is being designed from scratch or changed. We want to get to the point where the programmers convert the system specification and the organization requirements into instructions and codes, hence showing that it all depends on how well the initial processes and specifications were created and presented to the software team.

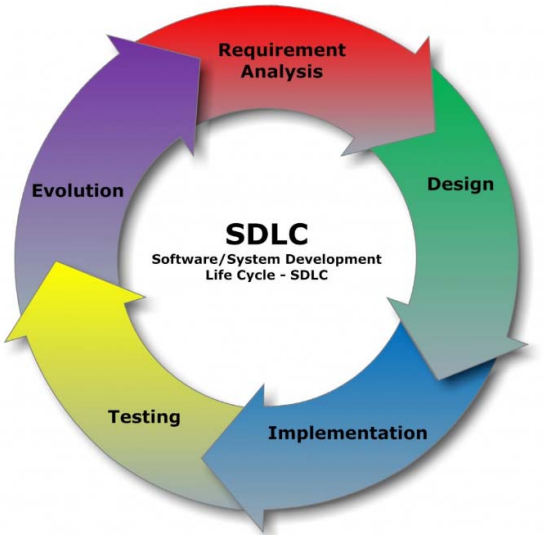


Fig. 3 SDLC Cycle [9]

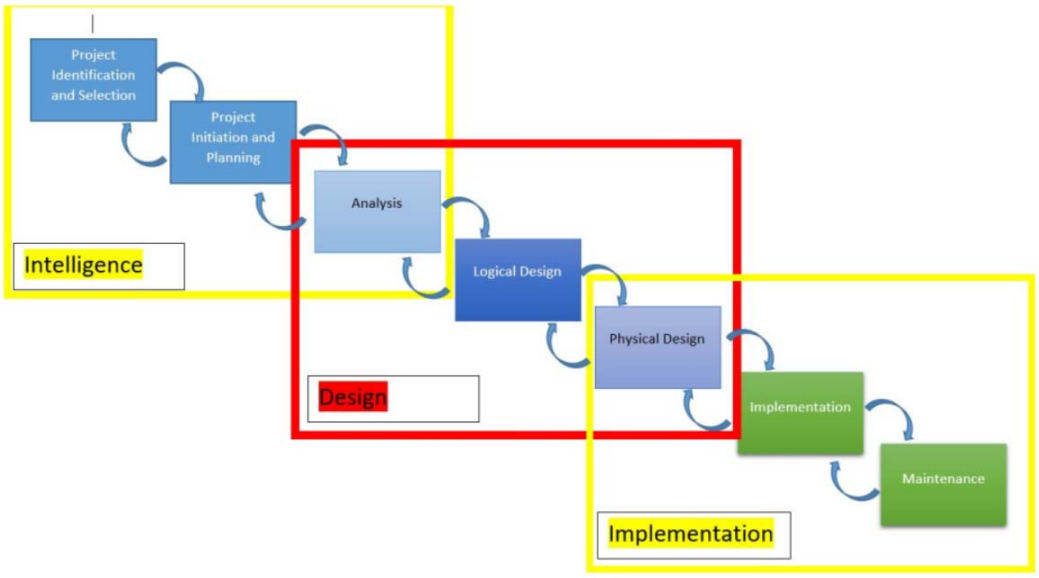


Fig. 4 SDLC Flow Diagram

As mentioned in Modern Systems Analysis and Design by Hoffer: business managers and functional team members are important for system development because they can set up the general requirements of the system together. [10]

In this paper we will use the System Development Life Cycle (SDLC) model as one of the examples to present our argument. "The System and Development Life Cycle is a common methodology for system development in many organizations, featuring several phases that mark the progress of the system analysis and design effort." [10, p.24].

In this paper we are interested in the design phase which compromise of the Analysis, Logical Design and Physical Design as seen in Fig. 4.

The analysis phase focuses on studying the organization's

current processes and procedures. It includes the current "information system" used, if any. Also in this phase the "to-be" system is discussed with a gap analysis between the current and future processes/system; that is what the users/ business managers require from the new system in comparison to the current system. [10]

The requirements are studied and alternatives are discussed with the possibilities of any re-engineering that needs to take place in order to get the best requirements implemented. Since this phase is the starting point of the system's design, we are suggesting that the team working on those steps need to think evolvability of the business processes. The team is building the platform and the initial structure of the system so the more they make those building blocks flexible, the better they are when changes start hitting the organization.

The following phases are the logical and physical design where the features of the "system analyzed" are described, programmed and then implemented. Hence in the analysis block, the functional specifications and the processes are defined and then all the diagrams, work flows and processes are converted into "structured systems design, which can be broken into smaller and smaller units for conversion to instructions written in a programming language." [8, p.28].

ERP/SAP Implementation

ERP implementation will be the second example on how a universal software package also starts with requirements and "to be" processes. In the example the ASAP method will be used.

ASAP

As seen in Fig. 5, there are 5 phases for an ASAP implementation. The part that we will focus on in this paper is phase number 2, which is the business blueprint phase. In this phase, the implementation team does the "Gap Analysis" between the current processes and what the to-be desired processes. The core team creates the requirements of the new system and defines the rules, the interfaces, the processes, the work flows and the integration points.

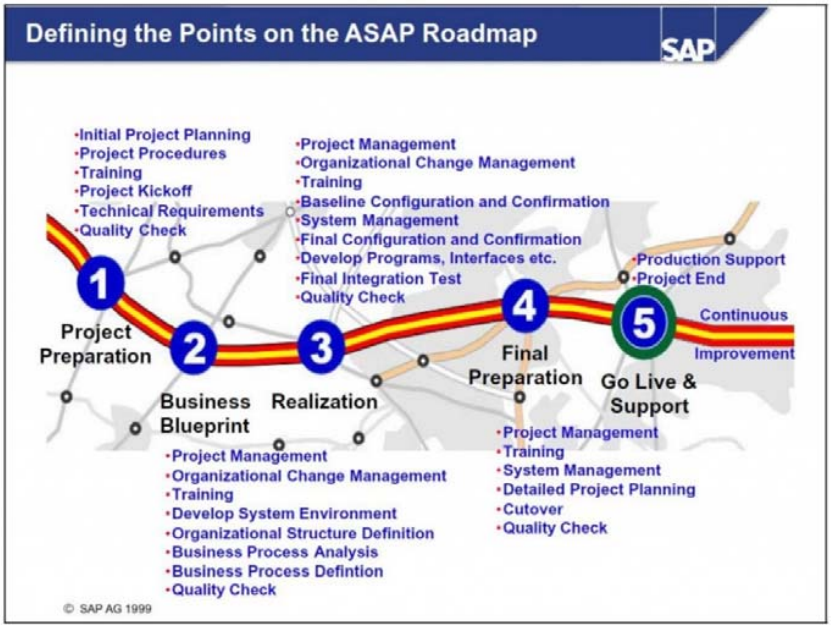


Fig. 5 ASAP Methodology [9]

It is important to emphasize that the processes and work flows are the starting point of any software implementation and that "a work-flow is generally brought into existence by a trigger. This can be for example, an event in the form of a customer order, the attainment of a point in time, or the end of a time period. A work-flow usually consists of process steps (actions, activities, tasks) that can in turn, be subdivided- if necessary at several levels." [12, p.46].

In the business blueprint phase the functional and technical specifications are created for the configurators and developers to implement. [9].

When the blueprinting is designed then the realization phase starts. This phase is mainly configuring the system by mapping the blueprinting and the "to be" processes into the software. If the configuration was not possible then customization occurs, which can be creating transactions to do the required job or sometime re-engineering the business processes to facilitate a configuration and avoid customization. We will not dive into that path and we will assume that according to SAP's tens of thousands of transactions, we can always find a transaction TRANn which will satisfy the part of the process (a single process block) that

we are configuring and give us exactly the results needed. For example, if we want to create a production order, we can use TRAN1 in SAP to do it or if we want to print a production order packet we can use TRAN2 to give me three documents, and so forth. Then in phase 3, there is a unit testing phase where we will test our individual transactions, followed by an integration testing that will test the whole configured workflow and business processes; the "to be" processes that we based our requirements upon. Hence in the integration testing we actually confirm that our blueprints created were mapped correctly into the software, as required. The idea here is not to talk about the implementation process but to emphasize that the starting point of an ERP implementation is again the processes and work flows; that is the blueprints. Hence if a process is not designed in a way that it can be evolvable, then the software that was mapped into will not be evolvable.

III. Conclusion

As seen in previous sections, if we can apply the normalized systems concept to the supply chain processes and make them modular, then implementing any changes in the future will be easier.

The goal of this paper, as mentioned in the introduction, is to increase the awareness among the implementation team. They need to know that it is very important to modularize the initial business processes in the blueprints, because it will impact the whole software system. The impact will not be for that initial implementation phase only, but for years to come as the business changes and grows.

If the system is not normalized the implementation team will be creating a very rigid system that will be very difficult and expensive to make changes to it later.

Also, as mentioned before, this paper does not prove the concept suggested. To be able to prove it we will need an implementation team that will create a case study before and after normalization of the business processes.

If we apply the normalized system concept at the design level and create blueprints of smaller modules of supply chain processes and submit them to the programmers, then the software implemented will be modular and any changes in the future to the processes or specification will impact smaller subsystems

References

[1] https://normalizedsystems.org/
[2] Langlois, R.N., 2002. Modularity in technology and organization. *Journal of Economic Behavior & Organization*, Vol. 49, pp 19-37
[3] Shehab, E.M., Sharp, M.W., Supramaniam, L., and Spedding, T.A., 2004. Enterprise Resource Planning: An integrative review. Business Process Management Journal 10(4).
[4] Parnas, D.L., Clements, P.C. and Weiss, D.M., 1984. The Modular Structure of Complex Systems. *IEEE*, 1984, pp 408-417.
[5] Ciraci, S. and van den Broek, P., 2006. Evolvability as a quality attribute of software architectures. *Journal of Physics Conference Series, January 2006*.
[6] Motwani, B., 2017. Impact of Resources in Enterprise Resource Planning (ERP) Implementation Process on Internal Process of an Organization. *Amity Business Review*, 2017. Vol. 18, No. 1.
[7] Supply Chain Management: theory and practices (2004) - Dr. Ir. Jack G.A.J. van der Vorst
[8] Normalized Systems Theory, From Foundations for Evolvable Software Toward a General Theory for Evolvable Design, Herwig Mannaert – Jan Verelst – Peter De Bruyn.
[9] https://toughnickel.com/business/ASAP-Methodology-SAP-Implementation-Phases
[10] Modern Systems Analysis & Design by J.A. Hoffer, J.F. George and J.S. Valaxich
[11] Mannaert, H., Verelst, J., De Bruyn, P., 2016. Normalized Systems Theory, From Foundations for Evolvable Software Toward a General Theory for Evolvable Design.
[12] SAP R/3 Process Oriented Implementation, Gerhard Keller, Thomas Tenfel.