What the Current System Development Trends tell us about Systems Development Methodologies: Toward explaining SSDAM, Agile and IDEF0 Methodologies

Abdulla F. ALLY, Zhang NING
University of Shanghai for Science and Technology, Shanghai 200093, China
abdulla@udsm.ac.tz

Abstract—Systems integration, customization and component based development approach are of increasing attention. This trend facilitates the research attention to also focus on systems development methodologies. The availability of systems development tools, rapid change in technologies, evolution of mobile computing and the growth of cloud computing have necessitated a move toward systems integration and customization rather than developing systems from scratch. This tendency encourages component based development and discourages traditional systems development approach. The paper presents and evaluates SSADM, IDEF0 and Agile systems development methodologies. More specifically, it examines how they fit or not fit into the current competitive market of systems development. In the view of this perspective, it is anticipated that despite of its popularity, SSADM methodology is becoming obsolete while Agile and IDEF0 methodologies are still gaining acceptance in the current competitive market of systems development. The present study more likely enrich our understanding of the systems development methodologies concepts and draw attention regarding where the current trends in system development are heading.

Keywords: Systems development trends, systems integration, systems analysis, system development life cycle.

I. INTRODUCTION

Systems development is one of the key issues in the field of information systems [1]. In view of this, various system development methodologies were built to guide the developers in the systems development process. Such methodologies include Structured Systems Analysis and Design Methodology (SSADM), Integrated Definition (IDEF) and Agile methodology. Each methodology has a specific framework. This framework controls and organises the system development processes. However, each is best suited to specific kinds of systems [2]. The system development methodologies have evolved over time, each with its own advantages and disadvantages. One of the widely-cited advantages of using methodologies to systems development industry is that it makes the development process more genial to project management, monitoring and control [3, 4]. But these benefits are not likely to be taken into consideration in practice in most of the systems development projects. This is probably because of reluctance of technology transfer [5] or ignorance among systems developers [6]. Some methodologies were derived from practice and concepts that are suitable to the old organizational environment [7]. Hence they require reconsideration of their roles in order to fit into today’s systems development markets. In today’s world of evolving technology, people start to deviate rigid and traditional methodologies and stick to systems integration, customization and component based development approach. In the present paper, SSADM, IDEF0 and Agile systems development methodologies are presented and examined. However, the emphasis of the paper is on the evaluation of the methodologies and their suitability into the current business and competitive environment of software development. The present study is devoted to furthering our understanding of systems development methodologies concepts and draw attention regarding where the current trends in systems development are heading.

II. ANALYSIS AND DISCUSSION

A. An Overview of the Current System Development Trends

The evolving technology has expanded to the point where a system (software) has become an important to do our daily activities. The system development is an integral part of our everyday life. The current trends in system development are leaning toward enriching mobile platforms, cloud computing platforms and much more to enable customers to do business seamlessly anywhere they go.

Within the mobile society we live in, the developers focus on enriching mobile platforms with more programs in order to enable customers to virtually do their business through their phones. On the other hand, the attention has also turned to cloud computing development. The Cloud computing development is the result of the evolution and adoption of existing technologies and paradigms. It describes a model on which a computing infrastructure is viewed as a cloud, whereby business organizations and individuals access applications from anywhere in the world on demand [8]. Cloud computing is now leading the world following the rapid growth of the demand for efficient and economical computational power [9]. A lot of business people and the major cloud technology developers continue to develop and
invest in emerging markets of cloud computing [10, 11]. According to William et al [12], cloud computing providers offer some Application Programming Interface (APIs) and systems development tools for developing clouds based solutions. This cloud tools allow developers to build seamlessly scalable applications in order to allow customers to run their ever day infrastructure in the cloud [12].

The increase in mobile platforms and cloud development facilitate attention to focus on systems integration and customization than developing systems from scratch. Companies might want to outsource the services to developers which may have readymade systems. These readymade systems may only require customization in order to meet user’s requirements. Today’s systems development projects evolve rapidly with changing expectations and competitive market [13]. This trend necessitates a move towards Rapid Application Development (RAD) that is based on the frequent tangible results which focuses on delivery of some functional components of the system in a short timescales [7]. The RAD involves iterative development and the building of prototype [13]. The delivery of a prototype allows systems developer to capture requirements in a more comprehensive way [2]. Furthermore, in modern competitive environment, changes to any system under development and pressure of the customer for immediate implementation of the system [14, 15] cannot be avoided. This market competition facilitates frequent change in systems requirement and the change might be very late in the development process which needs to be accommodated in order to meet customer’s satisfaction. In other words, customer might not be fully knowledgeable in the business and application. In this way, continuous evolutions of systems requirements are frequent. Nowadays, customers are interested in continuous improvement, frequent and flexible response to change and early delivery of the systems. This tendency necessitates a move from strict development approach towards the systems customization and collaborative development such as that adopted in launchpad.net projects.

B. A Conceptual Basis for the Methodologies

In the history of software development industry, system development was based on random and unsystematic methodological approach [7, 16]. However, some systematic methods were also available [17]. In either case, system development methodology can be described as an expression of the life cycle that is best suited for the development of a given system [18].

According to Bechoo lal et al [13], the systems development life cycle (SDLC) emerged in 1960’s and can be considered as the oldest systematic system development methodology [19]. The basic idea of SDLC is splitting up of system development into a series of distinct and sequential phases [19, 20], namely planning, analysis, design, development, testing, implementation, and maintenance as shown in fig.1.

- **Planning phase**: The primary activity that is carried out during this phase is to determine which system is required to meet organization objectives. This is where we develop the project plan.
- **Analysis phase**: It involves gathering business requirements and creates process diagrams.
- **Design phase**: It involves designing system model.
- **Development**: This stage involves developing databases and programs.
- **Testing**: Perform system testing with test conditions.
- **Implementation**: Writing a detailed user documentation and provide training for the systems users.
- **Maintenance**: Perform system maintenance and provides an environment to support system changes.

Basically, the SDLC is the foundation for all systems development methodologies [21]. However, their underlying principles are different. One of the early systematic methodologies emerged after SDLC is the Structured Systems Analysis and Design Methodology (SSADM) [13]. Since then a number of methodologies arose. These include Agile development methodology and IDEF0 methodology. The SSADM have been developed after the request of the United Kingdom Government [22], and was mainly practised by the government institutions, Fig. 2 shows the stages of SSADM methodology. Essentially, SSADM sticks to the traditional waterfall model [23], which requires review and commitment of each phase before starting the next phase. Once pass the phase you cannot go back to the previous phase. SSADM focuses on the Analysis and Design stages of the SDLC. The main problem with traditional waterfall model in the system development is its rigidity and inflexibility applied [4]. A number of previous studies criticized on the phase dependence of the waterfall life-cycles (see reviews in [20, 24, 25]). Other studies such as [26] pointed out there is imbalance between analysis and synthesis in the waterfall life cycle.
Moreover, this methodology requires all the requirements to be well understood from the beginning, yet client or systems users may not be able to clearly define all important requirements early in the project. That is to say, any changes that occur later in the project are more costly and thus are discouraged. Furthermore, SSADM provides little room to iterate development process hence reduce manageability when used.

In contrast to a more current agile systems development methodology, agile methodology allows evolution of systems requirements and solutions through collaboration [14]. It divides the system into small incremental functions, which are provided in iterative fashion [15] as shown in fig. 3. Each iteration involves teams that are working together on various phase to produce deliverables.

In this way, change in system requirement can be easily reviewed and accommodated at any time during the development process [27]. Literature shows that a key principle of agile methodology is based on agile manifesto [27, 28] which is constructed under the following key principles [14, 15], summarised as follows: The delivery of the working system should be rapid and frequent to meet customer requirements; Change in systems requirements are to be accommodated at any time during the development process; The working systems is the principal measure of progress; Best communication is face to face which facilitate good cooperation between customers and developers to establish and review systems requirements; Systems are built around motivated individuals, who should be trusted; The System development should be sustainable to maintain a constant pace and the development team should be self-organized; Regular adaptation to changing circumstance; simplicity is essential. Moreover, continuous attention to technical excellence and good design is important.

Phases of Agile methodology are illustrated in fig. 4 [29]. Agile methodology has focused on quick responses to requirement change and continuous development of the systems [30]. The process of Agile system development had started early in the system development industry and it is now becoming more popular due to its adaptability and flexibility. A more recent survey in [31] reported that 73% of respondents agree that agile methodology helps to complete software projects faster; 92% say it improves ability to accommodate changes to meet customer satisfaction; and 87% say that it improves productivity.

On the other hand, Integrated Definition (IDEF) refers to a family of modelling language, which covers a wide range of uses, from functional modelling to data, simulation, object-oriented analysis/design and knowledge acquisition [32]. The following is a list of the IDEF methods that are most commonly used in the field of software engineering [33] namely IDEF0: Functional modelling; IDEF1: Information modelling; IDEF1X: Data modelling; IDEF2: Simulation model design; IDEF3: Process description capture; IDEF4: Object-Oriented design; IDEF5: Ontology description capture; IDEF6: Design rationale capture; IDEF7: Information system auditing; IDEF8: User interface design; IDEF9: Scenario-Driven IS design; IDEF10: Implementation architecture modelling; IDEF11: Information artefact modelling; IDEF12: Organization modelling; IDEF13: Three schema mapping design and IDEF14: Network design.

IDEF methodology was originally built for manufacturing environment. However, its uses have now been adapted for wider use and for software development in general [35-37]. IDEF0 is a part of the IDEF family of modelling languages, and is built on the functional modelling language Structured Analysis and Design Technique [33]. IDEF0 was created to model the decisions, actions, and activities of an organization or system [34].
For new systems, IDEF0 can be used to define the requirements and specify the functionalities of the system, and then to design the system that meets the customer’s satisfactions. For existing systems, IDEF0 can be used to analyse the functionalities of the system and to record its underlying mechanisms. The result of applying IDEF0 to a system is a model that consists of a series of diagrams and text which helps systems developers to visualize a current system or a new system that may be necessary to meet new requirement. The IDEF0 model for system development is illustrated in fig. 5 [33]. IDEF0 comprises of functional modelling language for the systems analysis and design; systems development; business processes and integration of information systems [35].

Previous studies reported that an IDEF0 had been widely accepted [36-44]. It offers, among others, the following advantages: effective communications between system analysts and customers [33-34, 36-44] thereby facilitate better capture of systems requirements; better execution of information systems analysis and design [36-44]; and good management of large and complex information systems projects [36-44].

C. SSADM Versus Agile Versus IDEF0 Systems Development Methodologies

It is worth mentioning here that all methodologies outlined in this paper follow the same phases of primitive Waterfall system development model. However, some methodologies have rectified the phases and the principles of Waterfall model while others have not.

For example, SSADM completely concentrates on the analysis and design phase of the Waterfall Model. Hence SSADM is a Waterfall model [23] and has all advantages and disadvantages of waterfall model.

Whereas IDEF0 and Agile are another successors of Waterfall model, which have all the advantages of the Waterfall approach but its disadvantages were rectified. Fig. 6 shows the differences between SSADM, Agile and IDEF0 methodologies in relation to the development of information systems.

The main advantage of Agile and IDEF0 is the regressive scalability and flexibility. They help to standardize management and daily programmer work thereby enforcing a small, controllable development, release and testing cycle.
In other words, Agile and IDEF0 methodologies are based on Iteration-based development and allow continuous integration and test driven development.

In each of the iterations, a small piece of the system is delivered but more frequently in well-defined short bursts. In this way, overall project risk can be minimized and allows the project to adapt changes in systems requirements quickly. The developer might not be able to add enough functionality in each an iteration, to warrant a market release, but the goal is to have an available release (with less errors and more bug free) at the end of each iteration [45]. On the other hand, SSADM methodology uses linear framework development approach. Under SSADM, all system’s requirements are locked down at the start of the project, hence any changes in the decisions and implementations made from the previous stage can be hardly accommodated and more costly at a later stage.

Moreover, under Agile and IDEF0 methodologies, systems testing are conducted at different stages during the development life cycle. In the Waterfall approach, there is always a separate testing phase near the end of an implementation phase. However, in Agile and IDEF0, systems testing are usually done concurrently with coding, or at least, testing tasks start in the early days of iteration. Because the testing phase is done in each of the iterations, users can frequently use that small piece of the working system and validate the value. In this way, customer can, at least, make better decision about the system early in the development process. The flexibility and scalability to error check at any stage of the Agile and IDEF0 systems development life cycles makes them deliver the systems which are less erroneous and more bug free compared to SSADM, which tested the system at the end of the development module and under the capacity that may be difficult to correct.

D. Which Methodology favours the current system development trends?

In view of the above mentioned arguments and taking into consideration the extent of flexibility and adaptability of these methodologies, IDEF and Agile methodologies appear appropriate in the current competitive market of systems development. The backward scalability and flexibility of Agile and IDEF0 allow gathering of the system requirements in a more comprehensive way. Moreover, the delivery of systems to customers in an iterative fashion more likely will lead to a better customer satisfaction. On the other hand, despite of its popularity, the inflexibility and rigidity of SSADM, makes it inappropriate in the current competitive markets. However, the use of these methodologies in a real practice today is not positive [7]. While early studies reported a larger usage rates for the methodologies (reviews in [46, 47]), some recent studies such as [7] reported a smaller usage rates for the methodologies. It was further mentioned that most of the systems developers are not using the methodologies and they do not intend to adopt one.

III. SUMMARY AND CONCLUSION

The current development trends require system development methodologies that are: more flexible and scalable; quick to respond in requirement change and allow continuous development of the systems thereby enhancing a better customer satisfaction. SSADM, Agile and IDEF0 methodologies were presented and their underlying principles were analysed. While many methodologies have rectified their principles in order to suite the current organizational environment, SSADM methodology still remains obsolete. One should also be aware that SSADM was mostly made for British government institutions and derived from practice and concepts relevant to the old organizational environment. Therefore, there is a need to review its roles in order to fit into today’s competitive market. The interest in system integration and customization facilitates a move toward component based development approach.

In addition, changing nature of the business environment necessitate the short terms needs in the market [7]. Eventually, SSADM appears not appropriate with these kinds of changes and the current needs; and there agile and IDEF0 methodologies comes into the picture. Since IDEFO and Agile methodologies allow backward scalability and flexibility to make changes as per customer requirements, it is more inclined towards better customer satisfaction as compared to SSADM traditional methodology. As a concluding remark, in this ever-changing world of
technologies, it is really important to keep up-to-date methodologies and technologies.

ACKNOWLEDGMENT

The authors would like to thanks Ganesh Jaganathan for his valuable and useful comments on earlier draft of this paper. This work was supported by the National Natural Science Foundation of China under Grant No. 70971089 and Shanghai Leading Academic Discipline Project under Grant No. XTKX2012.

REFERENCES

[11] “The economy is flat so why are financials Cloud vendors growing at more than 90 percent per annum?”, FSN, 2013, accessed at <http://www.fsn.co.uk/channel_outsourcing/the_economy_is_flat_so_why_are_financials_cloud_vendors_growing_at_more_than_90_percent_per_annum#.VGD8IsnP8So> on 20th October, 2014.


Abdulla F. Ally received his Bachelor of Science in Computer Science in 2004 and Master of Science in Computer Science in 2008, both from University of Dar es Salaam, Tanzania. Abdulla is currently a Ph.D. candidate in Systems Analysis and Integration at the University of Shanghai for Science and Technology, China. He has participated in several research projects in computer science domain for more than 7 years. His research interests revolve around, but not limited to complex networks, modelling of networks and dynamic processes, graph theory and information systems

Ning Zhang received her B.E. degree in Wireless Communication from the Nanking Institute of the Posts and Telecommunications, China in 1982 and M.E. degree in Systems Engineering from East China University of Technology, China in 1991. From 1982 to 1984, she was an assistant engineer with the Shaanxi Wireless Communication Bureau, Xian, China. Currently, she is a Professor in the Business School at the University of Shanghai for Science and Technology (USST), China. Before joining the USST, she was an Assistant Professor at the Jiangsu Institute of Technology, China. Her research interests include complex network, network model, and graph theory.