DSS Development and Agile Methods: Towards a new Framework for Software Development Methodology

Natheer K. Garaibeh, *Member, IACSIT*

***Abstract*—DSS is a special type of IS and the theory of DSS have been evolved since the inception of the field and shaped into many diverse directions. As DSS has more specific nature (its core concept) than any other system we need a suitable Software Development Methodology that copes with the significant characteristics of such kinds of systems. We believe that Agile Methodology are the most suitable for DSS, despite this there were significant limitations for both Agile Methodologies and DSS Development Methodologies In this paper we will preview DSS Development Methodologies, compare between software development methodologies both agile and traditional according to their suitability for building DSS, and propose a new software development methodology.**

***Index Terms*—DSS, software development methodology, agile software development, knowledge management.**

I. Introduction

Decision Support Systems (DSS) are an important class of information systems (IS) that use data, models and knowledge to help managers solve semi structured and unstructured problems. Since the early 1970s, decision support technologies and applications [1] have evolved significantly. The advent of interactive computing coincided with that of DSS, the birth of personal computing and the web were contemporaneous with DSS's rapid growth, DSS pioneers came from [2] a wide variety of backgrounds and faced many challenges that they successfully overcame to demonstrate the value of using computers, information technologies and specific decision support software to enhance and in some situations improve decision making.

However, the term DSS and its offshoot, Executive Information Systems (EIS), have all but disappeared from vendors web sites and a new generation of systems have emerged, namely the Business Intelligence (BI) applications.[3]. This is the viewpoint of many researchers who involved in industrial application. For example Alter who argues that "decision support" provides a richer basis than "DSS" [4],and this is the viewpoint of many Practitioners,however in academic discipline the term of DSS serves as the basic concept of all the today's interest in building applications and technologies that support decisions, furthermore , Choosing an appropriate approach or methodology for building DSS [7] has been a popular and controversial topic in the Information Systems (IS) literature as a way of increasing DSS adoption and use.

Software engineering (SE) has long been argued to be at the core of IS. IS researchers perform software engineering research [6] to demonstrate the viability of new systems concepts, but existing research paradigms do not fully encompass the issues related to expanding the current capabilities of information systems. Furthermore SDM [29] have evolved from the classic waterfall model, to a spiral model, to prototyping and, recently, to agile development methodologies. The latter models seem to have worked well for many kinds of Information Systems.

The concept of agility, referring to development methodologies [25] that are more people oriented than process-oriented and emphasizing flexibility and adaptability over full description seems to positively affect the DSS development field as well as software development field. In other words, agile methodologies have more to do with the DSS development rather than just software development field. This combination between these two main ideas.

From this perspective, this paper argues that the process-based view that has underlined DSS development for so many years and agile development methodologies that are more people oriented than process-oriented could usefully give way to a knowledge- perspective since SDM is fundamentally [18] knowledge structuring activity.

A major goal of this ongoing research is to explore to what extent does agile methodologies can be applied to DSS development processes and whether they actually have advantages over the traditional approaches or they can just be used in the field of software engineering, The remainder of this paper is structured as follows: Section 2 focuses on the problem statement. Section 3 describes and discusses the literature review surrounding DSS development. Section IV discusses our preposition to solve the problem, and we conclude and present future work in Section V.

II. Problem Statement and Methodology

The inability of the DSS community [12] to come up with unified and standardized methods to develop DSS is a recurring topic that has kept researchers and practitioners busy for the past three decades. Interestingly enough, none of these approaches predominate and the various DSS development processes usually remain very distinct and project-specific.

A. Socio-Technical Factors

1) Human factors

In this paper, human factors cover the reasons why the people involved, users and decision-makers, subjectively

Manuscript received May 20, 2012; revised June 29, 2012.

Natheer K. Garaibeh is with Ajloun University College, Balqa Applied University, Jordan (e-mail: natheer_gharaibeh@bau.edu.jo; natheer_garaybih@yahoo.com)

oppose the computerized decision-making systems. This opposition is based mainly on the Communications gaps between users and developers, which are inevitable, Furthermore, building DSS is difficult [7] because people vary so much in terms of their personalities, knowledge and ability, preferences, the jobs they hold, and the decisions they need to make.

2) Conceptual factors

In this paper, conceptual factors cover the problems encountered by the development of a DSS which is often an undertaking of great complexity [7]. And the organizational environment of DSS [8] is subject to significant change and so even if the system requirements have been specified with some accuracy at the start of the project they are likely to change significantly over time as organizational structures and personnel change.

3) Technical factors

In this paper, technical factors cover the problems encountered by DSS related to purely software or hardware considerations., DSS [9] has more Specific nature than any other software - as shown in table-1 and is much more than just a DBMS, MBMS, GUI, interface, and knowledge component, DSS is complex system often composed of heterogeneous subsystems (various databases, complex mathematical libraries, proprietary data, etc.) and are therefore difficult to integrate in only one productive ,therefore we need for more analysis for the steps of DSS development.

B. Research Methodology

Due to the nature of difficulties mentioned, we will not claim to solve all of these problems; instead .we will try to find how to manage these factors in order to develop DSS successfully. SDMs are constantly evolving due to changing technologies and new demands from users, there are some SDMs suite DSS more than others; therefore, The essence of this research is to answer the following question:: How can we mix between Agile Development Methodologies (XP) and DSS development Methodologies?

We applied the following research methodology:

1) Investigated the current DSS development methodologies.
2) Investigated the SDMs and identify deficiencies and limitations for these methodologies according to DSS characteristics). We want to choose an approach that increases the chances of using DSS.
3) Propose a new SDM for development of DSS, based on 1 and 2.
4) Evaluate this methodology; by comparing it to the other methodologies, design case study and by conducting a scurvy.

III. Literature Review

A. Development Methodologies of DSS

Finding appropriate DSS development processes and methodologies [12] is a topic that has kept researchers in the decision support community busy for the past three decades at least. We will not review studies before 1995 because the DSS community has always shown great interest in the underlying technology and rapidly emerging Information Technology underpins DSS [26]. Studies on DSS development conducted before 1995 [16] [17] have identified more than thirty different approaches to the design and construction of DSS. Interestingly enough, none of these approaches predominate and the various DSS development processes usually remain very distinct and project-specific

In the DSS literature, experts prescribe a variety of approaches or methodologies for designing and developing DSS. Everyone does not however agree on what methodology works best for building different types of DSS. For example Gachet [11] who proposed a bipartite approach in which the software engineering part is separated from the knowledge engineering part. Another example is Turban [9] who described a development process consisting of 11 phases for DSS constructed by end users, also maracas [10] Arnot [8] and Zaraté [15] Many researchers preview and compare the Development methodologies of DSS [7][13][28] .

Power [7] mentioned three approaches for building DSS: systems development life cycle (SDLC) which is the most commonly encountered term used to describe the steps in a traditional systems development methodology, prototyping approach and end-user development of DSS. In both of the later two approaches a portion of the DSS is quickly constructed, then tested, improved, and expanded. Prototyping is similar to a related approach called rapid application development (RAD).

B. Discussion

Gachet [11] proposed a framework that solve this problem, whose cornerstone is the clear separation between the container of the DSS (responsible for the system part) and the contents of the IDSS (responsible for the knowledge base part).But it gives quite a bit of responsibility to the kernel component (The interface layer between the container and the contents), which has to be able to communicate with both the container and the contents of the IDSS.

The notion that a DSS evolves through an iterative process of systems design and use has been central to the theory of DSS since the inception of the field [23]. Terms such as ‘adaptive’ and ‘evolutionary’ capture the organic nature of the development of a DSS. It has been clear that the traditional approaches for analysis and design have proven inadequate because there is no single comprehensive theory of decision making, and because of the rapidity of change in the conditions which decision makers face.

The main problem of the previous DSS development methodologies is that they don’t handle the question of how to manage the human (knowledge management), conceptual and technical factors in order to develop DSS successfully; we discussed these factors in section 3. This means that we need to manage the steps of building DSS with respect to these three important factors, therefore we need SE methodologies.

From this perspective, there has been no research done to investigate the DSS development by the new methodologies of SE, such as Agile Development methodologies, which promise relevance to the DSS development according to its characteristics as we will show in the next section.

C. Comparison between Traditional Methodologies and Agile Methodologies

In this subsection we will compare between Traditional Methodologies and Agile Methodologies according to their suitability to DSS development and with respect to KM factors .These factors include: knowledge sharing and requirements volatility.

The main drawbacks of Traditional SE methodologies [24] are that it does not address issues of how well users internalize explicit knowledge and the sharing of tacit knowledge that is not externalized. Agile development approaches rely heavily on socialization through communication and collaboration to access and share tacit knowledge within the project team.

Traditional Methodologies work best when the requirements of the software project are completely locked in and frozen before the design and software development commences. However, in DSS projects which have an increasingly volatile business environment, firms are asking for lighter weight, faster and more agile methodologies that can accommodate the inevitable ongoing changes to requirements.

To solve this problem we need a software methodology that cope with knowledge management activities rates of unpredictable change in software projects. This is a major theme in the use of agile methods [19] [20].Agile methods emphasis on people, communities of practice, communication, and collaboration in facilitating the practice of sharing tacit knowledge at a team level. Agile processes use feedback, rather than planning as their primary control mechanism. The feedback is driven by regular tests and releases of the evolving software.

IV. The Proposed Solution

A. The Characteristics of DSS Development Methodology

1) Need for evolutionary

Developers of DSS need to adopt an evolutionary approach [8] because the systems they build generally address ill-structured decisions. It is extremely difficult a priori to specify the system requirements in such an nvironment. Constructing and using the initial versions of the system will help to clarify these requirements. The organizational environment of DSS is subject to significant change and so even if the system requirements have been specified with some accuracy at the start of the project they are likely to change significantly over time as organizational structures and personnel change.

2) Must be adaptive and iterative

The design of DSS must be adaptive ,the argument of this assumption is drawn from an analysis of the early DSS literature [23], and an analysis of the characteristics of many case studies described. DSS must evolve or grow to reach a 'final' design because no one can predict or anticipate in advance what is required. The system can never be final; it must change frequently to track changes in the problem, user, and environment because these factors are inherently volatile.

The more the customer is involved the more current requirements the product will satisfy. It can be easily seen that active user involvement [21] is a precondition for iterative-incremental development since the feedback from the end-user is necessary for its realization. And, both principles equally contribute to achievement of fitness for business purpose. This principle are often cited as typical features of other agile methods [22]

3) Need for cooperation

The proccess of DSS development must guaranree cooperation between the main stakholders, Zarate proposed that three main actors must be participated in the development process:end-users, experts and knowledge engineers [15].we add the developer as 4th actor. The aim is to help to design a system that the user will see as a real partner. It will therefore involve sharing intelligence about the situation by interacting and collaborating with partners rather than remaining a passive user. The knowledge engineer interacts directly with the expert and user to gain the domain knowledge through the development process. Then he interacts with the developer to build the final working DSS.

B. The Relevance of XP

Although XP is being widely used among mainstream software developers, its ideas have not been transferred into the DSS community yet. In this ongoing research, we will investigate reasons for adapting the values, principles, and practices of XP to DSS in order to synthesize a new methodology XPDSS

The three main Characteristics of the development processes of DSS extremely highlights the core principles and values of XP, The values of Extreme Programming [14] are communication, simplicity, feedback and courage, and these are further described through the principles rapid feedback, assumption of simplicity, incremental change, embracing change and quality work, which are well-suited for the needs of the DSS development process. But the main problem of XP when applying it to DSS is that it needs more exploration, this feature can be taken from any DSS framework in this paper we choose CAF-DMSS.

C. The Weaknese of XP

Although XP addresses the characteristics for DSS development methodologies, it is primarily weak on 4th and 5th chrematistics.

1) XP is poor in multi perspective view.

We have to modify XP to provide a wider perspective of views. XPDSS can increase the awareness of the scope by using a framework for DSS techniques.

2) There is no Big Design Up Front.

XP de-emphasizes up front design because it is claimed that everything is changing. Instead, a "metaphor" is used to describe the basic elements and relationships of the application [14]. However, for complex, DSS projects upfront architectural design is considered to be essential to solve this problem we need –before beginning XP.

3) XP is Extreme exploitation methodology.

As Stephens and Rosenberg [30] point out, in practice a number of XP practices are in fact anti-learning. Simple design and constant refactoring reduce the amount of reflection and thinking ahead. Pair programming may

discourage an individual from working through a problem if their partner knows how to solve it. Additionally, a customer representative based on-site creates a single point of contact for all the projects external knowledge needs. XP therefore represents an extreme knowledge exploitation strategy that relies on already skilled programmers and knowledgeable customer representatives to be successful. In the next section we will update the XP process by adding a new phase that represents the Exploration phase.

4) XP has limited scope of applications.

XP regards a software development project as a system of four control "variables": Cost, Time, Quality and Scope. A special problem arises when scope [14] is increased and time shall be fixed (schedule). Because you cannot control effectively with cost and quality, XP prevents this situation XPDSS can increase the awareness of the scope by using a framework for DSS techniques.

D.XPDSS

This review of theory and practice of DSS and producing a new methodology by applying DSS concepts may help the IS (Information systems) developers to have a clear mind of XP and its extended copy XPDSS, and use this methodology in the development of many kinds of applications.

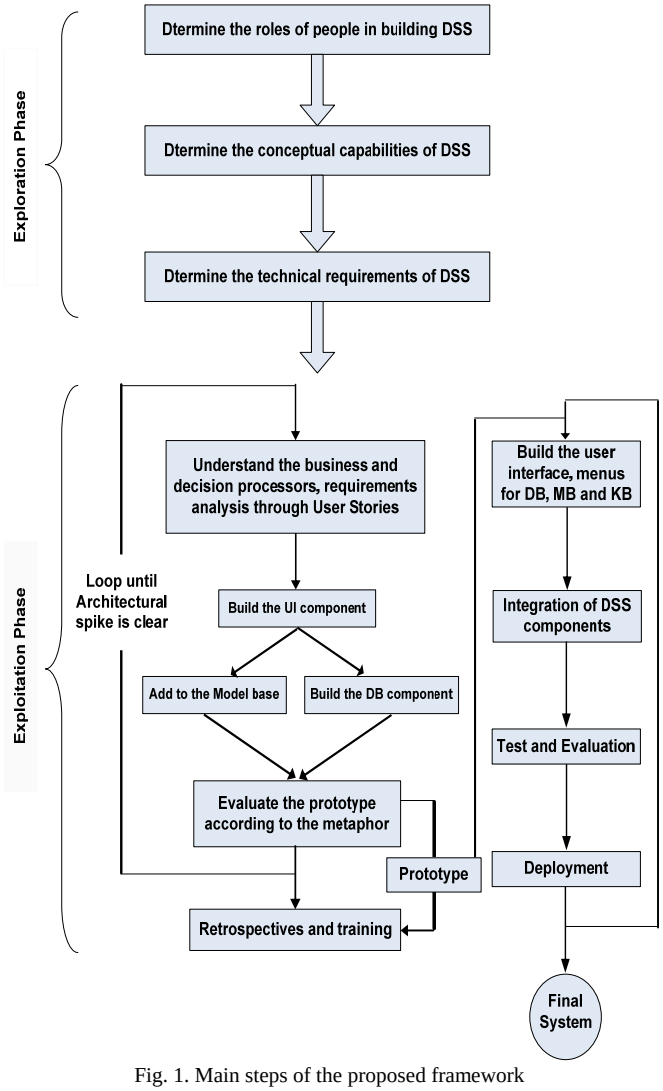

Fig. 1. Main steps of the proposed framework

XPDSS overcomes the limitation of both Agile Methodologies and DSS Development Methodologies. It consists of two main phases

- Explore phase: In determining the computation power and software requirements for any DSS software we need to a conceptual framework for DSS that can cope with the diverse techniques and capabilities of DSS. This can be achieved through the conceptual capability assessment framework for DSS (CAF-DMSS).By using this framework we can determine to which extent of the computation power the 3 main components of DSS.

- Exploit phase: After determining the nature and components of the initial spike for the given DSS, it will be easier to apply XP practices to produce the final application, Fig. 1 shows the main steps of the Proposed Framework

- Evaluation of XPDSS: There was always a sceptical Consideration about silver bullets [27] in software engineering literature, also Gachet [12] [13] wonders if there are one-size-fits-all solutions for DSS applications. Therefore it is not easy to say that the new methodology XPDSS is typically relevant to DSS. However, a XPDSS is currently under exploration and investigation.

V. Conclusions

The discussion of current development approaches of DSS illustrates the importance of the concept of evolution for DSS development. Terms such as adaptive and evolutionary capture the organic nature of the development of a DSS. However there has been no research done to investigate the DSS development by the new methodologies of SE, such as Agile Development methodologies. The good news for both the agile community and DSS community is that the seeds of XP have a huge potential to thrive and prosper in the DSS development. XPDSS which has been emerged from combining XP and DSS is a possible seed of XP.

We have described the initial step of an ongoing research effort towards establishing a SDM for building DSS, the next phases of this research are in progress, it is foreseen that integration between Agile and DSS Development Methodologies can be broadened into a more common model (XPDSS).

The intent of the proposed framework is not to serve as a "silver bullet" or panacea to all DSS development problems; instead, it provides a systematic way to develop a DSS by making the best use of well-organized XP practices and several mechanisms from different disciplines, such as DSS theory. By combining XP and DSS we increase the opportunities of applying a rich field of problem solving concepts such as DSS, with effective practices of XP.

References

[1] J. P. Shim, M. Warkentin, J. F. Courtney, D. J. Power, R. Sharda and C. Carlsson, "Past, Present and Future of Decision Support Technology," *Decision Support Systems*, 33, pp 111-126 , 2002.

[2] D. J. Power, "A Brief History of Decision Support Systems". DSSResources.COM, World Wide Web, [Online]. Available: http://DSSResources.COM/history/dsshistory.html, version 4.0, March 10, 2007.

[3] C. Carlsson E. Turban, "DSS: directions for the next decade", *Decision*

Support Systems, Volume 33, Issue 2, 2002, Pages 105-110 .
[4] S. Alter, "A Work System View of DSS in its Fourth Decade", Decision Support Systems, 38(3), 2004.
[5] Pressman, R.S, *Software Engineering: A Practitioner's Perspective*, 5th edition, McGraw-Hill, New York, 2000.
[6] D. G. Gregg, U. R. Kulkarni and A. S. Vinze , "Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems". *Information Systems Frontiers 3*(2): pp 169-183 (2001).
[7] D J. Power, *Decision Support Systems Hyperbook*. Cedar Falls, IA: DSSResources.COM, HTML version, 2000, accessed on (19/8/2008) at URL http://dssresources.com/subscriber/password/dssbook
[8] D. Arnott," Decision Support Systems Evolution: Framework, Case Study, and Research Agenda" (Working Paper. No. 2002/07). Melbourne, Australia: Decision Support Systems Laboratory, Monash University, 2002.
[9] Turban, Aronson, and Liang, *Decision Support Systems and Intelligent Systems*, Seventh Edition, Prentice Hall, 2005.
[10] G. M. Maracas, *Decision support systems in the 21st century*. Upper Saddle River, NJ, Prentice Hall, 2003.
[11] A. Gachet, "Software frameworks for developing decision support systems: a new component in the classification of DSS development tools." *Journal of Decision Systems* 12(3/4): 271-281. , 2003.
[12] A. Gachet and R. Sprague , " A Context-Based Approach to the Development of Decision Support Systems", *Proceedings of the Workshop on Context Modeling and Decision Support-Paris*, 2005.
[13] A. Gachet and P. Haettenschwiler, "Development Processes of Intelligent Decision Making Support Systems": *In Intelligent Decision-Making Support Systems (i-DMSS): Foundations, Applications and Challenges, Springer* pp 97-121, 2006.
[14] K. Beck, Extreme Programming Explained: Embrace Change. Longman Higher Education 2000.
[15] P. Zarate and C. Rosenthal-Sabroux, "a cooperative approach for intelligent decision support systems" in *Proceedings of the Thirty-First Hawaii International Conference (HICSS.1998)* Volume: 5, On page(s): 72-81 vol.5 , 1998.
[16] B Arinze, "A Contingency Model of DSS Development Methodology." *Journal of Management Information Systems* 8(1). pp 149-166, 1991
[17] K. B. C. Saxena, "Decision support engineering: a DSS development methodology". *24th Annual Hawaii International Conference on System Sciences (HICSS'91)*, Los Alamitos, CA, IEEE Computer Society Press. 1991
[18] B. Crawford, C. Castro and E. Monfroy, " Knowledge Management in Different Software Development Approaches", in Advances in Information Systems, Volume 4243 ,pp 304-313 Springer Berlin / Heidelberg , 2006.
[19] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New Directions on Agile Methods: A Comparative Analysis", *in Proceedings of the 25th International Conference on Software Engineering ICSE'03: IEEE Press*, 2003.
[20] D. Cohen, M. Lindvall, and P. Costa,"An Introduction to Agile Methods", vol. 62. ,*Elsevier*, Amsterdam , 2004
[21] M. N. Aydin, F. Harmsen, K. Slooten., and R. A. Stagwee , "An Agile Information Systems Development Method in use". *Turk J Elec Engin*, 12(2), 127-138, 2004.
[22] K. Beck. et al., 'Manifesto for Agile Software Development', The Agile Alliance, February 2001, [Online]. Available: http://www.agilealliance.org/
[23] P. G. W.Keen and M. S. Scott Morton, *Decision support systems: an organ.izational perspective*. Reading, Mass., Addison-Wesley Pub. Co. 1978
[24] T. Chau, F. Maurer and G. Melnik, "Knowledge Sharing: Agile Methods vs. Tayloristic Methods", *Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 302, 2003.
[25] M. Fowler and J. Highsmith. 2001. The Agile Manifesto. Software Development Magazine. August.
[26] Sdmagazine, [Online]. Available: http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm
[27] F. P. Brooks "No silver bullet. Essence and accidents of software engineering". *IEEE Computer*, 20(1), 1987.
[28] Andrew Blair, John Debenham, Jenny Edwards, "A comparative study of methodologies for designing IDSSs", *European Journal of Operational Research* 103,277-295, 1997.
[29] B. W. Boehm: "A view of 20th and 21st century software engineering." *ICSE 2006*: 12-29 , (2006)
[30] M. Stephens and D. Rosenberg, "Extreme Programming Refactored: The Case Against XP", *Apress* , 2003.

Natheer K. Gharaibeh received his B.S. degree (Computer Science) in 1999 from Yarmouk University – Irbed - Jordan. In 2002, he earned his Master Degree in Computer Information Systems from UBFS (University of Banking and Financial Science) - Amman. A Ph.D. was received in 2009 in Computer Information System from UBFS – Amman.

He is employed as Assistant Professor at Ajloun University College, Balqa Applied University in Jordan from Oct. 2009 till now. Before that he worked as full time Lecturer in Balqa Applied University. He also work as part-time Lecturer at Jordan University of Science and Technology (JUST) and other Jordanian universities. He published more than ten papers in International Conferences and Journals. His current research interests are: Decision Support systems, Data warehouse, Data mining, ontology and Software Engineering.