The Role of the Collaborative Integrated Model (MDSIC) in Agile Software Development - Case Study and Practical Advice

PhD José Luis Cendejas Valdéz
MGTI Gustavo Abraham Vanegas Contreras
Research; Cuerpo Académico Transferencia Tecnológica para la Construcción de Software, Universidad Tecnológica de Morelia, Vicepresidente Pino Suarez No. 750 Cd. Industrial, Morelia, Michoacán 58200/ México
PhD Heberto Ferreira Medina
Research; Instituto de Investigaciones en Ecosistemas y Sustentabilidad, Universidad Nacional Autónoma de México, antigua carretera a Pátzcuaro No.8701 Col. Ex Hacienda de San José de la Huerta, Morelia, Michoacán 58190/ México.
MTI Alfonso Hiram Ginori González
Research; Instituto de Radioastronomía y Astrofísica, Universidad Nacional Autónoma de México, Antigua Carretera a Pátzcuaro No.8701 Col. Ex Hacienda de San José de la Huerta, Morelia, Michoacán 58190/ México.

Doi: 10.19044/esj.2018.c3p1 URL:http://dx.doi.org/10.19044/esj.2018.c3p1

Abstract

This research aims to determine the importance of the generation and application of models in the software development area, performing a comparison of existing models and their applicability. One of these models is the collaborative integrated software development model (MDSIC). There are several methodologies and models that help in software development, but most of them have processes in place that make the development more complex instead of agile. The MDSIC proposes five levels of better practices that should be followed in software development projects. Also, the model supports the main areas of knowledge proposed by the Project Management Institute (PMI), thereby generating software in line with the objectives of the organizations. The MDSIC is supported by an internet platform (MDSIC v1.0), which has been developed using Java Server Pages technology and responsive web design. This platform has generated a knowledge base using social business, thus generating an information bank helpful in obtaining the experiences of specialists, proposing best practices in building agile software

projects. Therefore, this article also aims to show matching indicators and results of implementating MDSIC in software projects, evaluating the needed parameters to generate good quality software, and thus align technology with the goals of the organizations.

Keywords: Software development; MDSIC; Agile methodologies; Quality assurance

Introduction

Development of software made to measure in Mexico represents high costs for organizations and many of these projects do not meet the minimum requirements to software factories have to improve their processes. Organizations with different business line looking for the fluidity of information processes with the help of software development companies, called software factories. Those factories systematize and improve the processes of organizations.

According to (Piattini and Garzás, 2010) the term "software factories" conceptualizes an organization wich main objective is to produce quality software, implying a specific way of organizing work, with a considerable specialization, as well as processes formalization and standardization. For optimal software development several fundamental elements must converge to obtain a custom made product that provides proper process functioning in organizations.

Among fundamental elements are: 1) hardware; 2) software; 3) qualified personnel (technically as well as working with processes); 4) project administration; 5) agile models for software construction. The purpose of these elements is to expedite, ease and fulfill different projects of software development towards covering organizations' objectives. Therefore, in this paper is presented MDSIC as part of the industry in Center-West Mexico's region. MDSIC helps to achieve a product based on norms, quality assessment based on indicators and cover needs of enterprises with line of business in software development.

I.

A. Literature review

This section offers a literature review on use of different models for software development, also experiences generated using the software development model implementation (MDSIC) in Mexico. Nowadays there is a need to create software based on models that give certainty to enterprises having quality products and allowing a direct impact on their objectives. With the goal that models will aid the enterprises developing software, not otherwise, enterprises end up working for the models.

At this time software has unique challenges, such as: a) Form factors, b) User's technology, c) Usability, d) Design/user interaction, e) Programmers' choice for mobile devices implementation, f) Development processes issues, g) Programming tools, h) User interface design, i) Applications portability, j) Quality and k) Security. Additionally, look for development process time reduction.

One of the best ways to fight complexity in software development is with abstraction decomposition and problem break out. This leads to use of models that allow all the elements mentioned to interact. Business process modeling role in informatics systems (software) construction has a great importance due to these systems grow in scale and complexity, Barjis (2008). An example of business process modeling is based in theoretical concepts of the DEMO methodology, which is built upon graphical notations using Petri Networks. Both, DEMO concept and Petri Networks have been studied broadly in different research lines. DEMO methodology was developed and implemented in several real life projects, Dietz (2006). Therefore, models can be found in all areas such as software engineering.

In (Greenfield and Short 2003), it was concluded that: "The software industry remains reliant on the craftsmanship of skilled individuals engaged in labor intensive manual tasks. However, growing pressure to reduce cost and time to market, and to improve software quality, may catalyze a transition to more automated methods". In Cendejas et al (2014), is mentioned that for the last three decades software development has been immerged in a problematic from which has been difficult to get over. The main issue on this matter is, to develop quality products that satisfies organizations' needs and objectives.

In addition, the software is not aligned with the goals and objectives of the organization. Software is built by IT experts who are dedicated to analysis, design and development, but are never accompanied by experts in the organizational processes that benefit product development in a formal way. There is a need to analyze how to improve the software industry, and describe the best technologies that can be used to support this view. "Therefore it is suggested that the current software development paradigm, based on object orientation, may have reached the point of exhaustion, and models are proposed for its successor". In the last decade, this has progressed compared to what Booch (2002), one of the creators of UML estimated in 2002.

According to Booch (2002), in that year only 5% of developers used UML in its projects and the majority used it for documentation. In several studies, (Piattini and Garzás, 2010), concluded that: "The model-driven software development (MDSD) was founded with the objective of integrating models and code as participants in software production process. The development of any system software needs to be addressed with two different perspectives: a) the perspective that addresses issues related to the application

domain (the problem domain) and b) the perspective that addresses aspects of software technology used to implement the system (the solution domain). The problem domain usually has nothing to do with the software technology. For the end-user, software is a mere tool that should not cause concerns".

(Quintero and Anaya, 2007), discusses the role of models as fundamental in software development to enhance elements of software reuse and facilitate the work of the different roles involved in the process. In many cases the use of models and methodologies for software development requires time, effort and investment, and if the staff is not trained delays may occur in the delivery of software projects. Here is where the models help to solve real projects and provide flexible solutions to the needs of organizations through software development.

There are different models and methodologies that function as support tools for software development. In a recent study about models and methodologies Somerville (2005), conceptualize the following:

- Software development model: is a simplified representation of software development process, presented from a specific perspective.
- Software development methodology: Is a structured approach for software development including system models, notations, rules, designs suggestions and process guidance.

Another way of making software is through agile methodologies, allowing to carrying out a more effective and faster tracking scheme. Harleen et al., (2014); say that agile methodologies follow an iterative approach to build software quickly, where the entire software development life cycle is divided into smaller iterations, which helps minimize overall risk. Agile software development approach refers to the iterative and incremental strategy involving self-organizing teams and functional teams that work together to create software. Some of the existing agile methods are: Crystal Methodologies, Dynamic Software Development Method (DSDM), Lean Software Development, Scrum and Extreme Programming (XP). Table 1 describes each according to their references.

Table 1. Description of leading methods for agile development

Method agil	Description	Reference
Crystal Methodologies	A family of methods for co-located teams of different sizes and criticality: Clear, Yellow, Orange, Red, Blue. The most agile method, Crystal Clear, focuses on communication in small teams developing software that is not life-critical. Clear development has seven characteristics: frequent delivery, reflective improvement, osmotic communication, personal safety, focus, easy access to expert users, and requirements for the technical environment.	Cockburn (2000, 2004).

Dynamic software development method (DSDM)	Divides projects in three phases: pre-project, project life-cycle, and post project. Nine principles underlie DSDM: user involvement, empowering the project team, frequent delivery, addressing current business needs, iterative and incremental development, allow for reversing changes, high-level scope being fixed before project starts, testing throughout the lifecycle, and efficient and effective communication.	Stapleton (2003).
Lean software development	An adaptation of principles from lean production and, in particular, the Toyota production system to software development. Consists of seven principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity, and see the whole.	Poppendieck & Poppendieck (2003).
Scrum	Focuses on project management in situations where it is difficult to plan ahead, with mechanisms for ‘‘empirical process control”; where feedback loops constitute the core element. Software is developed by a self-organizing team in increments (called ‘‘sprints”), starting with planning and ending with a review. Features to be implemented in the system are registered in a backlog. Then, the product owner decides which backlog items should be developed in the following sprint. Team members coordinate their work in a daily stand-up meeting. One team member, the scrum master, is in charge of solving problems that stop the team from working effectively.	Schwaber & Beedle (2001).
Extreme Programming (XP)	Focuses on best practice for development. Consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-h week, on-site customers, and coding standards. The revised ‘‘XP2” consists of the following ‘‘primary practices”: sit together, whole team, informative workspace, energized work, pair programming, stories, weekly cycle, quarterly cycle, slack, 10-minute build, continuous integration, test-first programming, and incremental design. There are also 11 ‘‘corollary practices”.	Beck (2000, 2004)

According to Sutherland et al., (2008); the XP methodology receives more bibliographical attention because it applies conceptual premises to solve a problem that is slightly different from the evolutionary development of applications. (Schuwaber and Sutherland 2011), comment that organizations are focusing their attention to the agile methodology named Scrum. Scrum is used for managing software development, whose main objective is to maximize the return on investment for the company and generate innovation.

Harleen et al., (2014); propose that the agile development promotes stakeholder involvement in projects where those stakeholders enable monitoring of the activities, which increases productivity and profit. Agile development encourages users to participate actively in the entire product development. Pressman (2006), found that: "Modern computer software is characterized by continuous change, very short delivery times and an intense need to satisfy customers/users. In many cases, the time-to-market is the most

important management requirement. If this requirement is lost, the software project itself may lose its meaning."

In recent years the technology acceptance has been investigated by the theory of diffusion of innovations and models of social psychology Bhattacherjee (2000). The main focus of the theory of diffusion of innovations and for the adoption of an innovation is communication Rogers (2003). Often the diffusion of innovation within a population can occur from a very small proportion, which can be modeled mathematically for selection Bohlmann et al, (2010). The diffusion of an innovation can be a "special kind of communication." According to Rogers (2003), it comes from word of mouth and the existence of adopters will depend on the influence of early users.

Kiron et al, (2012); proposed in his research at the Massachusetts Institute of Technology (MIT), published in MIT Sloan Management Review (MIT SMR) and Deloitte in the spring of 2012, that "social business is an activity that uses social media, social software and social networks to enable more efficient and effective mutual connections between people, information and resources. These connections can facilitate business decisions, actions and outcomes in different areas of the companies" Yunus et al. (2010); report that in the coming years there will be a growing interest in building business models based on social participation, because humans have an instinctive natural desire to improve the lives of their fellowmen when possible.

A real innovative option is the collaborative integrated software development model (MDSIC). Cendejas et al. (2005); mention that "the collaborative integrated software development model (MDSIC) offers experts an easy way to interact with it through five levels that provide best practices for software development; these levels also consider the basic functions proposed by the Project Management Institute (PMI), which allows generating quality software aligned with organizational goals.

MDSIC allows evaluating software quality using a series of indicators that must be considered for optimum performance of a given software. These indicators are supported by quality standards. A key part of MDSIC is the creation of a knowledge base that feeds through social business, which is generated using social networks (Facebook, Twitter, StumbleUpon, Pinterest etc.), thereby producing a data bank with opinions of experts in software development. Cendejas et al. (2005); propose the use of MDSIC through a series of steps that facilitate agile project management and software development.

This model consists of five levels: 1) Level 0: Problem detection; 2) Level 1: Analysis and design; 3) Level 2: Development; 4) Level 3: Implementation; 5) Level 4: Quality indicators. MDSIC also contemplates the five basic functions covered under the Project Management Institute (PMI), which are: 1) Integration of project management; 2) Scope; 3) Time; 4) Cost;

5) Quality. Figure 1 presents the general structure proposed by the MDSIC including its elements.

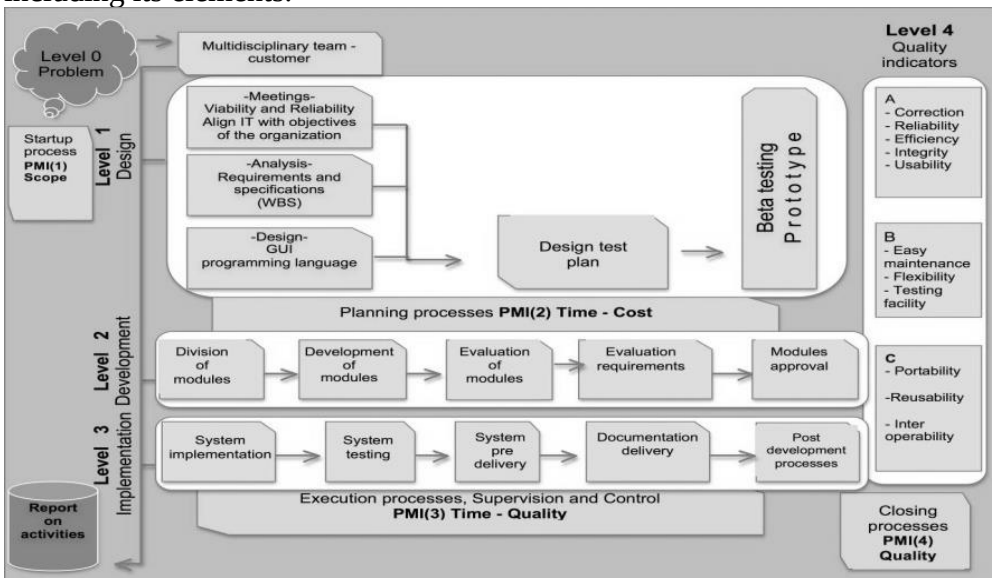

Figure 1. Collaborative Integrated Software Development Model (MDSIC)

B. Methodology

MDSIC has been the basis for software development in several companies in Mexico and has served as a medium for monitoring and providing continuity in several of those projects. It has become a tool that has contributed to achieving the objectives in each project and thus helps enterprises, which act as clients of software developments, to be more competitive. The methodology of this research was to implement MDSIC in different projects and use its indicators to measure the quality of the software produced. Having identified the problem, the research objectives were established and the nature of the investigation, which defines procedures to obtain the information needed to solve the problem, is described.

A cross-sectional study was conducted with the following nature of research: quantitative, field, quasi-experimental and explanatory, Kothari (2004). This generated a synthesis analysis of different models and methodologies for agile software development, besides obtaining coincident indicators. Figure 2 shows the process followed for carrying out this research.

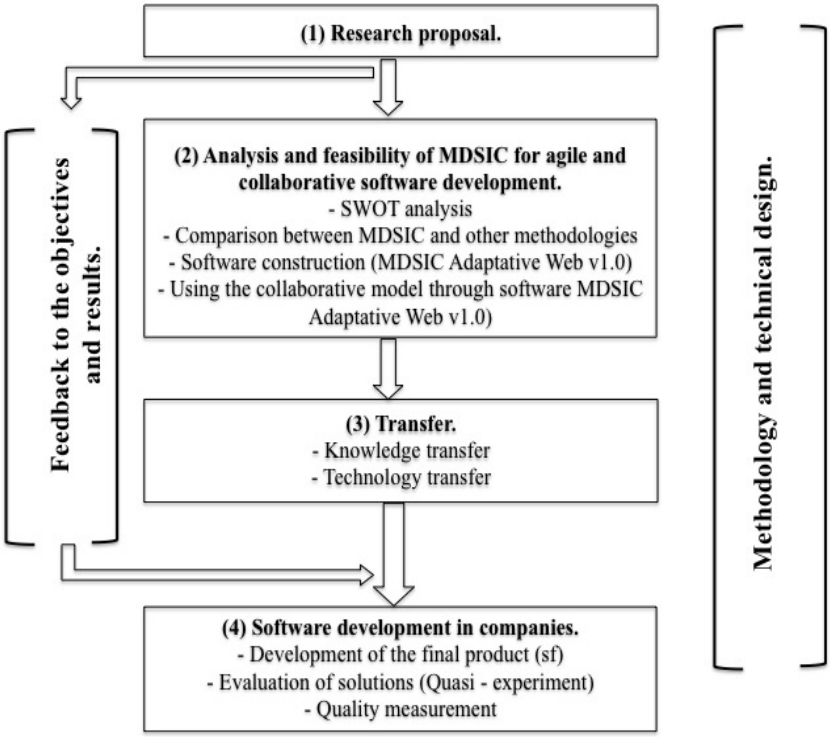

Figure 2. Description of the methodological process

Using this prior study, an analysis was conducted to determine the feasibility of applying the MDSIC model in software development projects based on the strengths, weaknesses, opportunities and threats (SWOT) analysis compared to other methodologies. According to Bockle et al (2004), "The SWOT analysis allows an assessment of the strengths and weaknesses factors that together diagnose the internal situation of an organization and also its external evaluation; that is, opportunities and threats".

With help of the SWOT analysis, behavior of MDSIC compared with eight of the most commonly used methodologies was identified. This comparison was based on the areas of: 1) stages considered; 2) projects size; 3) quality assessment and 4) application of social business. Figure 3 shows the comparison of stages considered by each one of the models and/or methodologies, in addition to size of the projects.

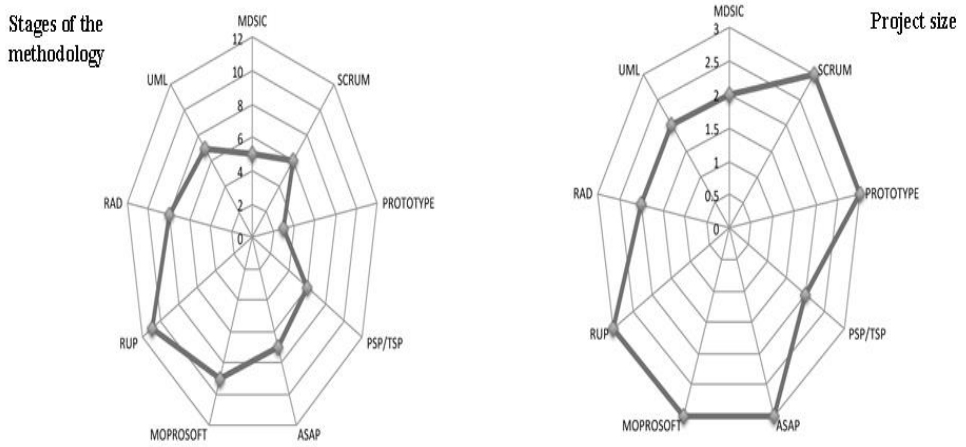

Figure 3. Stages and size of projects

Quality in software development can be measured in two ways: 1) by the degree of precision with which each product (software) conforms to the needs of every customer and 2) through the ratio of defects or product errors. Figure 4 shows the comparative evaluation of MDSIC referring to the use of indicators that assess the quality of software and the implementation of social business as a key element of the agile software development.

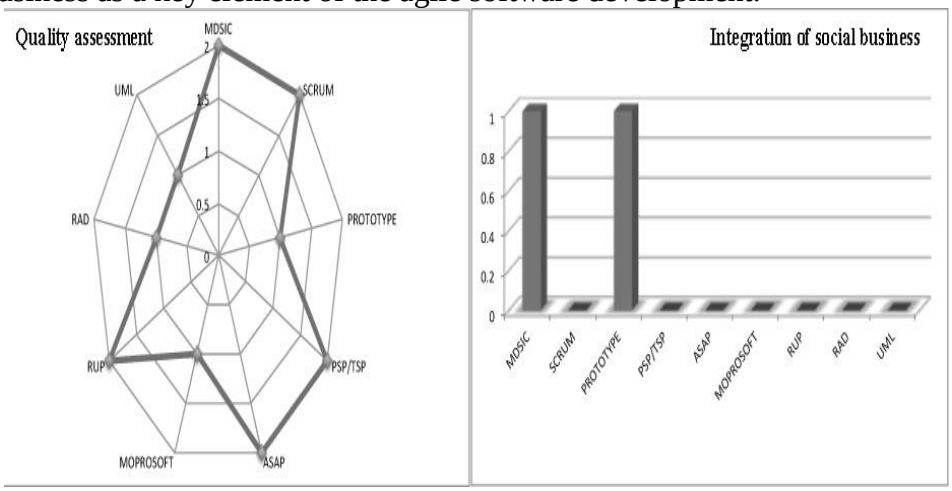

Figure 4. Quality and integration of social business

A survey was conceived and was answered by 52 software development companies from different states of the central-western area of Mexico. The survey had 29 questions that were designed by the Likert scale, where a reliability analysis was performed using Cronbach's Alpha study with help of statistical software for social sciences (IBM's SPSS), obtaining a value of reliability acceptance 0.812. This shows a good consistency in the responses obtained.

Based on this test it was identified that the instrument (survey) designed and implemented is valid and reveals different coincident indicators that should be considered to improve the quality in agile software development. The following process was to conduct a study of *Pearson* correlations of the main direct and indirect variables of the study to determine their affinity Bockle et al (2004). Table 2 shows correlations between variables with the greatest impact (> = 0.6).

Table 2. Correlations between variables with the greatest impact

Pearson's correlation > = 0.6	1	2	3	4	5	6
1. - Processes defined at work.	100	.83				
2. - Work includes project management – PMI.		100	.83			.68
3. - Development staff works under processes.			100			.68
4. - Organization with more than 15 years of experience.				100	.77	
5. - Organization with more than 16 developers.					100	
6. - The use of models and methodologies enhance the efficiency of results.						100

C. Results

An essential part of MDSIC is the "**activity report**", which has a presence through a system that is implemented based on a technology known as "responsive web", which is a way of programming that allows the system to adapt to the size and shape of any device that connects to it. The software accompanying MDSIC aims to capture and store the information generated from software projects. In addition to creating a knowledge base enhanced by expert developers looking to propose improvements in the processes of software development. This allows collaborative work from its multiuser nature as shown in Figure 5.

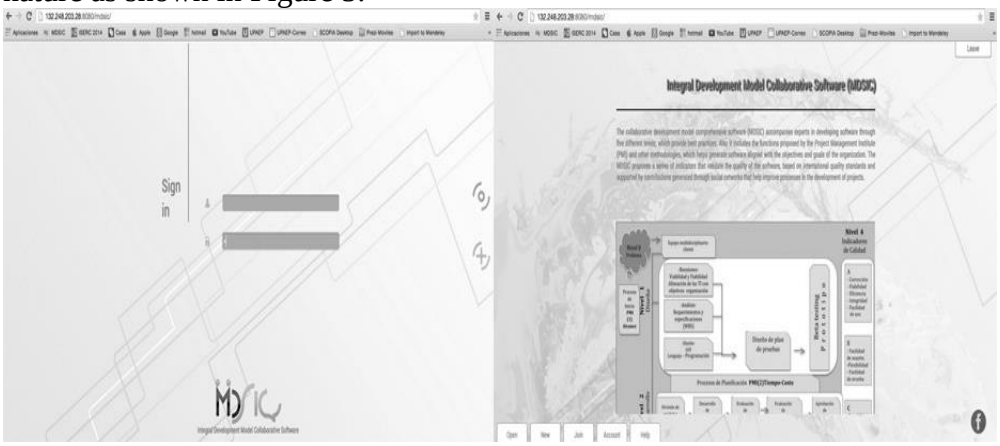

Figure 5. MDSIC v1.0 screens for user's registration, validation and welcome

In MDSIC v1.0 there are different levels and roles, where users can participate as: 1) project manager; 2) customer; 3) analyst; 4) designer; 5) developer and 6) QA (quality assurance). The role of project manager is the highest level since it is responsible for creating, managing and monitoring the entire project from level 0 to level 4 as proposed by MDSIC. In addition, it is responsible for capturing information from the memorandums of the meetings at every level, as shown in Figure 6. The creation of the other roles depends on the needs of each project.

Figure 6. MDSIC v1.0 software screen for user's participation

The projects developed through MDSIC v1.0 have the facility to measure the progress of these projects through the quality module, which allows to measure the progress of each of the levels. Thus the project manager, quality assurance (QA) and the collaborative team can measure the progress of each project graphically according to plan, as shown in Figure 7.

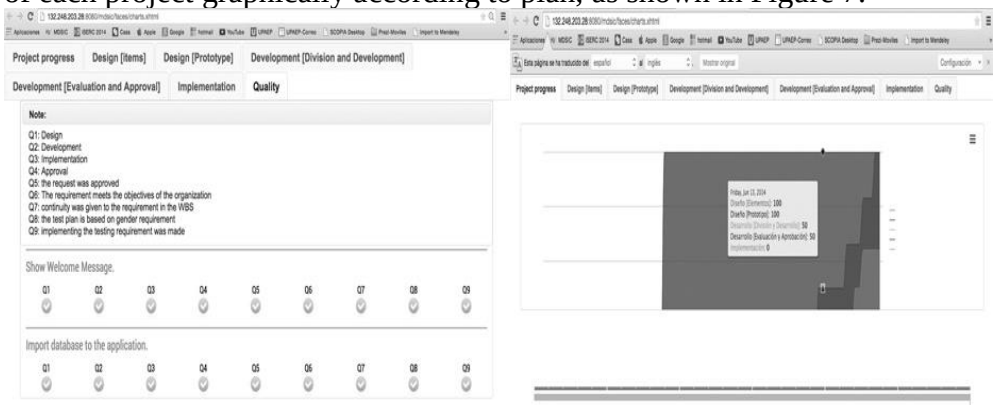

Figure 7. Graphic displays of project progress in the MDSIC v1.0 software

Table 3 shows the coincident indicators in software development projects using the MDSIC model.

Table 3. Coincident indicators in projects based on MDSIC

Project	Multidisciplinary team	WBS Planning	Prototype	Development modules	Post development	Quality KPI´s	Core f. PMI
LBS CIEco UNAM - Morelia.	✓	✓	✓	✓	✓	✓	✓
Species catalogue CIEco UNAM - Morelia.	✓	✓	✓	✓	✓	x	✓
Ixtapa is the destination	✓	x	x	✓	✓	x	x
JonaBachi	x	✓	✓	✓	✓	x	x
Development MDSIC v1.0	✓	✓	✓	✓	✓	✓	✓

To measure the impact of the application of MDSIC, a quasi-experimental study was conducted, to compare the development of software before and after the use of MDSIC in two projects. For results of the study, a questionnaire was applied. The questions were designed based on the "Likert" scale, where the lowest value is 1 and represents the answer "strongly disagree" and the highest value is 5 and the answer is "strongly agree". The results of the questions can be seen in Table 4.

Table 4. Results of quasi-experimental study

Question	**Software factory A**		**Software factory B**	
When you are developing software:	Before MDSIC	With MDSIC	Before MDSIC	With MDSIC
1.- Use a methodology	5	5	1	5
2.- Form a multidisciplinary experts team	2	4	2	5
3.- Generate viability and feasibility analyses before the project	1	5	1	5
4.- Know the goals and objectives of the organization requesting the project	4	5	3	5
5.- Generate a document for project management (Gantt, WBS)	2	5	4	5
6.- Generate a prototype of the software to develop for customer approval	2	5	1	5
7.- Generate the development stage through modules	3	3	3	5
8.- Take into account the approval of modules by the client	1	4	1	4
9.- Consider implementing a testing stage before definitive implementation	1	5	3	5
10.- Write development documentation	3	5	1	4
11.- Generate quality measures through indicators	1	4	1	5
Totals	**25**	**50**	**21**	**53**

The results of the questions made to expert developers who used MDSIC in companies "A" and "B" are shown in Figure 8, where it can be seen the behavior of the items listed before and after using MDSIC.

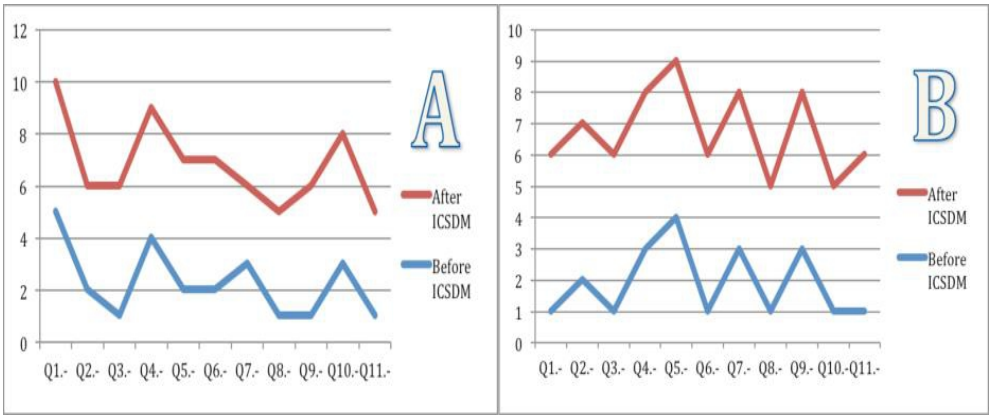

Figure 8. Project development before and after using MDSIC

Using MDSIC can significantly increase the productivity of software engineers, which results in reducing the effort and cost required to develop and implement software. According to Piattini & Garzás (2010), the turning point on a software development project can be around three points; impact, cost and benefit. Therefore, the evaluation of companies "A" and "B" was conducted with reference to those parameters.

Conclusion

The problems identified in the field of software development in the last three decades is mainly due to not having well defined methods for building software; this can be offset by using the model MDSIC; it has proven to be a tool that helps software development companies to develop projects that line up with the goals and objectives of organizations, thus contributing to their productivity. MDSIC aims to integrate all involved by forming teams of collaborative work that allow significant progress in building the software.

The need for documenting software projects is very important and MDSIC, with its system MDSIC v1.0, enables to register and document all the processes of software development. This application has multi-user features and was designed to function as a responsive technology; MDSIC v1.0 automatically adjusts to any device. MDSIC v1.0 can be used at the following address: http://132.248.203.28:8080/mdsic/.

The research results show the advantages of applying the MDSIC on agile development, by evaluating groups of software development companies. This research is relevant to the agile software development as it provides a better understanding and organization on this issue, in order to improve investment in resources, efforts and agile principles. It is concluded that the application of MDSIC improves process control and the quality of software is measured by the proposed indicators.

The experiences of software developers who have used MDSIC improved a knowledge base through the use of social networks and social business. This knowledge base stores best practices and experiences using MDSIC v1.0, and has improved the building of software development projects.

This work contributes with relevant information to research focused on software engineering and process modeling, in addition to professionals in the use of agile methodologies, allowing the identification and best practices to achieve success in agile software development. In the area of statistics, this study confirms that research in software engineering can be certified and validated by the multivariate analysis. Furthermore, the work contributes a quantitative research that encourages organizations to use agile principles in software development.

At present version 2.0 of the MDSIC software is under development, aiming at the development for mobile devices, with adherence to the agile development of custom software. Consequently, it is advisable to software industry professionals to use this article as a map of issues related to the topic, as they can benefit from the analysis in order to better understand trends in agile software development. It is expected that the proposals made in this document provide guidance for future research.

References:

1. Piattini V. M. G., & Garzás P. J., 2010. *Fábricas de software, experiencias, tecnologías y organización.* (2da. Edición) Madrid, España. Editorial Ra – Ma.
2. Barjis, J. (2008). *The importance of business process modeling in software systems design. Science of Computer Programming, 71*(1), 73–87. doi:10.1016/j.scico.2008.01.002
3. Dietz, J.L.G.(2006), *Enterprise Ontology - Theory and Methodology*, Springer, New York.
4. Greenfield, J., Short, K., Cook, S. y Kent, S. (2003). *Software factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley.
5. Cendejas Valdéz, J. L., Vega Lebrún, C. A., Careta Isordia, A., & Ferreyra Medina, H. (2014). *Design of the integrated collaborative model for agile development software in the central-western companies in Mexico*. *Nova Scientia , 7* (13), 133-148.
6. Booch, G. (2002). *Growing the UML*. *Software and Systems Modeling, 1*(2), 157-160.
7. Quintero, J. B., & Anaya, R. (2007). *MDA y el papel de los modelos en el proceso de desarrollo de software*. *Redalyc*, (8), 131–146. Escuela de ingeniería de Antioquia.

8. Someerville, I. (2005). *Ingeniería del software* (septima ed.). (A. B. María Isabel Alfonso Galipienso, Trad.) Madrid, Madrid, España: pp:130-145 Pearson.
9. Harleen K. Flora, Wang X., Swati V Chande. "*An Investigation into Mobile Application Development Process, Challenges and Best Practices*". *International Journal of Modern Education and Computer Science (IJMECS)*. 2014.
10. Cokburn A., Selecting a project's methodology, IEEE Software 17 (4) (2000) 64–71.
11. Cockburn A., Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley, 2004, ISBN 0-201-69947-8.
12. Stapleton, J. (Ed.). (2003). *DSDM: Business focused development*. Pearson Education.
13. Poppendieck M., Poppendieck T., Lean Software Development – An Agile Toolkit for Software Development Managers, Addison-Wesley, Boston, 2003, ISBN 0-321-15078-3.
14. Schwaber K., Beedle M., Agile Software Development with Scrum, Prentice Hall, Upper Saddle River, 2001.
15. K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000, ISBN 0-201-61641-6.
16. K. Beck, Extreme Programming Explained: Embrace Chage, second ed., Addison-Wesley, 2004, ISBN 978-0321278654.
17. Sutherland, J., Jakobsen, C. & Johnson, K., 2008. *Scrum and CMMI level 5 The magic potion for code warriors.* s.l., IEEE.
18. Schwaber, K. & Sutherland, J., 2011. The Scrum Guide: *The Definitive Guide to Scrum, The rules of the game*, s.l.: Scrum.org
19. Pressman, R. *Ingeniería de software. Un enfoque práctico.* España,McGraw.Hill, 2006. 61 p.
20. Bhattacherjee, A., 2000. *Acceptance of e-commerce services: the case of electronic brokerages*. IEEE Trans. Syst., Man Cybern., Part A: Syst. Hum. 30 (4), 411–420.
21. Rogers, E.M., 2003. *Diffusion of Innovations*, 5th ed. Free Press, New York
22. Bohlmann, J.D., Calantone, R.J., Zhao, M., 2010. *The effects of market network heterogeneity on innovation diffusion: an agent-based modeling approach*. J. Prod. Innov. Manag. 27 (5), 741–760.
23. Kiron, D., Palmer, D., Phillips, A., & Kruschwitz, N. (2012). *Social Business : What Are Companies Really Doing? MIT Sloan Management Review* , 31.
24. Yunus, M., Moingeon, B., & Lehmann-Ortega, L. (2010). *Building Social Business Models: Lessons from the Grameen Experience*. *Long*

Range Planning Elsevier, *43*(2-3), 308–325. doi:10.1016/j.lrp.2009.12.005

25. Hernández S., R., Fernández C., C., & Baptista L., P. (2010). *Metodología de la investigación.* (J. Mares C., Ed.) México, Perú: McGRAW-HILL.
26. Kothari, C. (2004). *Methods of Data Collection*. In C. R. Kothari, *Research Methodology methods and thecniques* (pp. 95-151). New Delhi: new age international (p) limited, publishers.
27. Bockle, G., Clements, P., McGregor, J. D., Muthig, D. y Schmid, K. (2004). *Calculating ROI for software product lines*. IEEE software, 21(3), pp. 23 - 31.