Extending SEMAT kernel to deal with developer error

Marcel J. Simonette, Lucas Lago, Edison Spina

***Abstract*—**Human errors during the software systems development processes are related with software system failure. Usually, these kinds of errors are consequence of the way of work of development team. Although there are a many reliability techniques to steer software systems development process, the repertory of models and techniques to deal with humans, the fundamental element of the process, are very limited. The interplay between Soft Systems Engineering and Software Engineering must be increasingly present in organizations that develop software systems. Soft System Engineering approach helps the software system development team to coordinate their work processes; it allows the development of a resilient environment to human error during development process, and steer the team to the development of software systems with quality. The Endeavor dimension of the SEMAT kernel has the essential elements of Software Engineering, and can be extended, to deal with the human error as early as possible in the software systems development process. This paper is about Work, Way of Work, and Team, three essentials elements of the SEMAT kernel, and Soft System Engineering as a practice that extend the kernel to deal with human error in the software system development.

***Keywords*—**Soft System Engineering, Software Engineering, Software Process, Software Reliability.

I. Introduction

THERE are several organizations in which the development of software systems is not the main business; nonetheless, these companies have a department or area that works with Information Technology (IT). When an IT department defines a team to a software system development endeavor, it is necessary to define a way of work that allows both the team members integration and cooperation, and the understanding of what must be implemented. Furthermore, the teamwork deals with the implementation itself and minimizes systems failures through system validation and verification.

This work was supported in part by the Institute for Technological Research of São Paulo under project 879505A.

M. J. Simonette is with Knowledge Engineering Laboratory (KNOMA) - Department of Computer Engineering and Digital Systems (PCS) of Escola Politécnica da Universidade de São Paulo, São, SP, Brazil. Phone: +5511 30 (910677); +55 11 9 9768 8822; e-mail: marceljs@usp.br.

L. Lago, is with Knowledge Engineering Laboratory (KNOMA) - Department of Computer Engineering and Digital Systems (PCS) of Escola Politécnica da Universidade de São Paulo, São, SP, Brazil; also, he is with Institute for Technological Research of São Paulo, São Paulo, SP, Brazil (e-mail: llago@ipt.br).

E. Spina is with Knowledge Engineering Laboratory (KNOMA) - Department of Computer Engineering and Digital Systems (PCS) of Escola Politécnica da Universidade de São Paulo, São, SP, Brazil (e-mail: spina@usp.br).

The software system development team coordination must be done with focus on the delivery of a software system that achieves both the customer needs and expectations. This kind of mantra of software system development management can be reached by a way of work that considers not only the functional and non-functional aspects of the business demand, but also promotes an approach to deal with human errors that may occur during the software system development process. These human errors - software developer errors - can be classified in two broad categories [1]:

1) Development error made during analysis, design, and coding activities.
2) Debugging errors made during attempts to remove faults identified during verification and validation.

There are several reliability models to software systems, [2], [3], [4] and [5] are only some examples; nevertheless, the repertory of models and techniques to deal with software developer errors are very limited. In any software system development processes, humans are essential. Despite all the Software Engineering methods and practices, the software system development endeavor is still heavily dependent on human activity, and, as argued by Stutuzke and Smidts [1], a single error committed by any member of de development team can inject multiple faults into a software system. Xiong and Li [6] state that these multiple faults occur because software systems are nonlinear systems, in which small changes may bring big impacts to the entire systems.

Soft Systems Engineering offer practices that can enable the integration and collaboration among IT team members, which enable the development of a single vision of the different parts of a system to be developed [7]. It is the vision of the software system as a whole that promotes the identification of the system dependability issues in the beginning of iterations of the work of the software system development endeavor [8], [9]. This approach also allows team managers to identify some factors that have influence in developer errors in software system development processes. Some of these factors are [1], [10]:

1) Lack of resources, tools, to the team, which can be caused by insufficient knowledge, or lack of consideration of proper preconditions or side effects of a decision.
2) Team ability.
3) Time pressure under which team are required to work.
4) Familiarity of the team with the type of system and business rules.

Work, Team and Way of Work are present in one of the three areas of concern that were identified in the Software Engineering Method And Theory (SEMAT) kernel. They take part of the things that "we always work with" in Software Engineering. They are the alphas of the SEMAT kernel Endeavor context [11]. These three alphas have related to IT team activities, things to which team members must be motivated to do, as it is not possible to develop reliable software systems without having professional and motivated personnel [12].

This paper is about Soft Systems Engineering through the life cycle of software systems. A socio-technical approach that promotes the integration of the people that take part of the software system development endeavor; looking for a knowledge construction about the system as a whole, to support Software Engineering practices and to deal with human error during software system development process. This approach contributes to enable IT team to develop software systems that adds value to the organizations, contributing to organization performance and efficiency.

II. SEMAT Kernel – The Essence

SEMAT kernel is the first answer to a "Call for Action" [13], a little step in the process of redefining Software Engineering.

An opportunity identified by a business area of an organization is a search for solution to the IT department of this organization. A solution in which the people that work in the software system development process make use of Software Engineering and Soft System Engineering practices to understand what must be done, validate this understanding and implement the solution. Although there are several practices to conduct this process, there is a set of essentials elements that are universal and are present in all the endeavors to develop high quality software systems. These elements – called alphas in SEMAT kernel – deal with "things we always work with" (Fig. 1) and "things we always do" (Fig. 2) in a software system development process. These elements provide an integrated set that enables the predictability of the delivery, and the continuous improvement in the way of work, communication, and understanding of what, and how, the team is working in a moment, and what they must to do in the following project increments [11].

SEMAT kernel does not compete with existing software system development methods. It is not a new methodology to software system development. It is an answer to the need of redefining Software Engineering [13]. Essence is a kernel of universal elements that are both present in several software system development methods and that enable project measurements of progress and health. The kernel was first published in the SEMAT submission to OMG call: Foundation for Agile Creation and Enactment of Software Engineering [14]. More than a conceptual model, the kernel provides:

1) A framework for software system development teams to think about the endeavor progress and the state of their efforts.
2) A common basis to discuss, improves, compare, and share Software Engineering best practices and methods.
3) A framework to software system development teams assembles practices from different origins, continuously improving the way of work.
4) A framework for defining metrics that are independent of practices, to evaluate both the quality of developed software and the methods used to develop it.
5) And the most important, a way to help software system development teams to understand where they are, what they should do next, and where they need to improve.

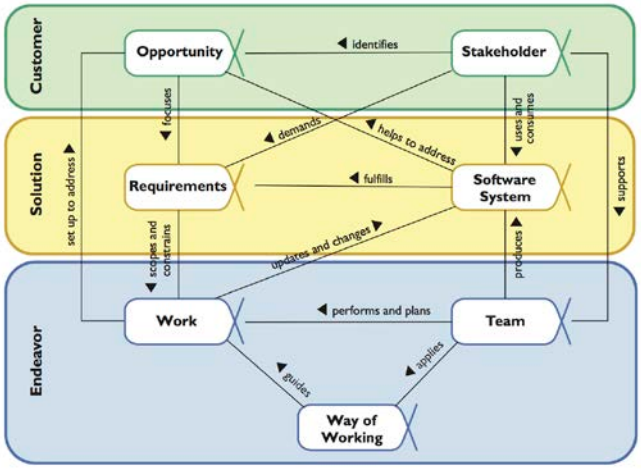

Fig. 1 "things we always work with" in software system development process

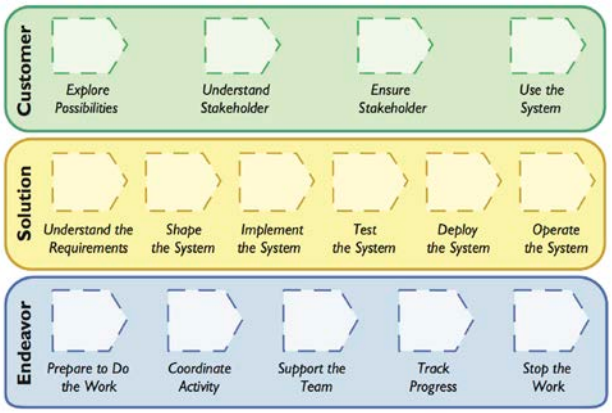

Fig. 2 "things we always do" in a software system development process

A. Endeavor Alphas

Alphas are representations of the essential elements that must be monitored to guide the success of software system development process. Each alpha has a set of states, and each state has a checklist that identifies whether the state has been met. Alphas must be seen together, not in an approach that considers each alpha individually [11].

The kernel alphas that belong to the Endeavor area of concern are: Work, Way of Work, and Team. Observing the

Endeavor kernel dimension at Fig. 1, it is noted that the Team performs and plan the Work, and that the Team applies Way of Work, which guide the Work (Fig. 3).

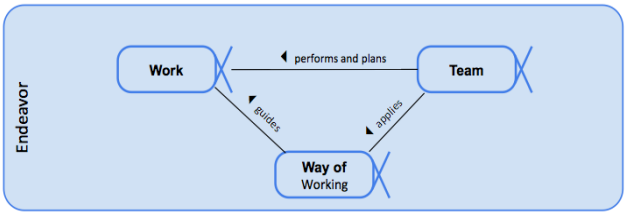

Fig. 3 the alphas of the kernel Endeavor area of concern and their interrelationships

B. Endeavor Kernel Dimension And The Software Inherent Complexity

IT team members are professionals that are under pressure both to be efficient and to produce software systems that meet stakeholders need; they must address the identified business opportunity. There are different ways of work that these professionals can use to successfully fulfill their mission. However, in this endeavor, they must not forget that the work to be performed depends on understanding the requirements of the problematic situation that appears as an opportunity.

Whichever the Way of Work applied by the Team, there is an issue present in all software system development endeavors: the identification of the problematic situation, the opportunity and the requirements that such problematic situation brings. This identification process is essentially an activity related to people talking with people [15], [16], and [17].

Team members work with information and knowledge, obtained through interactions with stakeholders, to understand the opportunity and requirements. In this process, and especially in the implementation of the software system, these team members need to work in a focused way to deal with the underlying complexity of the software system development process [18], [19]. This complexity has also been highlighted by Frederick Brooks in his classic 1987 article: "*No Silver Bullet: Essences and Accidents of Software Engineering*" [20], in which he points out that due to the large number of distinct elements that are present in the software system development process, the development of this type of system is an activity that is more complex than any other type of man-made construction. As evidence of the amount of elements that exist in the software system development process, Clarke & O`Connor [21] argue that there are 40 factors and 170 sub-factors that are present in software system development context.

C. Endeavor Dimension And Its Relationships

Fig. 4 shows all the relationships of the SEMAT kernel alphas that belong to the Endeavor area of concern:

1) Work has relationship with Requirements – Opportunity – Software System.
2) Team has relationship with Software System – Stakeholder.
3) Way of work does not have relationship with alphas of other dimension than Endeavor. However, it is an essential link between Team and Work, and is under direct influence of the factors mentioned by Clarke & O`Connor [21].

Despite the apparent simplicity of the relations between the alphas shown in Fig. 4, there are some challenges in these relations:

1) The cooperation among Team members.
2) The cooperation among Team members and stakeholders.
3) The developer error.

These relationship and the attention to developer error are essential to IT team realize a work that add value to the organization.

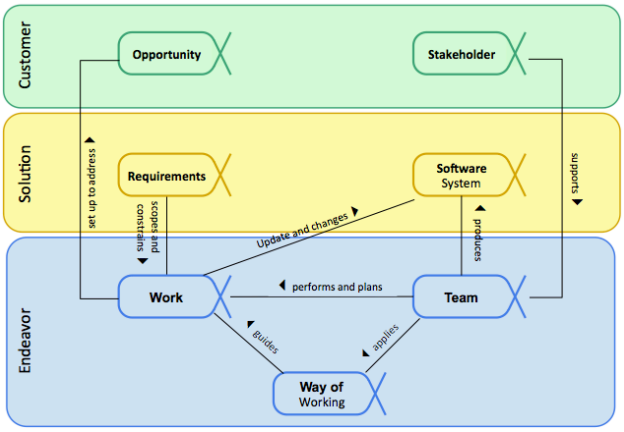

Fig. 4 alphas of kernel Endeavor dimension and their relationship with other kernel alphas

III. Soft Systems Engineering

The Cartesian way of dealing with a problem is to divide it into smaller and simpler parts, as much as possible. This is the most successful technique used by Engineering, and this approach enables humanity to reach the current state of technology. However, even with the successes of this approach, there are problems in which this approach cannot succeed; Soft System Engineering uses System Thinking to deal with these problems. Hitchins [7] argues that System Thinking caught the attention of engineers when they realized that the Cartesian approach have difficulties to deal with systems that include people.

According to Senge [22], Systems Thinking is a discipline to see a system in its totality, a kind of framework to view the system components interrelationship more than to view the components, to view the system patterns of change more than to view system static images. It is an essential approach for effectiveness of environmental management, appropriate leadership and to keep well-regulated interpersonal relations in an enterprise [23].

Hitchins [7] states that Soft Systems Engineering uses Systems Thinking to understand the nature of the problem, seeking practical experiences and interactions with the

problem, trying to understand it and proposing solutions that may not solve it, but can provide a response that improves the problem understanding, allowing the development of a solution which is the best at the moment.

System engineering focuses on a variety of elements, analyzing, designing, and organizing those elements into a system that can transform information and control it [24]. Using a Soft Systems Engineering approach, that is more qualitative than quantitative and that make use of past experience, IT team members can have a practice at the Endeavor context that enables them to deal with developer error and team relationship in software system development process. The human dimension in a software system development process has a complexity that demands team managers to adapt their practices to how people responds to strategies to reach the states of Endeavor essentials alphas of SEMAT kernel.

A. Soft Systems Methodology

Soft Systems Methodology is one of the methods used by Software System Engineering to address complex issues and problems related to the presence of people in unstructured (problematic) situations [7].

Soft Systems Methodology promotes the agreement of the multiple problem views and multiple interests of the people with interest in the problematic situation, and may be represented by a seven-stage model. Stages one and two explore the problematic situation and express it in a rich picture. Stage three is the root definition of the relevant systems describing six aspect of the problem, which are called CATWOE, they are: Customers, Actors, Transformation process, Worldview, Owner, and Environment constrains. In stage four, the conceptual models of the relevant systems are developed, and, in stage 5, the conceptual model is compared with the perceptions of the real situation. In stage six, an action plan is developed for the changes, which are feasible and desirable; and in stage seven, the action plan is implemented. As a method developed from the Soft Systems Thinking, Soft Systems Methodology does not produce a final answer to the problematic situation, it seeks to understand the problem situation and find the best possible response [7], [25].

IV. Extending SEMAT Kernel

Although the developer error during software system development process may happens independently of the development method adopt by the software system development team, it is not a concern present in the "thing we always work with" or "things we always do" of the SEMAT kernel. This absence is not a problem of the kernel concept. The kernel is defined as the essentials elements of Software Engineering, not as "all elements" a team need. However, different kinds of software endeavors have different risks and constrains, and developer error can brings risks to software system development endeavors.

The SEMAT kernel can be scaled to address challenges presents in any kind of software system development endeavor, providing guidance beyond what the kernel provides [11]. This additional guidance comes in forms of practices, which are extensions to the kernel.

The existence of people in the Endeavor context is a strong characteristic of this context, in which relationship among team members occurs. Soft Systems Methodology approach can provide details that team members might need to have a common understanding of how to conduct the software system development to develop a resilient environment to developer error during development process.

SEMAT kernel has a simple language that provides a way to describe practices [11]. The use of Soft Systems Methodology as a practice describe by the kernel language is a way that is easy for inexperienced and experienced team members apply Soft Systems Methodology in the endeavor context.

A. Soft Systems Methodology Practices

The purpose of this practice is to ensure that the team has a common understanding about software system development endeavor to develop a resilient environment to developer error during development process.

In the Endeavor context the Soft Systems Methodology practice can be applied in terms of "things to work with" (Fig. 5) and "things to do" (Fig. 6). Considering the "things to work with", this practice provides guidance to clarify the software system development environment to team members. An action plan to steer the team relationship, and actions to a resilient environment, is the work product attached to the Way of Work alpha. And on "things to do", this practice provides guidance on how to conduct the several activities of the Soft Systems Methodology, namely:

1) Agree on Conceptual Model through the execution of the first five stages of the Soft Systems Methodology. It includes the build of a picture of the environment and identification the core activities required to develop a resilient environment through the comparison of ideal model of the environment with the team perception of the real endeavor environment.
2) Action Plan development trough the execution of Soft Systems Methodology stage six, in which an action plan is developed for the changes in environment, which are feasible and desirable.
3) Action Plan implementation trough the execution of 7 of the Soft Systems Methodology.
4) Observe situation as a way to identify the necessity of a new cycle of the Soft Systems Methodology to control or to improve some action in the environment.

The mapping from Way of Work alpha states to activities of the Soft Systems Methodology practice is showed at Fig. 7. This mapping guides team members in what to do to achieve a particular Way of Work state. For example, the practice recommends conduct the activity Action Plan development (through execution of Soft Systems Methodology stage 6) to achieve the Way of Work Foundation Established state.

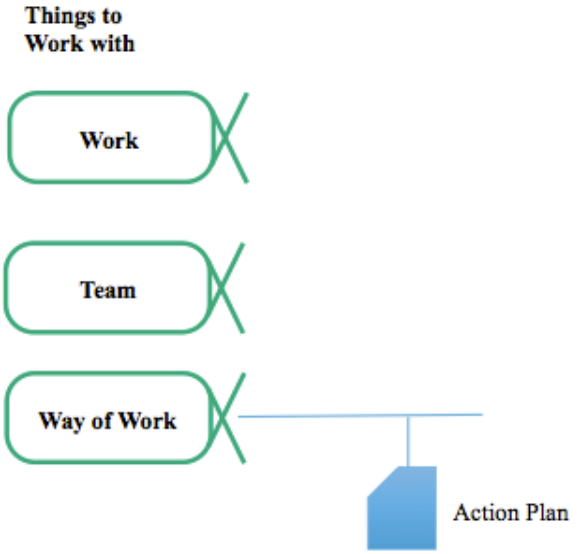

Fig. 5 Soft Systems Methodology practice: Things to work with

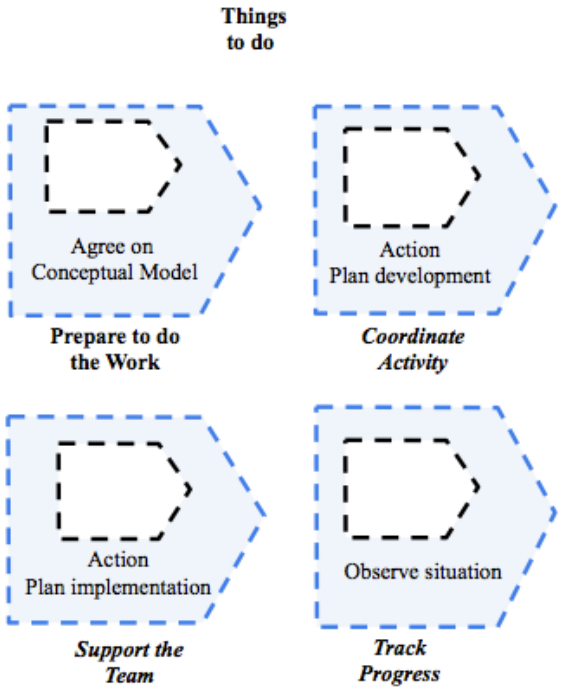

Fig. 6 Soft Systems Methodology practice: Things to do.

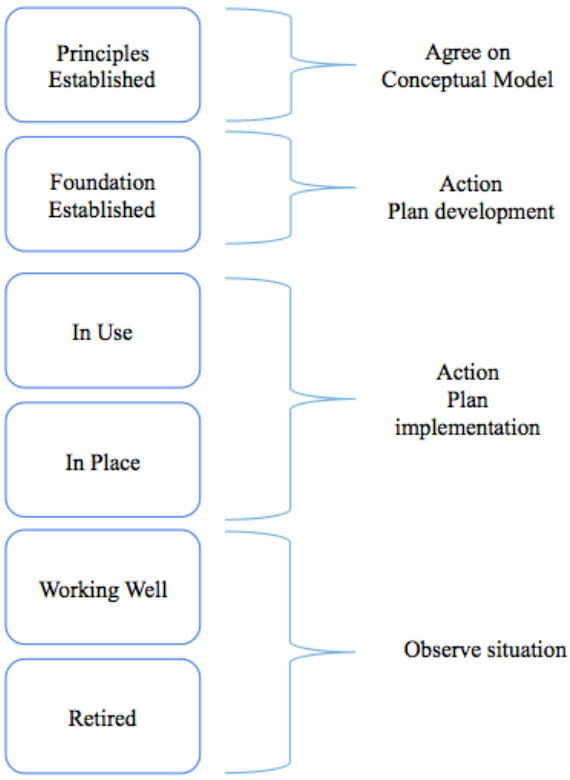

Fig.7 using Soft Systems Methodology practice: Mapping Way of Work alpha states to activities

V. Conclusion And Future Work

It is impossible to predict all the developer errors that can occur in software system development process. The different fashions of Software Engineering methods and practices do not deal effectively with the inherent complexity of software system development because they are based on code development, without considering the inter-relationship between:

1) Technical components (hardware & software).
2) Issues related to the organization or the business process.
3) Issues related to the relationship between the team members.
4) Issues related to the relationship between team members and stakeholders.
5) The real intentions of software system use by the users.

The Soft System Engineering systemic approach enables the IT team to understand what has to be developed, considering the various inter-relationships, and inspire a socio-technical environment in which the IT team objective is more than build a software system, is to add value to the organization business.

The authors of this paper are conducting researches about SEMAT kernel alphas extensions to offer an answer that considers the inherent complexity of software system development process, in which the deal with developer error has a strong presence.

REFERENCES

[1] M. A. Stutzke and C. S. Smidts, "A stochastic model of fault introduction and removal during software development," *IEEE Transactions on Reliability*, vol. 50, no. 2, pp.184-193, Jun 2001. Doi: 10.1109/24.963126

[2] J. Musa, *Software Reliability Engineering: More Reliable Software, Faster, and Cheaper*. 2nd ed., Bloomington: Author House, Bloomington, 2004.

[3] K. Rekab, H. Rhompson, H., and W. Wu, "A multistage sequential test allocation for software reliability estimation," *IEEE Transaction on Reliability*, v. 62, no. 2, pp. 424-433, Jun 2013. Doi: 10.1109/TR.2013.2259195

[4] A. Amin, L. Grunske, and A. Colman, "An approach to software reliability prediction based on time series modeling," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1923-1932, Jul. 2013. Doi: 10.1016/j.jss.2013.03

[5] H. Okamura, T. Dohi, and S. Osaki, "Software reliability growth models with normal failure time distributions," *Reliability Engineering & System Safety*, vol. 116, pp. 135-141, Aug. 2013. Doi: 10.1016/j.ress.2012.02.002

[6] J. Xiong and L. Li, "Nonlinear and Quantitative Software Engineering Method Based on Complexity Science," in *Recent advances in Computer Science: Proceedings of the 17th International Conference on Computers (part of CSCC '13), Proceedings of the 1st International Conference on Artificial Intelligence and Cognitive Science (AICS '13), Proceedings of the 1st International Conference on Innovative Computing and Information Processing (INCIP'13)*, O. Nakov, M. Voznak, V. Vasek, and A. Naaji (Eds.). Rhodes Island, Greece July 16-19, 2013. WSEAS Press, 2013. ISBN: 978-960-474-311-7.

[7] D. K. Hitchins, *Systems Engineering: A 21st Century Systems Methodology*. Chichester: John Wiley & Sons, 2008.

[8] J. C. Laprie, "Dependable computing and fault tolerance. Concepts and terminology," in *Proceedings of 15th IEEE International Symposium on Fault- Tolerant Computing*. IEEE, Ann Arbor, MI, pp. 2–11, 1985. Doi: 10.1109/FTCSH.1995.532603

[9] A. Avizienis, J. C. Laprie, and B. Randell, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004. Doi: 10.1109/TDSC.2004.2

[10] J. Rasmussen and K. J. Vicente, "Coping with human errors through system design: implications for ecological interface design," *International Journal of Man-Machine Studies*, vol. 31, no. 5, pp. 517–534, Nov. 1989. Doi: 10.1016/0020-7373(89)90014-X

[11] I. Jacobson, P. W. Ng, P. E. McMahon, I. Spence, and S. Lidman, *The Essence of Software Engineering: Applying the SEMAT Kernel*. New Jersey: Addison-Wesley Professional, 2013. ISBN: 978-0321885951.

[12] D. Kumlander, "Key Success Factors in Personnel Motivating Projects," in *Proceedings of the 7th WSEAS International Conference on Applied Informatics and Communications (AIC`07)*, vol. 7, pp. 200-205, Athens, Greece, Aug. 2007.

[13] I. Jacobson, P. W. Ng, P. E. McMahon, I. Spence, and S. Lidman, "The Essence of Software Engineering: The SEMAT Kernel, " *Queue 10*, vol. 10, no. 10, Oct. 2012. Doi: 10.1145/2381996.2389616

[14] OMG - *Foundation for Agile Creation and Enactment of Software Engineering RFP*. Available: http://www.omg.org/cgi-bin/doc?ad/2011-6-26

[15] D. C. Gause and G. M. Weinberg, *Exploring Requirements: Quality Before Design*. New York: Dorset House Publishing Co., 1989.

[16] K. Holtzblatt and H. R. Beyer, "Requirements gathering: The human factor," *Communications of the ACM*, vol. 38, no. 5, pp. 30-32, 1995. Doi: 10.1145/203356.203361

[17] N. Maiden, "Trust Me, I'm an Analyst," *IEEE Software*, vol. 27, no. 1, pp. 46-47, Jan.-Feb. 2010. Doi: 10.1109/MS.2010.22

[18] T. Demarco and T. R. Lister, *Peopleware: productive projects and teams*. 2nd Ed. New York: Dorset House Pub. Co, 1999. ISBN: 978-0932633439.

[19] M. Poppendieck and T. Poppendieck, *Lean software development: an agile toolkit*. New Jersey: Addison-Wesley, 2003. ISBN: 978-0321150783.

[20] F. P. Brooks, "No Silver Bullet: Essences and Accidents of Software Engineering," *IEEE Computer*, vol. 20, no. 4, pp.10-19, 1987. Doi: 10.1109/MC.1987.1663532

[21] P. Clarke and R. V. O'Connor, "The situational factors that affect the software development process: Towards a comprehensive reference framework," *Information and Software Technology*, vol. 54, no. 5, pp. 433–447, 2012. Doi: 10.1016/j.infsof.2011.12.003

[22] P. Senge, *The Fifth Discipline: The Art & Practice of the Learning Organization*. New York: Currency Doubleday, 2006.

[23] U. Ogrin and D. Kralj, "Sustainable business and environmental Indicators," *WSEAS Transaction on Communications*, vol. 8, no. 3, pp. 331-342, Mar. 2009. Available: http://www.wseas.us/e-library/transactions/communications/2009/29-164.pdf

[24] E. G. Gallardo, J. O. Hernández, and S. A. De Los Rios, "Knowledge representation of acquisition and control systems with graphical programming using UML," in *Proceedings of 6th WSEAS Int. Conf. on Algorithms, Scientific Computing, Modeling and Simulation (ASCOMS`04)*, Cancun: Mexico, May 2004, ISBN 960-8052-98-X. Available: http://www.wseas.us/e-library/conferences/cancun2004/papers/485-425.pdf

[25] P. Checkland, "Soft Systems Methodology: A Thirty Year Retrospective," *Systems Research and Behavioral Science, vol.* 17, pp. S11–S58, Nov. 2000. Doi: 10.1002/1099-1743(200011)17:1+<::AID-SRES374>3.0.CO;2-O

Marcel J. Simonette was born in São Paulo, Brazil, in 1965. Currently, he is Ph.D. student at Knowledge Engineering Laboratory (KNOMA) - Department of Computer Engineering and Digital Systems (PCS) of Escola Politécnica da Universidade de São Paulo. He holds a degree in Electrical Engineering, 1991, and a M.Sc. in Electrical Engineering (focused on system engineering at sociotechnical systems), 2011; all his titles were obtained in Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil.

He has experience in European Commission Projects as team member in BELIEF (FP6), BELIEF 2 (FP7), and VertbrALCUE (Alfa3). His research areas are the following: Software engineering, requirements engineering, system engineering, and sociotechnical systems.

Lucas S. M. Lago was born in Franca, São Paulo, Brazil, in 1987. Graduated in Computer Engineering at University of São Paulo, Brazil in 2010 and is on the master degree program at the same university.

He is an assistant researcher at the Institute for Technological Research of São Paulo, Brazil.

Edison Spina was born in Jundiai, SP Brazil, in 1958. Currently he is Professor at Escola Politécnica da Universidade de São Paulo. He holds degree in Electrical Engineering, 1981, M.Sc. in Electrical Engineering, 1990, and PhD. in Electrical Engineering, 1988; all his titles were obtained at Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil.

He has experience in Electrical Engineering with emphasis in Human Factors in Reliability and projects working mainly on quality and reliability, technological convergence and heterogeneous networks, systems engineering, and telecommunications engineering. Also, he has experience in European Commission Projects as task leader in INSTINC Project (FP6) and as Brazilian team coordinator for BELIEF (FP6), BELIEF 2 (FP7), and VertbrALCUE (Alfa3).

Dr. Spina is member of the International Relations Committee of Escola Politécnica da Universidade de São Paulo; counselor of the Brazilian Bar (Lawyers) Association (OAB) Science and Technology Committee; a founding member of the iRIOT (Research Group of Interdisciplinary Research for the Internet of Things); member of the IEEE R9 South Brazil Board.