Towards a Metrics Model for DevOps,
Results of a Case Study in an Industrial Company

Jos Trienekens
University of Technology,
Faculty of Industrial Engineering and Innovation Sciences
Eindhoven, The Netherlands
email: j.j.m.trienekens@tue.nl

***Abstract*—Recently in the software industry, a methodology called DevOps has emerged, which aims at the integration of software development and deployment (i.e., operations/maintenance) to improve the performance of the overall software process. DevOps contributes to the multi-dimensional problem of software integration, approaching this problem from an organizational point of view. DevOps originates from lean and agile methodologies and stresses the improvement of the entire process flow, overall product quality improvement based on customer feedback. This paper presents a case study at Philips IT The Netherlands on the implementation of DevOps, in particular on the iterative identification and specification of a metrics model to monitor the effectiveness of DevOps.**

Keywords-DevOps, agile; organizational integration; metrics; case study.

I. INTRODUCTION

Philips IT is a centralized IT organization servicing three business domains, respectively Healthcare, Lighting and Consumer Lifestyle. Within IT, there exist two large parties: IT Delivery, where development projects are planned and executed, and IT Infrastructure & Operations (I&O), which is responsible for the implementation and the daily operations. The latter includes maintenance and control of the IT systems, e.g., providing (helpdesk) support. Delivery has been adopting SCRUM methods over the last three years and their software development methods and techniques become increasingly agile [2], [3]. Currently, there are over 100 SCRUM teams. These teams are multidisciplinary and collaborate with relevant partners on both a business and a technical level. Partners are located across the world, thus collaboration in the SCRUM teams takes place virtually. While Delivery has adopted agile methodologies, I&O has been working in accordance with the Information Technology Infrastructure Library framework, ITIL [4]. Over the years, the two parties have had different objectives and strategies. On the one hand Delivery is pressing for faster software releases (e.g., SCRUM cycles are currently two weeks long), and on the other hand I&O, which considers system stability of the highest importance and plans releases monthly. Recently, the management has decided that Delivery and I&O should integrate and should align their processes to improve the overall efficiency, e.g., to release deliverables in a balanced way and more often without compromising on the quality of the releases. To establish this closer collaboration between Delivery and I&O, DevOps has been introduced. This methodology originates from lean methodologies and stresses the improvement of respectively work flow, final product quality, team communication and customer feedback [1]. The methodology is process flow oriented, which means that it focuses at deliverables moving through the processes, on increasing development speed and decreasing waiting times. The implementation of DevOps has been started with a limited number of teams within Delivery. Because agile software development methods are currently in use at Delivery and also I&O is looking at ways to implement agile methods, it was decided to make explicit use of agile and lean principles in the implementation of DevOps [6], [7], [9]. To monitor and control the DevOps implementation, an initial metrics model had to be developed. In Section II, we will address the background of agile methods and techniques and the key principles of DevOps. Section III will present the methodology used in the research to develop the initial DevOps metrics model. In Section IV, a case study on the development of the metrics model will be presented, following an iterative approach within the company Philips IT. In this case study, researchers in close collaboration with Delivery and I&O practitioners have developed in three cycles an initial metrics model. Section V presents a discussion and Section VI finalizes the paper with conclusions.

II. BACKGROUND AND REFERENCE FRAMEWORK

Agile software development originated from the ‘The Agile Manifesto’ [5] and consists of several values and principles for faster and better software development. Four values are respectively: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan. While there is not a single definition of agility, most approaches incorporate the idea of adaptability to the environment and quick value creation [6]: “agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like.” While this definition is focused on software development, similar trends have been previously seen in other disciplines. In [7] for example, agility is related to "flexibility" and "leanness". However, several differences exist between the terms. According to [8], agility consists of two components: flexibility and speed, hereby stating that flexibility alone is not enough to be agile. In [9], particularly flexibility is addressed, with respect to decision making, and speed with respect to short iterations in development. Comparing agility to leanness, these both complement each other with regard to simplicity and quality, but the economy perspective of the approaches is different [10]. While leanness attempts to remove 'waste' entirely, agility removes waste only to the extent that it does not hinder the ability to change [11]. Next to these definitions on agile a multitude of methods have been developed. Table I reflects the characteristics of a selected set of them.

TABLE I. AGILE METHODS.

Agile method	Description
Scrum [9]	The development is organized in sprints (short iterations of about 2 to 3 weeks) by self-organizing teams. Each sprint, i.e., restricted time, goes through planning, design, testing and review. Features that need to be developed are stored in a 'Backlog' where the product owner decides, which work items will be worked on in the following sprint.
Extreme Programming (XP) [5]	Focuses on best practice and consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40h week, on-site customer collaboration, and coding standards.
Lean software development [11]	Based on seven principles: remove waste, amplify learning and knowledge management, decide as late as possible, deliver as fast as possible, empowered teams, build integrity, and see the whole picture.
Kanban [20]	Kanban is based on the theory of constraints and comes with six core practices; visualize, limit work in progress (WIP), manage flow, make policies explicit, implement feedback loops, improve collaboratively & evolve experimentally.

The agile methods show quite some similarities regarding speed (e.g., fast delivery), small releases (e.g., limit work in progress), remove waste (e.g., manage flow), implement feedback loops (e.g., customer collaboration) and learning and experimentation, and knowledge management. Scrum stresses additionally the self-organization of teams and other team-work characteristics. Since 2009, DevOps has been introduced, which focuses on the way development and deployment (i.e., operations/maintenance) can be integrated [1]. While development teams and deployment teams have often different goals or key performance indicators, DevOps attempts to align the work to be done, and to satisfy the different goals. For example, as development teams want to deploy more and more often, deployment teams strive often towards the exact opposite, i.e., to keep all systems running and stable. However, and in accordance with DevOps, an entire organization should be aligned and/or integrated. To reach this, DevOps proposes to follow three subapproaches [12], see Table II.

TABLE II. THREE APPROACHES OF DEVOPS.

Systems thinking	Stresses that it is more beneficial to look at the performance of an entire system, than at the performance of specific parts of that system.
Amplify feedback loops	Allows understanding of the customer by the teams and availability of knowledge where it is needed.
Culture of continuous experimentation and learning	Experimentation and learning helps to more quickly adapt and respond to changes or problems.

To use these three subapproaches of DevOps as a reference framework, the three approaches can be elaborated on the basis of agile principles. *Systems thinking* refers to looking at problems in relation to the performance of an entire system, also addressed as 'overall quality of work'. This approach ensures that the performance of a system as a whole is more important than the performance of separate parts of the system (e.g., a development and a deployment part). This approach can make use of agile principles (see Table I) such as remove waste, decrease incidents and continuously focus on (process) flow to increase performance. *Amplifying feedback loops* leads to early knowledge of issues and problems, so that a system can quickly be adjusted where needed. Implementing this second subapproach should lead to, with reference to agile issues in Table I, in particular an understanding of, and responding to customers. To deliver finally value, the feedback should come from the people (i.e., customers) who will use the product or service and from those who maintain it. The third subapproach, i.e., a *culture of continuous experimentation and learning*, supports the other two, to ensure that improvement should be a continuous process and should lead to, with reference to the agile principles in Table I, respectively: facilitating knowledge storage and retrieval, and reflection on deliverables and on the way of working. Regarding 'culture of learning and experimentation' references can be made to specific constructs or organizational learning [13], such as the acquisition of knowledge, either through external sources or internal development, the distribution of knowledge, and the interpretation of knowledge (i.e., the way that people within an organization share and use the knowledge).

To implement DevOps on the basis of the three foregoing subapproaches, with the references to agile principles, and to monitor the effectiveness of it, performance indicators or metrics have to be defined. Regarding the development of metrics the Goal-Question-Metric (GQM) approach will be used [14]. Based on well-defined goals of a particular object under study, here the DevOps process, asking questions and getting answers regarding the achievement of the goals, will lead to a well-founded set of metrics. To support the definition of goals, the development of questions /answers, and the derivation of metrics, particular templates will be used [15].

III. Methodology of the case study

The first step in the case study was defining the goals, making use of structured templates [14]. This has been done in collaboration with 'those working in the environment itself' to ensure the understandability and the applicability of the metrics [16]. In this step, we made use of the background as explored in Section II, in particular regarding the three subapproaches of DevOps and the agile pinciples identified. In step 2, a set of metrics has been derived from the defined goals. In this step, in meetings with experts from practice, questions have been developed regarding the defined goal(s) [17]. Subsequently, metrics have been derived to measure the performance. The metrics have formed together an initial metrics model. In step 3, iterations have been executed to elaborate and validate iteratively the set of metrics [18]. These iterations have been stopped in case the set of metrics didn’t change significantly from its previous iterations. The first iteration has been executed with respectively the Manager I&O and the Global Demand Manager (management level above Delivery and I&O). These representatives were selected because the assignment, of the case study at hand, originated from them. A second iteration has been executed with the Delivery Manager. Its position was close to the teams in that the metrics had to be applied.

IV. Towards an initial metrics model for DevOps, the case study

A. *Goal definition for the measurement of DevOps*

To support the goal definition, the following template has been applied [15].

TABLE III. GQM GOAL DEFINITION TEMPLATE.

Analyze	**The object under measurement**
For the purpose of	Understanding, controlling or improving the object
With respect to	The quality focus of the object that the measurement focuses on
From the viewpoint of	The people who have a stake in measuring the object
In the context of	The environment in which measurement takes place

The object under measurement, see Table III, is in this case study the integrated development and deployment process, i.e., the DevOps process within the company. The purpose for the measurement is to further understand this process and if possible to improve it. The focus will be on the three subapproaches within DevOps, respectively systems thinking, feedback loops and a culture of learning and experimentation. The people who have a stake in measuring the object, i.e., reflecting the three viewpoints are respectively the Global Demand Manager, the Delivery manager and the I&O manager. Table IV shows the goals as defined on the basis of the template.

TABLE IV. THE DEFINED GOALS FOR DEVOPS MEASUREMENT.

Goal 1	Analyze the development and deployment process within Philips IT to further understand and improve with respect to systems thinking from the viewpoint of the IT management.
Goal 2	Analyze the development and deployment process within Philips IT to further understand and improve with respect to feedback loops from the viewpoint of the IT management.
Goal 3	Analyze the development and deployment process within Philips IT to further understand and improve with respect to culture of learning and experimentation from the viewpoint of the IT management.

B. *Formulating questions to derive metrics for DevOps.*

Regarding the goal of ‘systems thinking’, it was decided to look at the performance of the process as a whole (i.e., also addressed as the ‘overall quality of work’) opposed to its separate parts. This has lead to the following two questions: what is the current performance of the entire process, and do changes in the process improve the performance of the entire process? Regarding the goal of ‘feedback loops within the system’, the following questions are formulated: what is the current state of feedback loops within the process? Is the customer satisfied with the feedback that can be given? How well can the process respond to feedback? Do changes in the process improve the state of feedback loops within the process? Regarding the the goal of ‘culture of learning and experimentation’, questions are formulated about the current state of the culture, and the improvement of learning and experimentation [13].

C. *Deriving an initial metrics model for DevOps*

Deriving initial metrics for DevOps systems thinking

Following GQM, i.e., answering the questions, metrics have been derived. To describe the performance of the entire process, the average cycle time of a user story has been discussed. While this metric only takes into account the speed of development, it was decided to choose a second metric regarding the ‘overall quality of the work’. The rationale is that higher quality leads to less rework, which should lead to a better lead time [19].

TABLE V. METRICS FOR SYSTEM THINKING.

Questions to goals	**Metrics**
What is the current performance of the process?	Average cycle time of a user story Number of incidents after deployment Costs of a feature
Do changes in the process improve the performance (average lead time: avglt; average number of incidents: avgni) of the entire process?	Avglt of a user story after change -- * 100% Avglt of a user story before change Avgni after deployment after change --- * 100% Avgni after deployment before change

In the case study company, in particular I&O teams are already measuring the amount of incidents that occur following an implementation. Regarding changes in the process, two metrics have been derived (based on the foregoing metrics) to reflect the differences between the performance before and after a change. Table V presents the derived metrics.

Deriving initial metrics for DevOps feedback loops

Initially, the amount of feedback loops has been defined as metric. However, this metric appeared to be depended on the length of the process. To take the length of the process out of the metric, the average time between feedback moments (i.e., the contact points with customers) has been chosen. A problem with this would however be that if the only feedback moment is located at the end of the process, the average time would be same as if the feedback moment would be right in the middle of the process. To cover this, an additional metric has been defined to keep track of the maximum time within a process without feedback. When this time is very close to the average time between feedback moments, the feedback moments will be evenly spread out over the process. Regarding customer satisfaction, a qualitative metric has been defined by asking the customer whether he would like to have the next feedback moment quicker than the time since the last feedback moment. Regarding how well the system can respond to given feedback, a first suggestion was to look at the amount of work, which has to be redone within the process. This can be quantified by the amount of time spent from the moment of feedback until the process reaches the same point again. While this could be difficult to measure in practice, also an easier metric has been defined, i.e., the total time spent on rework during the process. An overview of the second set of metrics relating to feedback loops is shown in Table VI.

TABLE VI. METRICS FOR FEEDBACK LOOPS.

Questions to goals	**Metrics**
What is the current state of feedback loops within the system?	Average time and mMaximum time between feedback moments
Is the customer satisfied with the feedback that can be given?	Need of the customer to have the next feedback moment quicker or later than the time since the last feedback moment
How well can the system respond to feedback?	Time spent from feedback moment untill reaching the same point, total time spent on rework (after feedback)

Deriving initial metrics for DevOps culture of learning and experimentation.

Regarding the current state of learning, two metrics have been defined, respectively with respect to the fact whether new knowledge is actively being stored and whether stored knowledge can be actively retrieved. To determine if knowledge is being shared, as well as whether a mechanism is in place to make sure that knowledge is actually being stored, a metric has been defined on the reflection of a team on its work and learnings points being defined after a project (or a ‘sprint’). Finally, a metric has been on the reflection of a team on their way of working (and thus takes time to improve). The metrics are shown in Table VII.

TABLE VII. METRICS FOR LEARNING AND EXPERIMENTATION.

Questions to goals	**Metrics**
What is the current state of learning and experimentation within the system?	Amount of new knowledge stored during the process Extent to that previously acquired knowledge can be retrieved Extent to that teams reflect on their work and learning points after a project or sprint

D. Iterative refinement of the initial metrics model

First iteration.

The designed metrics model has been refined in the first iteration in two separate sessions. In these two sessions, the initial metrics model was briefly explained, in particular regarding the understandability of the logic of the interrelations between goals, questions and metrics. Subsequently, the participants were asked to come up with alternatives or changes to or extensions of the metrics. Regarding the metrics for ‘systems thinking’, there were three (summarised from the two sessions) main points of feedback. First, regarding the ‘user story’, it was decided that a different unit of measurement had to be used, namely a ‘feature’. The reason was that in the process, a collection of user stories moves through the process simultaneously, except for the part of the process where they are developed. Consequently, measurement of user stories would not provide information about the entire process. Secondly, it was decided that the specification of cost within the process should be further defined. Considering the fact that this process contains quite some knowledge work, and no tangible products, the cost of a feature should be calculated on the basis of the hours spent, the amount of people working on it, and the number of features being worked on. Thirdly, it was decided that by using the metric on the first question periodically or continuously, the second question on change, see Table V, would be irrelevant, and could be removed, see Table VIII.

TABLE VIII. METRICS FOR SYSTEM THINKING, BASED ON FIRST FEEDBACK.

Questions to goals	**Metrics**
What is the current performance of the process?	Average cycle time of a feature Average waiting time of a feature Number of incidents as a result of the feature after deployment The cost of a feature through a process: Hours spent Number of people Number of features being worked on

Regarding the metrics for ‘feedback loops’, in one session the participants mainly agreed on the proposed metrics and suggested some small changes in terminology. In the second session a different understanding of what should happen in feedback loops lead to discussions. On the one hand, it was understood that feedback would internally lead to more insight in how fast changes in the system were executed, while on the other hand the importance of feedback to customers was stressed. It was also suggested that feedback moments with customers had to be changed to so-called ‘touch points’ for a better understanding within the company. These discussions lead finally to Table IX.

TABLE IX. METRICS FOR FEEDBACK LOOPS, BASED ON FIRST FEEDBACK

Questions to goals	Metrics
What is the current state of feedback loops within the system?	Average time between customer touch points Maximum time between customer touch points
How well can the system respond to feedback?	Time spent on feedback untill reaching the same process step Time spent on rework
How fast can the system respond to changes in a process?	The average time a change is seen at the end of the entire process

Regarding the metrics for ‘culture for learning and experimentation’, it was initially more difficult to find useful metrics. Some feedback included the addition of metrics related to the capabilities of the team members, and to how well people could perform the activities of other team members. However, by just measuring the capabilities, it would mean that you can get a culture of learning by simply hiring the people with excellent capabilities. Also suggestions were made that the number of value propositions should be counted. Here, a value proposition would mean a member making a suggestion for a change in the process, or a team, with an estimated value that is estimated by implementing the change. However, this suggestion was rejected because of the time that it would require. It was decided then that the focus for learning should be put on the time spent on improving the teams that perform their daily work. Thus measuring their time spent on storing and retrieving knowledge, and on learning (i.e., reflecting) and improving. Experimentation was considered as very relevant and some discussions lead to a metric on the introduction and subsequent discovery of faults by different teams, see Table X.

TABLE X. METRICS FOR LEARNING AND EXPERIMENTATION, BASED ON FIRST FEEDBACK

Questions to goals	Metrics
What is the current state of learning and experimentation within the system?	The amount of time spent to store new knowledge during the process The amount of time spent to retrieve previously acquired knowledge Amount of time spent on reflection of a team on their work and on learning points after a project or sprint? Amount of time spent on reflection of a team on the way of working after a project or sprint? Percentage of discovered faults by a team with respect to introduced faults by another team (experimentation).

Second iteration

The second iteration consisted of one session and has been carried out with only the Delivery Manager. The feedback in this session mainly consisted of small updates and clarifications. This feedback was more on the confirmation (and validation) of the changes in the foregoing session then in actually changing the metrics. Regarding the first and the second subgoal, two particular terms had to be clarified. Firstly, cycle time was changed to lead time and secondly the cost of a feature was further elaborated by adding service costs. Although the feedback consisted of serious doubts regarding the time that the extra work of experimentation would cost, i.e., introducing and discovering faults, experimentation was kept in the metrics model.

V. DISCUSSION

Metrics development to measure the performance of DevOps requires a structured aproach and a clear reference framework. The implementation of DevOps could be based on three subapproaches, with an explicit reference to agile and lean principles. The application of GQM to determine metrics could profit from this reference framework. The reference framework facilitated the development of questions, the interpretation of the answers and the initial determination of metrics. However, the reference framework is still qualitative and should be investigated furter. Although GQM is an approach that has received positive response in literature, criticism states that the outcome is rather unpredictable as it is still possible to derive many different metrics that describe a particular defined goal. However, our experience in the case study has shown that by carrying out feedback loops, it is possible to discuss and (re)define metrics and to reach consensus on metrics in close collaboration with responsible experts from practice. Although not all derived metrics have clear references to literature, interesting similarities could be found. Regarding the first DevOps subapproach of ‘systems thinking’, parallels have been found in lean manufacturing and agile literature with respect to average lead time of ‘user stories’ and the amount of ‘features being worked’ on simultaneously [19]. However, we couldn’t find Scrum-specific similarities, e.g., regarding our metrics addressing costs and quality (e.g., number of incidents). Regarding the second DevOps subapproach of ‘amplifying feedback loops’, the parallels between our metrics model and literature are more hidden, but are most certainly present. For instance, the time spent on rework (after feedback) is mentioned in agile and lean literature as the percentage of ‘units sent for rework’ [19]. The other metrics found, such as number of approvals, are more closely related to software development in general and are less present in literature on Scrum. Regarding the third DevOps subapproach ‘a culture or learning and experimentation’, the derived metrics turned out to be quite different than what was previously found in literature [13]. Metrics (areas) in literature addressed appeared to be too abstract. Therefore, we have chosen simpler and more direct metrics in terms of ‘time spent on…’.

Reflection on the metrics model from a literature point of view showed that the agile principles identified in lean manufacturing literature turned out to be quite helpful in particular with respect to the first subapproach. However, the metrics investigated in literature on Scrum could not be used in our metrics model. The reason for this is most likely the focus of Scrum metrics. While agile and lean manufacturing metrics focus on the entire process, similar to the focus of our initial metrics model, Scrum metrics focus on teams working within this process. A preliminar conclusion could be that Scrum metrics are probably too team-specific to address the goals of an entire DevOps process. But this should be investigated further, preferably in case studies in that the initial metrics model has to be validated and elaborated further.

VI. CONCLUSIONS

This paper shows that regarding the integration of software development and deployment activities, on the basis of Devops, an initial metrics model could be developed. This metrics model has been developed in a structured way, in a small number of iterations, with responsible practitioners. The objective of the metrics model is the measurement of the effectiveness of the DevOps implementation. The structured GQM-development of the initial metrics model was facilitated by a reference framework, i.e., consisting of the three elaborated DevOps approaches and the agile and lean principles in Section II. This reference framework will also provide a basis for further refinement of the metrics model. Although interesting, and for the company useful, results have been obtained, the metrics model is still in an initial state. In future research and case studies, we will continue the iterative development of the metrics model, towards a well-founded and transparent measurement of the effectiveness of DevOps.

ACKNOWLEDGMENT

The author would like to thank Sander Kruis for his valuable MSc thesis project at TU/e and Philips IT Eindhoven, The Netherlands.

REFERENCES

[1] G. Kim, K. Behr and G. Spafford, The phoenix project: A novel about IT, DevOps, and helping your business win. IT Revolution Press, 2013.

[2] M. Mamun and J. Hansson, “Review and Challenges of Assumptions in Software Development”, Chalmers University and University of Gothenburg, Sweden, (2011), http://publications.lib.chalmers.se/records/fulltext/local_154439.pdf

[3] S. Downey and J. Sutherland, “Scrum metrics for hyperproductive teams: How they fly like fighter aircraft”, 46th Hawaii International Conference on System Sciences (HICCS), 2013, pp. 4870-4878. Hawaii.

[4] ITIL 2011 - The Big Picture, Retrieved 7, 2015, http://cfnpeople.com/downloads/itil_poster_the_big_picture_cfn_people.pdf.

[5] K. Beck and C. Andres, Extreme programming explained: Embrace change, Addison-Wesley, 2000.

[6] J. Erickson, K. Lyytinen and K. Siau. “Agile modeling, agile software development, and extreme programming: The state of research”, Journal of database management, 16 (4), 2005, pp. 13-18.

[7] K. Conboy and B. Fitzgerald, “Towards a conceptual framework of agile methods: A study of agility in different disciplines”, ACM workshop on Interdisciplinary software engineering research, 2004, pp. 37-44, Newport Beach.

[8] Z. Zhang and H. Sharifi, “A methodology for achieving agility in manufacturing organisations”, International journal of operations & production management, 20 (4), 2000, 496-512.

[9] K. Schwaber and M. Beedle, Agile development with Scrum, Prentice Hall, 2001.

[10] K. W. Young, R. Muchlhaeusser, R. Pigging and P. Rachitrangsan, “Agile control systems”. Journal of automobile engineering, 2001.

[11] M. Poppendieck and T. Poppendieck, Lean software development: An agile toolkit for software development managers, Boston: Addison-Wesley, 2001.

[12] G. Kim, DevOps distilled, Part 1: The three underlying principles. Retrieved 7, 2015, IBM Developerworks: http://www.ibm.com/developerworks/library/se-devops/part1/index.html

[13] S. López, J. Peón and C. Ordás, “Managing knowledge: the link between culture and organizational learning”, Journal of knowledge management, 8 (6), 2004, pp. 93-104.

[14] R. Van Solingen and E. Berghout. The goal question metric method: a practical guide for quality improvement of software development, McGraw-Hill Inc, 1999.

[15] V. Basili, G. Caldiera and D. Rombach, “Goal Question Metric Paradigm”, Encyclopedia of Software Engineering, 1, 1994, pp. 528-532.

[16] S. Pfleeger, “Lessons learned in building a corporate metrics program”, IEEE Software, 1993, pp. 67-74.

[17] J. McNiff, Action research; Principles and practice, London & New York: Routledge, 2013.

[18] R. K. Yin, Case study research design and methods, Newbury Park: Sage Publications, 1989.

[19] D. F. Duque and L. R. Cadavid, “Lean manufactoring measurement: The relationship between lean activities and lean metrics”, Estudios Gerenciales, 23 (105), 2007, pp. 69-83.

[20] D. J. Anderson, Kanban: Successful evolutionary change for your technology business, Blue Hole Press, 2010.