IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.Doi Number

Simulation of Software Development Team Productivity Incorporating Social and Human Factors: A System Dynamics Model

Luz M. Restrepo-Tamayo[1], Gloria P. Gasca-Hurtado[1], and Johnny Valencia-Calvo[2]

[1]Universidad de Medellín, Carrera 87 No. 30-65, 50026, Medellín, Colombia
[2]Universidad de Aysén, Obispo Vielmo 82, Coyhaique, Aysén, Chile

Corresponding author: Johnny Valencia Calvo (email: johnny.valencia@uaysen.cl).

This work was supported by Universidad de Aysén, Chile, with funding from the Chilean National Ministry of Education's institutional strengthening projects under Project URY21991.

ABSTRACT Managing software development work teams requires planning resources and activities to complete projects and deliver products satisfactorily and successfully. Estimating project time is part of the planning stage and is mainly conducted using methods based on technical factors. However, since software development is a process involving people with high levels of interaction, it is necessary to consider non-technical factors in project management. This paper presents a simulation model to support informed decision-making during planning, considering that non-technical factors, specifically social and human factors, can affect product delivery time. From a systems perspective, software development is a complex system. Therefore, System Dynamics (SD) modeling based on the rework cycle archetype is used. The resulting model allows for analyzing the productivity of software development teams, integrating three key social and human factors: communication, leadership, and teamwork. The generated burndown charts are used to demonstrate that the model constitutes a basal structure oriented to understand the productivity behavior of work teams. By taking a systemic approach, the model introduces new ways to identify dynamic behaviors and facilitates the prediction of possible scenarios in the evolution of tasks, which helps work teams manage their risks. Additionally, leadership strategies in accordance with the team's status and a good perception of communication can reduce rework and improve the ability to deliver software products on time. To the best of our knowledge, the literature reported on approaches that holistically integrate these elements is limited, which makes this proposal a significant contribution to the discipline.

INDEX TERMS complex systems, human factors, performance analysis, productivity, system dynamics, software development management.

I. INTRODUCTION

Project management is a process aimed at establishing clear objectives, assigning responsibilities, optimizing resources, fostering communication, and monitoring progress [1]. This approach can be interpreted from a systemic perspective since it integrates different performance domains whose interactions and interdependencies directly influence the achievement of the expected results. Among these domains, planning is key because it builds the documents (the path) toward the achievement of the objectives; it defines the scope, estimates resources, structures schedules, and adjusts budgets [2]. The interest in this domain is justified by indicators such as those of the Standish Group Report of 2020, which states that only 35% of software development projects were successful in terms of the estimated time and budget [3].

Because of the above, the estimation of labor effort, duration, and costs has been a topic of interest for research related to software development project management [4]. Various estimation methods, such as function points [5], [6], source code sizing [7], and parametric estimation methods [8], [9], are used to predict the required resources. These methods help to plan, estimate, and control projects, set realistic deadlines, and allocate resources efficiently.

However, effective estimation in software development projects must consider both technical and non-technical factors since the latter also influence team productivity [10].

The inclusion of such factors in planning and estimation is important [11] as it provides a more comprehensive view of team performance and facilitates the identification of actions for improvement [12]. Despite their importance, most studies focus primarily on technical factors [7], in part due to the inherent complexities of quantifying non-technical factors [13], [14]. The above constitutes an obvious line of work to provide solutions in the face of rigorous quantitative methods that include these factors and explore how they affect productivity and their integration into the decision-making process through simulation [15]. This line of work is addressed in this study, which aims to offer analysis tools for the improvement of project management through the strengthening of fundamental areas such as planning and estimation.

System dynamics simulation is an effective paradigm for analyzing complex systems due to its ability to capture non-linear relationships, feedback loops, and behaviors that arise from the dynamic interaction between system components, including systems that involve social aspects [16], and is, therefore, helpful in addressing the goal mentioned above. Although this paradigm has been used to improve software engineering project management [17], [18], the approach has been predominantly technical [19], paying little attention to the social components involved in the development process [20].

Based on the identified gap, this study proposes an SD-based simulation model that incorporates three social and human factors: communication, leadership, and teamwork [21]. Integrating these factors into software estimation in a formal way, using appropriate measurement tools, provides a more holistic view of the team and decreases the subjective component represented by their inclusion. That can lead to more realistic estimates, facilitating informed decision-making and more efficient project management.

Considering the above, the main contribution of this work is a simulation model designed to support decision-making in the planning stage of software development projects. This model analyzes team productivity from a perspective that includes social and human factors based on the formality of complex systems modeling. It also provides a basic framework for understanding team productivity behavior, considering non-technical factors, and offers a framework for team leaders to identify and decide which social and human factors should be intervened.

From the above description, this study answers the following research question: What elements should be considered when designing a dynamic model to study the productivity of software development teams when considering social and human factors?

After the introduction, Section 2 relates the existing literature supporting this study. Section 3 presents the details of the simulation model and the variables that were taken into account. Section 4 presents the results, which are discussed in Section 5. Finally, Section 6 presents conclusions and future work.

II. RELATED WORK

Ensuring the success of software development projects in the context of high information technology consumption and a booming digital economy requires effective team management. Although numerous studies have focused on the importance of technical factors, recent research has highlighted the relevance of non-technical factors in the performance of software development teams. In this regard, the following are some studies from the literature that address strategies to approach the challenges of managing software development teams.

A. SOFTWARE DEVELOPMENT PROJECT MANAGEMENT

The software development process may follow a traditional approach, whose sequential execution includes stages such as requirements gathering, technical solution design, coding, testing, debugging, delivery, and maintenance [22]. However, this software development approach often makes it difficult to modify requirements, which can lead to rework and customer dissatisfaction [23]. In contrast, agile approaches offer an alternative through short iterations of time, where each iteration functions as a self-contained mini-project in which requirements are refined [24]. Continuous communication with the customer allows quick adjustments to be made according to their needs, facilitating an agile and satisfactory delivery of the final product [25].

Software development, under both approaches, is executed by applying initiation, planning, execution, control, and closure processes so it can be framed as a project [2] and, therefore, must be managed [26]. However, software projects are challenging to manage because they have intangible progress; they are complex, depend on customer requirements, and are subject to change [27]. Thus, it is relevant to consider the definition and estimation of aspects such as cost, time, quality, and scope of software development [25].

According to the Project Management Body of Knowledge (2021), planning is one of the key domains in project management. A correct implementation of planning helps to minimize risks, optimize available resources, and increase the probability of success in project execution. This process includes, among other things, the estimation of required resources, the identification of potential risks, and the development of strategies for their management. In this context, the objective of this study (which aims to analyze the estimation of the duration of software development projects, incorporating social and human factors) is justified.

Estimation is a fundamental process in software development project management, as it allows estimating the amount of effort, time, and resources required to complete a project [28]. Due to the complexity, uncertainty, and flexibility inherent in software development [27], estimation

can be challenging. However, it should not be left to chance [26]. To address this challenge, several estimation methods help during the management process [29].

Estimation methods can be classified into non-algorithmic techniques, soft computing techniques, and algorithmic [4]. Non-algorithmic techniques are characterized as flexible and experience-based, and among them, estimation is based on analogy and expert judgment. Soft Computing Techniques, which include Fuzzy logic-based estimation and Artificial Neural Networks, provide a cost-effective solution to problems that are difficult to model mathematically. Finally, the Algorithmic or Parametric techniques are characterized by being less flexible and using mathematical models; among these, we have the Function Point Based Analysis, the Constructive Cost Model, Putnam's SLIM model, SEER-SEM, and the Use Case Based Estimation. The latter category could also include models based on machine learning [30].

Due to their qualitative nature, non-technical factors can be considered in non-algorithmic techniques and Soft Computing Techniques; moreover, some of them have been included in Parametric techniques. For example, the use of Case-Based Estimation considers motivation [8], and a recent proposal for estimation in agile approaches considers face-to-face communication [31]. In both cases, the factors are assigned a weight and a valuation as input data, but the need for concrete elements to measure them is still evident.

A set of non-technical factors, called social and human factors, are perceived as productivity influencers of software development teams [32]. Communication [33], leadership, and teamwork [34] are particularly important among these factors as they help teams meet the challenges of Industry 4.0 [35]. Despite their relevance, these factors still have the potential to be explicitly integrated into estimation processes.

B. REWORK CYCLE

Successful project execution requires a well-defined set of tasks, which is influenced by productivity and the number of people involved in the project. However, the quality of the work can generate rework that may not be immediately detected, which in turn can increase the number of tasks to be performed. This dynamic is known as the Rework cycle, and its structure is presented in Fig. 1 [36].

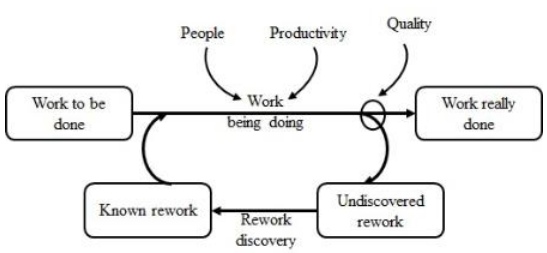


FIGURE 1. The structure of the Rework Cycle (Source: Cooper, 1993).

This cycle has been included or adapted in studies related to project management [16]. For example, it has been used to explore how processes are related at project and business levels and how they affect workload variations [37]. A reformulation allows multiple defects per task [38]. Moreover, it has been considered to study long-range projects, such as those related to construction [39], [40], or aerospace processes [41], which demonstrates its versatility in different disciplines.

Particularly in software engineering, the Rework Cycle has been used as a basis to analyze, from simulation, the management of software development projects. This cycle has allowed studying the influence of factors, such as the availability of prerequisites, morale, or experience, on quality and productivity [42]. It has been applied to research the interaction between knowledge sharing and trust during the requirements analysis phase [43]; it has also been used to understand the interaction between a learning system and defect discovery [44], to analyze the influence of schedule pressure and overtime on productivity [17]. It has also been used to select the best project planning alternative regarding rework uncertainty [18]. In this sense, it is reasonable to start from this archetype [41], [44] to include social and human factors in the estimation of software development projects.

III. MATERIALS AND METHODS

Under systemic thinking, software development processes can be considered complex due to the multidirectional and non-linear interaction dynamics between developers [45]. Decisions made during the project require continuous feedback [46], and the effect of interventions on the process does not have linear or immediate impact [7], [47]. From a strategic perspective, SD is the most appropriate simulation paradigm for this study because it allows for the modeling of the aggregate behavior of productivity at the software development team level, transcending the individual analysis of developers. This approach facilitates the understanding of collective patterns and strategic decision-making, providing a holistic view that is essential for effective software project management.

The modeling process using SD is iterative and consists of five steps [16]: (1) problem articulation, (2) dynamic hypothesis formulation, (3) simulation model formulation, (4) validation, and (5) policy design and evaluation (see Fig. 2). This simulation process has been established as the method for this research work.

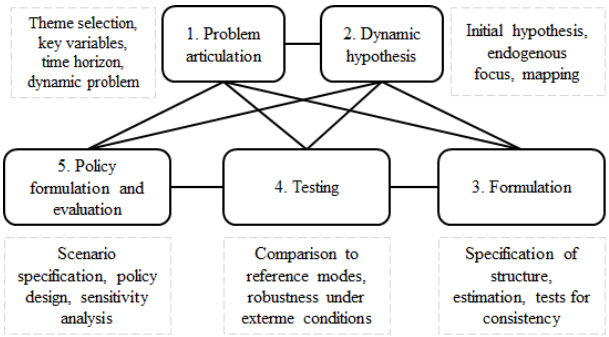


FIGURE 2. **System Dynamics' modeling process (Based on** [16]**).**

In Step 1, it is necessary to identify the problem, why it is a problem, the variables to be considered, and the variables' historical behavior [16]. In this case, the problem articulation focuses on describing the software development process from a dynamic and complex perspective, integrating non-technical factors with the appropriate theoretical support. This approach explores theories related to team dynamics and the behavior of productivity patterns over time. One of the relevant aspects of this step is to characterize the variables, which can be done using a hybrid methodology that includes both a literature review and a survey-based study [48].

The objective of Step 2 is to build a conceptual model or dynamic hypothesis. That requires declaring variables and defining the underlying causal relationships to condition the system of interest's behavior using two tools: a model boundary chart and a causal diagram [49].

The model boundary chart outlines the scope of the model under analysis by categorizing variables as endogenous, exogenous, or excluded [16]. Endogenous variables are those whose behavior is determined by the relationships and feedback within the system. Exogenous variables are those whose behavior is determined by external factors and are not directly affected by the system's internal dynamics. Finally, excluded variables are those that, although they could be relevant for analyzing the system, are omitted because (among other reasons) they add unnecessary complexity to the model or because the literature justifies their absence.

In a complementary manner, the causal diagram formalizes the construction of the dynamic hypothesis using graphs, which reflect whether the causal relationships between two variables, A and B, are positive or negative [50]. A causal relationship is positive (+) if increasing A increases B or decreasing A decreases B (see Fig. 3a). A causal relationship is negative (-) if increasing A decreases B or decreasing A increases B (see Fig. 3b). Causal relationships that are affected by time or information delays are represented by the || symbol.

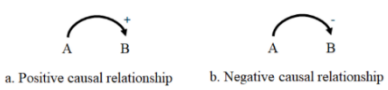


FIGURE 3. **Causal Relationships.**

Feedback loops consist of two or more causal relationships among variables in such a way that, following the causality, one returns to the first variable. Feedback loops can also be positive (+) or negative (-) [51]. The positive ones, also called reinforcement loops, generate exponential growth behavior (see Fig. 4a). Negative or balancing ones generate equilibrium or goal-seeking behavior (see Fig. 4b). Feedback structures generate fundamental modes of systemic behavior, also called archetypes [52]. These structures are frequent in business situations and are constituted by delays, reinforcement loops, and balance loops [53].

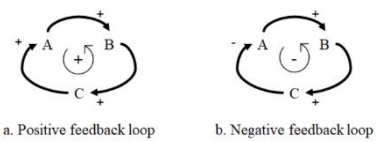


FIGURE 4. **Feedback Loops.**

Step 3, corresponding to the formulation phase, involves defining the input variables along with their respective units, constructing the stock and flow diagram, and providing the equations for all variables included in the diagram [50]. Specifically, the stock and flow diagram serves as an intermediate step bridging the conceptual model and the mathematical representation of the system's behavior [54].

The stocks represent accumulation within the system at a given time and are altered by the inflows and outflows. Thus, stocks are the current values of the variables that result from the cumulative difference between inflows and outflows. Graphically, rectangles are used for the stocks, arrows for the inflows and outflows, valves controlling the flows, and clouds for the sources and sinks of the flows [55], as shown in Fig. 5.

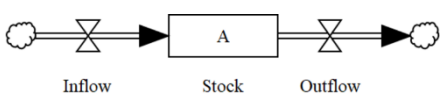


FIGURE 5. **General Structure – Stock and Flow Diagram.**

An SD simulation model is composed of stocks, flows, and decision functions. Decision functions control the flow rates between the stocks from the available system information [56] so that the model's mathematical formulation is done through differential equations [57].

Step 4 suggests tests related to the model's structure and behavior [58]. The model structural tests are:

- Boundary adequacy: Assess the appropriateness of the model boundary for the purpose at hand.
- Structure assessment: Ask whether the model is consistent with knowledge of the real system relevant to the purpose.
- Dimensional consistency: Ensure that variables have correctly specified units, are consistent, and have real-world equivalence.

- Parameter assessment: Ensure that each constant (and variable) has a clear, real-life meaning.
- Extreme conditions: Ensure that the model is robust to extreme values of the input data.
- Integration error: Ensure that the results are not sensitive to the choice of time step or integration method.

The model behavior tests are:

- Behavior reproduction: Evaluate the ability of the model to reproduce the behavior of the system.
- Behavior anomaly: Tests for behavioral anomalies examine the significance of relationships or formulations by asking whether anomalous behavior arises when the relationship is removed or modified.
- Family member: Evaluate whether the model can generate the behavior of other instances of the same class that the system imitates.
- Surprise behavior: Examines unexpected or anomalous behavior.
- Sensitivity analysis: Inquire whether conclusions change in ways that are important for their purpose when assumptions vary within the plausible range of uncertainty.
- System improvement: Find out if process modeling helped change the system for the better.

Finally, in Step 5, different scenarios are built, their sensitivity is studied, and the corresponding recommendations are made [49].

IV. RESULTS

This research aims to present a comprehensive simulation model of software development team productivity incorporating social and human factors. The study addresses the first four steps of the SD modeling process (see Fig. 2). The first step, problem articulation, specifically the characterization of variables, was informed by a prior study. The subsequent steps—dynamic hypothesis formulation, model formulation, and model testing and validation—were fully developed within this research. Step 5, corresponding to Policy Formulation and Evaluation, is identified as future work, focusing on developing a simulation framework that integrates data analysis and intervention strategies. This section concludes by presenting simulation scenarios and discussing the model's limitations.

A. PROBLEM ARTICULATION

This study focuses on the following social and human factors: communication, leadership, and teamwork; this is mainly because these factors are required explicitly by software engineering within the Industry 4.0 context [35]. In a previous study, the characterization of these three factors included their definitions, their relationships with other variables, and the methods used for their measurement. A mixed methodology, including systematic literature mapping and a survey-based study, was proposed and used for this purpose [48]. This characterization allowed the identification that teamwork promotes productivity through leadership actions empowered through communication.

From a systemic perspective, software development can be considered a complex system in which several factors, including team size, influence productivity. Some studies have shown that increasing the size of the development team tends to reduce its productivity [59]. Teams with less than nine members tend to display higher levels of productivity compared to larger teams [60]. Agile methodologies, such as Scrum, recommend teams of up to ten members [61], called small teams [62]. These small teams are dynamic [63], [64] and go through five stages over time [65], [66] as shown in Fig. 6.

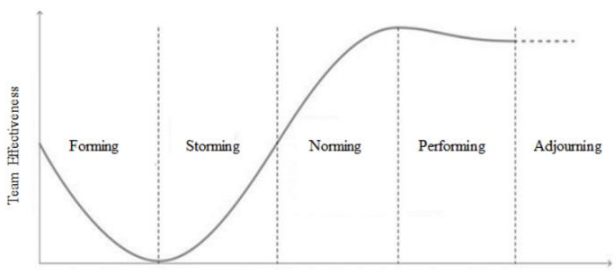


FIGURE 6. Stages of team development, adapted from [65], [66].

The five stages of team development are:

- *Forming:* The members begin to get to know each other and define the team's objectives. Communication is formal, and people tend to be cautious and dependent on the group leader.
- *Storming:* This is a stage of conflicts and challenges. The members question norms and roles. Communication is open and frank, which can lead to disagreements and tensions.
- *Norming:* The team begins to overcome conflicts and define norms and processes to work efficiently. Communication seeks to resolve conflicts and reach consensus. There is an atmosphere of trust and comfort in working together.
- *Performing:* This stage focuses on executing tasks and achieving objectives. There is creativity, initiative, and learning. Communication is task-oriented. Team members leverage individual strengths and abilities. This stage is the most productive.
- *Adjourning:* Occurs when the team prepares to disband. Communication focuses on reflecting on the team's achievements and experiences.

Some teams may go through all the stages [67], [68] or experience setbacks and jumps among them [69]. Each stage involves aspects related to communication among team members and the different roles of the leader [70]. It is advisable to adopt a mentor role if the team is in the forming

stage and an instructor role if it is in the storming stage. If the team is in the norming stage, a coaching role is advisable, while in the performing stage, it is appropriate to assume a facilitator role.

Tuckman's theory provides insight into the development of teams over time, highlighting the relationship between each stage and team effectiveness. It also explains how communication and group dynamics evolve, even in virtual environments [71]. Team leaders and team members can use this theory to manage the transitions and challenges that arise at each stage [72], including through gamification [73]. To identify the stage the team is in, the Group Development Questionnaire (GDQ) [74] or its short version, GDQS [75], can be used. Alternatively, a retrospective questionnaire available in the literature can be used [76].

Teams with low levels of conflict are expected to have high performance [77]. Under this scenario, the project progress can be in line with the ideal progress, which can be evidenced by a burndown chart [28], [78]. Since this graphical representation evidences the aggregate progress of the team [79], then the time horizon of interest is limited by the expected duration of a work cycle.

The analysis of a burndown chart allows revealing problems that require corrective action [79]. On the other hand, this type of chart can evidence anti-patterns of methodologies such as Scrum (Fig. 7), where late acceptance of work done or low team progress may be associated with systemic problems of both the team and the organization [80].

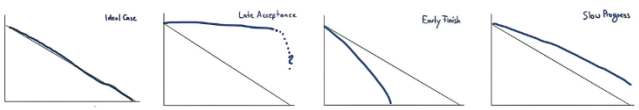


FIGURE 7. Patterns - burndown chart (based on [80]).

B. DYNAMIC HYPOTHESIS

Non-technical factors are intrinsically complex and rooted in culture and social structures, so they do not undergo noticeable changes in short periods [81]. This fact is supported by the use of retrospective instruments to learn about the perception of communication [82], the perception of leadership [83], and the teamwork stage [75]. The latter instrument allows for the estimation of the team's percentage performance according to Tuckman's model.

In software estimation, the team's size is important due to the influence of the experience, skills, and availability of the members on estimating the effort required to complete a project. Having a large team can allow work to be distributed among more people, and potentially, this can speed up the project's development. However, it can also introduce complexities, such as the need for greater communication among its members [10]. In fact, previous studies suggest that it is not advisable to add people to a team when the project has already started [84], [85], [86]. Thus, the size of the team is an element that is analyzed in the estimated nominal capacity and, therefore, is not included in this analysis. Similarly, in this analysis, it is assumed that the estimated nominal capacity considers the technical capacity of the people who constitute the team.

Considering the above description, Table I presents the endogenous, exogenous, and omitted variables that delimit this study's understanding of software development teams' productivity behavior.

TABLE I
MODEL BOUNDARY CHART

Type	Variables
Endogenous	Work to be done - Work done – Rework - Error rate – Discrepancy - Teamwork capacity.
Exogenous	Estimated nominal capacity - Ideal progress - Perception of communication - Strategy from the leader's role - Initial teamwork stage.
Omitted	Team size - Technical capacity

When managing a software development team, the team must remember that work capacity depends on both technical and non-technical factors, which is relevant in the project estimation phase. The estimation process requires calculating the effort, time, and resources needed for its completion [28]. In this study, the estimated effort of the project is represented through the work to be done, which should correspond to the work completed once the work cycle has finished.

The discrepancy is the difference between the work to be done and the ideal progress. The work to be done represents the amount of work that is actually left to be delivered. In contrast, the ideal progress represents the amount of work that should be left to be delivered if no variations occur during the work cycle. In this way, if the work to be done increases, then the discrepancy also increases. Under this scenario, errors arise because delays mean having less time to complete the same number of tasks, which, in turn, increases rework and, thereby, increases the work to be done [36].

On the other hand, if the discrepancy increases due to the growth of the work to be done, then the team is completing fewer tasks than it should, making it necessary to intervene in the team. This intervention may be related to strategies from the leader's role [70], given that leadership is defined as the "ability to direct and coordinate the activities of other team members, assess team effectiveness, assign tasks, develop team knowledge, skills, and abilities, motivate team members, plan and organize, and establish a positive atmosphere" [64].

Effective communication enables successful teamwork [87], promotes problem-solving [64], and serves as a relevant element for coordination between leadership and teamwork [88]. Finally, improvements in teamwork lead to greater productivity [83]. Consequently, improving teamwork increases the team's work capacity and, therefore, reduces the work to be done.

Based on the endogenous explanation and the classification of variables, Fig. 8 presents the causal diagram constructed in Stella Architect®. It evidences a balancing loop called "B1: productivity and social and human factors" and a reinforcing loop called "R1: rework cycle."

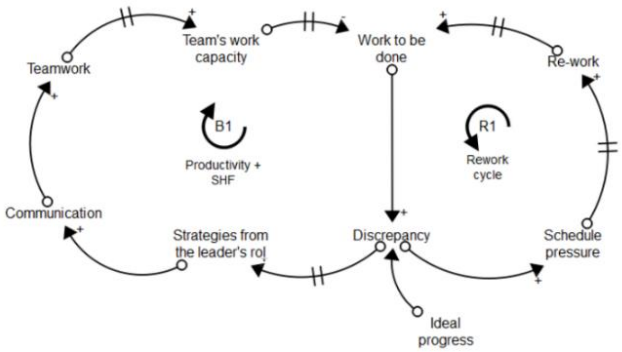


FIGURE 8. **Causal loop diagram.**

C. FORMULATION

The simulation model presented in this study consists of three sectors. The first is related to the generation of rework; the second presents the relationship between the team's work capacity and social and human factors. Finally, the third presents the behavior of the error percentage. For these sectors to adequately reflect the dynamics of the system, specific input data are necessary, which are detailed in Table II.

TABLE II
INPUT DATA

Variable (units)	Description
Estimated nominal capacity (effort/time)	Amount of work that a software development team estimates it can accomplish in each period. Source: Project estimation.
Maturity level (dimensionless)	Organizational capacity to systematically manage and improve its software development processes. Source: Certification – Historical data.
Identified problems (effort)	Technical difficulties identified by the work team during the execution of assigned tasks. Source: Report – Historical data.
Solved problems (effort)	Technical difficulties that the work team can resolve before they are identified as rework Source: Report – Historical data.
Time for rescheduling (time)	The time that the team takes to replan the identified rework. Source: Report – Historical data.
Perception of communication (dimensionless)	The perception that team members have regarding communication. Possible levels: 0 (poor perception, if the average responses per person are between 1 and 39); 1 (good perception, if the average responses per person are between 40 and 50). Source: Teamwork Quality – Communication.
Strategy from the leader's role (dimensionless)	A strategy that adopts the role of the leader during the period of interest. Possible levels: 0 (no strategy); 1 (mentor); 2 (instructor); 3 (coach); 4 (facilitator). Source: Types of Strategy.
Initial teamwork stage (dimensionless)	The state in which the work team is according to Tuckman's model. Possible levels: 1 (forming); 2 (storming); 3 (norming); 4 (performing) Source: Group Development Questionnaire Short

Within the rework cycle, there are four stocks: work to be done, work done, rework, and error percentage. The 'work to be done' becomes 'work done' through the 'work rate,' which depends on the team's work capacity and the error percentage. If that percentage is greater than 0%, the work done is less than expected, as it generates rework that needs to be replanned. Under these conditions, the rework must be done again, increasing the time required to complete the project. In this way, replanning allows for modeling the delay generated by analyzing and rescheduling rework as new work to be done (Fig. 9).

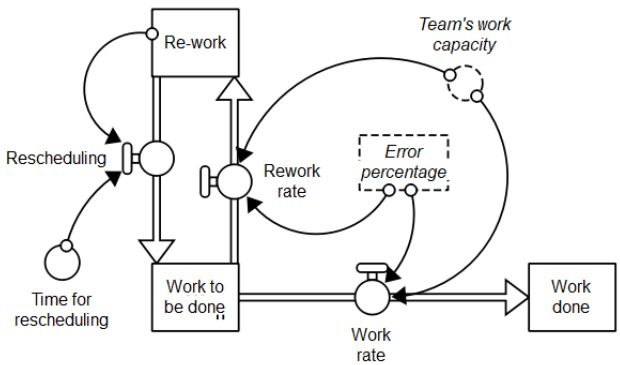


FIGURE 9. **Sector - Rework cycle.**

The initial value of the work to be done corresponds to the effort estimated by the team to complete a milestone or a cycle of the project. The work rate is calculated as the multiplication of the team's work capacity and the complement of the error percentage. The rework rate is calculated as the team's work capacity multiplied by the error percentage, considering a time delay of one unit of time, which represents the period needed to identify the errors made. The rescheduling considers the time delay associated with the team's delay in reallocating the erroneous work. The equations corresponding to the variables involved in the rework cycle sector are detailed in Table III.

TABLE III
REWORK CYCLE FORMULATION

Variable (units)	Equation
Rework (effort)	$\frac{\mathrm{d(Rework)}}{\mathrm{dt}} = \text{Rework rate} - \text{Rescheduling}$
Work done (effort)	$\frac{\mathrm{d(Work\ done)}}{\mathrm{dt}} = \text{Work rate}$
Work to be done (effort)	$\frac{\mathrm{d(Work\ to\ be\ done)}}{\mathrm{dt}} = \text{Rescheduling} - \text{Work rate} - \text{Rework rate}$
Rework rate (Effort/Time)	Team's work capacity × Error percentage
Work rate (Effort/Time)	Team's work capacity × (1 - Error percentage)
Rescheduling (Effort/Time)	Rework / Time for rescheduling

Estimating a project includes defining the time units required to complete it [27]. Thus, in this model, the estimated time is the result of dividing the work to be done by the estimated nominal capacity. If it is assumed that in each period, the amount of work dictated by the estimated nominal capacity is done, it is possible to establish the ideal progress. Considering these variables, the discrepancy represents the difference between the work to be done and the ideal execution.

A negative discrepancy indicates that the ideal performance exceeds the work to be done, meaning that the team is completing more tasks than estimated. There is no need for interventions, as no delays in project completion are evident. On the other hand, a positive discrepancy indicates that tasks are not being executed as planned; therefore, it is necessary to identify the cause of the delay and define a plan of action.

Variations from the ideal execution may arise because the team is not in its most productive stage. In this way, the performance percentage depends on the stage the team is in, according to Tuckman's model (Fig. 8). The result of applying the short version of the Group Development Questionnaire [75] provides the input data for the variable referred to as the initial stage of teamwork. If there is no historical information about the team, it will be assumed that its initial state is Forming.

Leadership can support the actions that guide the team toward achieving objectives [64]. These strategies vary according to the team's stage [70]. According to situational leadership theory, there is no single leadership style that is effective for all situations [89]. However, a leader must adapt his or her style according to the maturity and competencies of the team or individual he or she leads, as well as the specific circumstances of the environment [90]. For this model, the Strategy from the leader's role is input data. It is possible to modify this strategy based on team members' perceptions of formal leadership [83]. If the members agree with the premises set forth in the instrument, then it is assumed that the current strategy is well perceived.

Considering that communication is a necessary coordination mechanism to promote team effectiveness [88], good perception can enhance the strategy from the leader's role. In this way, the perception of communication is input data that can be obtained from the results of evaluating the communication of the Teamwork Quality instrument [82], [91].

Taking the above into account, the effect of teamwork reflects the relationship between the initial stage of teamwork, the strategy from the leadership role, and the perception of communication. Since the team's performance depends on the stage it is at, according to Tuckman's model, the team's work capacity is the estimated nominal capacity, affected percentage-wise by the effect of teamwork. Fig. 10 relates social and human factors to work capacity.

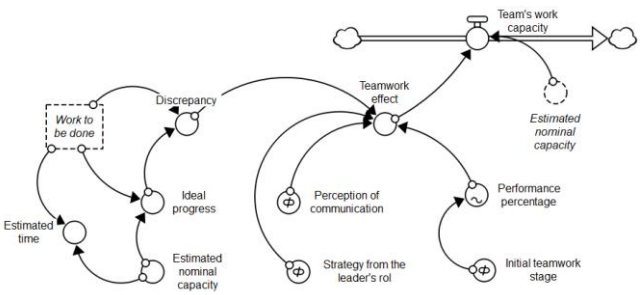


FIGURE 10. **Sector - Work capacity.**

Table IV provides a detailed outline of the equations governing each term involved in the work capacity sector. The INIT() function is a built-in feature in Stella Architect®, designed to retrieve the initial value of the variable specified as its argument. Meanwhile, the GRAPH() function represents a graphical relationship between the variable Performance percentage (y-axis) and Initial teamwork stage (x-axis) based on the data points specified in the table. Additionally, the term TIME is an integrated variable in the simulation environment, representing the progression of time within the dynamic model.

TABLE IV
WORK CAPACITY FORMULATION

Variable (units)	Equation
Team's work capacity (Effort/Time)	Estimated nominal capacity × Teamwork effect
Ideal progress (effort)	INIT (Work to do) - Estimated_nominal_capacity × TIME
Performance percentage (Dimensionless)	GRAPH (Initial teamwork stage) Points: (1,000, 0.400), (2,000, 0.200), (3,000, 0.700), (4,000, 0,900)
Discrepancy (effort)	Work to be done – Ideal progress
Estimated time (time)	INIT (Work to do) / Estimated nominal capacity

The formulation of the auxiliary variable Teamwork Effect (see Table V) depends on the level at which each of the three social and human factors is situated according to the descriptions presented in Table II, in which the NORMAL() function generates a random number from a normal distribution, where the first argument represents the mean, and the second represents the standard deviation. If the strategy from the leader's role is suitable according to the initial level of teamwork, and if good communication is also perceived, then the team will have a higher performance percentage than if poor communication is perceived or if the strategy from the leader's role is not the most appropriate for the level at which the team is.

TABLE V
TEAMWORK EFFECT FORMULATION

Perception of communication	Levels of Teamwork and Strategy from the leader's role	Equation
Good perception	Forming – Mentor Storming – Instructor Norming – Coach Performing – Facilitator	Performance percentage × (1 + NORMAL (0,2;0,1))
Misunderstanding	Forming – Mentor Storming – Instructor Norming – Coach Performing – Facilitator	Performance percentage × (1 + NORMAL (0,1;0,1))
Any other combination of levels of social and human factors		Performance percentage × (1 + NORMAL (0,05;0,1))

The percentage of rework depends on the process' maturity characteristics [92], and it increases as the time to complete the project runs out [93]. However, communication can reduce misunderstandings and problems [64]. Thus, the stock and flow diagram corresponding to the error percentage is presented in Fig. 11.

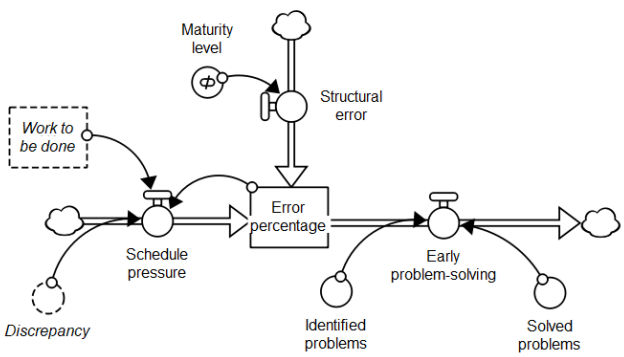


FIGURE 11. **Sector - Percentage of error.**

Schedule pressure can be calculated as the ratio between the actual completion time and the proposed completion time [94]. However, this model proposes its calculation as a function of the work to be done. If the discrepancy is positive, the time pressure is nonexistent because the estimated work is being fulfilled. But if the discrepancy is negative, more tasks need to be completed in less available time, which, in turn, increases the error percentage.

On the other hand, the rework expressed as a percentage of the total development effort depends on the process's maturity. Under the framework provided by the Capability Maturity Model Integration (CMMI), the percentage of rework according to each of the maturity levels is presented in Table VI [92].

TABLE VI
PERCENTAGE OF REWORK ACCORDING TO PROCESS MATURITY

Level	Process maturity	Rework (percent of total development effort)
Level 1	Immature	≥ 50%
Level 2	Project controlled	25% - 50%
Level 3	Defined organizational process	15% - 25%
Level 4	Management by fact	5% - 15%
Level 5	Continuous learning and improvement	≤ 5%

Approximately 79% of the companies certified in CMMI are at level 3 [95]. However, it is important to acknowledge the existence of companies that are considered small organizations as they are made up of fewer than 25 people [96]. For these companies, implementing the ISO 29110 standard is beneficial since it promotes good practices in software development and maintenance [97]. Some studies suggest that the implementation of the ISO 29110 standard can reduce rework to levels comparable to those of CMMI level 3 [98]. Considering the above, the maturity level is a required input in the model. If the company has historical data on the percentage of rework or a CMMI certification, it can be one of those suggested in Table II. In the absence of this information, Level 3 will be assigned if ISO 29110 has been implemented or Level 1 otherwise.

Finally, a strategy to mitigate errors when developing software is to promote communication within the team [93], which may be related to the team's ability to solve problems [99]. In this way, the early solution of problems can be calculated as the ratio between the identified problems and the resolved problems. These values can be obtained from the collaborative platforms used by software development teams [48].

Table VII presents the formulation for the error percentage sector. Here, the IF THEN ELSE structure represents a logical comparison, while the RANDOM() function generates a uniformly distributed random number between the two specified arguments. Additionally, STOP TIME defines the simulation's temporal duration, which can either be set to a fixed value or depend on another variable. In this case, STOP TIME aligns with the Estimated time.

TABLE VII
ERROR PERCENTAGE FORMULATION

Variable (units)	Equation
Error percentage (Dimensionless)	$\frac{d(\text{Error percentage}}{dt} = \text{Structural error} + \text{Schedulde pressure} - \text{Early problem solving}$
Schedule pressure (Dimensionless / Time)	IF Discrepancy < 0 OR Work to do = 0 THEN 0 ELSE (Error percentage + (Discrepancy / INIT (Work to be done))) / STOP TIME
Early problem-solving (Dimensionless / Time)	(Solved problems / Identified problems) / STOP TIME
Structural error (Dimensionless / Time)	IF Maturity level = 1 THEN 0.5 / STOP TIME ELSE IF Maturity level = 2 THEN RANDOM (0,25; 0,50) / STOP TIME ELSE IF Maturity level = 3 THEN RANDOM (0,15; 0,25) / STOP TIME ELSE IF Maturity level = 4 THEN RANDOM (0,05; 0,15) / STOP TIME ELSE RANDOM (0; 0,05) / STOP TIME

Before the testing, it is important to clarify the assumptions and limitations of the proposed model. In software estimation, the size of the team is considered, as the experience, skills, and availability of its members influence the effort required to complete a project. While a larger team can better distribute tasks and accelerate development, it also introduces greater complexities, especially in terms of communication [10]. In fact, previous studies suggest that adding more people to a team after the project has started can be counterproductive [84], [85], [86]. Therefore, the team size is analyzed within the estimated nominal capacity, but it is not considered as an input in the model.

Now, if the team size has been considered to determine the estimated nominal capacity, it can be inferred that the team leader has already identified the members. In this way, one of the model's assumptions is that the team members have the necessary technical skills to carry out the project. Furthermore, the model assumes that the scope of work will remain the same during the simulation period.

Regarding teamwork, this proposal employs the sequence of small group stages proposed by Tuckman [62], [65] since it is related to communication and leadership [70]. This proposal has limitations related to the lack of rigorous quantitative research since it was constructed based on a literature review and observations of some work groups [100]. Furthermore, there is a lack of analyses related to (i) the possibility of a team being in multiple stages simultaneously, (ii) the possibility of a team reaching the performing stage, and (iii) the factors that influence the rate of progression from one stage to another [101].

Despite the reported limitations in the literature regarding Tuckman's proposal, it is recognized as a model that facilitates the understanding of the intrinsic complexity of team dynamics and can be used to manage it [102]. It is a widely referenced theory in the field of human resource development [100]. It has been used in research related to group work in educational projects [103], [104], [105] and under agile methodologies [106], [107], [108], which justifies its selection for this research.

Alternatively, the Multifactor Leadership Questionnaire (MLQ) is an instrument that allows the identification of whether the type of leadership is transformational, transactional, or laissez-faire, and it has been used in software engineering [48]. However, it is a licensed instrument that requires expertise to interpret its results, and some studies recommend limiting its use due to validity issues [109], [110]. In this study, the model focuses on leadership perception, allowing a more accessible analysis without the need to identify a specific type of leadership or obtain a license for this instrument.

D. TESTING

Assessing the simulation model is essential for identifying and correcting errors in the early stages of the modeling process [16]. Although this evaluation should be conducted from the beginning, various tests must be executed, as their verification strengthens the reliability of the model before its use [111]. The results in Table VIII indicate that the model structure reflects the productivity behavior of software development teams during short cycles.

TABLE VIII
MODEL STRUCTURAL TEST

Test	Testing procedure and results
Boundary adequacy	Expert pairs in SD and Software Engineering reviewed both the causal diagram and the stock and flow diagram. Based on this review, the naming of some variables was adjusted, and others were reformulated to enhance the understanding of the model and its subsequent application.
Structure assessment	The research team reviewed the variables and equations to ensure that the model adequately represents the system's knowledge, is coherent, and is relevant to the analysis's purpose.
Dimensional consistency	The dimensions of the variables were assigned according to the literature on the productivity of software development teams. During the formulation of the model, the absence of warnings in Stella Architect® related to the dimensions of the variables was verified.
Parameter assessment	Expert peers in software engineering verified that all the variables made sense in the reality of team management.
Extreme conditions	Each equation was inspected, and it was verified that the model responded appropriately to extreme values of the input variables. The variables were modified both individually and collectively to conduct this test.

Test	Testing procedure and results
Integration error	It was verified that the simulation results were similar when changing the time delta and the integration method.

Behavioral tests intend to verify that the model adequately reproduces the observed behavior patterns in the real system, under normal conditions and in stress scenarios. By comparing the model's output with historical data and conducting sensitivity analyses, it is possible to assess whether the variations in the model's parameters produce coherent and realistic responses. During this process, it is possible to identify and correct potential discrepancies, ensuring that the model is a reliable tool for analysis and decision-making. Table IX relates the types of behavioral tests conducted and the results. The model is thus appropriate.

TABLE IX
MODEL BEHAVIOR TEST

Test	Testing procedure and results
Behavior reproduction	The simulation results were compared with real data. The resulting R2 values, exceeding 90% (Table X), suggest that the model reproduces the behavior of the system of interest in this study within the assumptions defined for this research and in accordance with the previously presented dynamic hypothesis.
Behavior anomaly	All the variables presented in the causal diagram (Fig. 7) are relevant, as anomalous behaviors were evidenced when they were removed from the model.
Family member	Project management methodologies, commonly associated with software development, are applicable to a wide range of disciplines. In this regard, the proposed simulation model can replicate the behavior of teams that use burndown charts to monitor their progress, regardless of the application domain.
Surprise behavior	The research team analyzed each of the variables graphically and in tabular form, and no anomalous behaviors were identified that suggested errors in the formulation.
Sensitivity analysis	The sensitivity analysis using Stella Architect® allowed us to identify that changes in the parameters affect the productivity of software development teams within a range consistent with the real system.
System improvement	Including non-technical factors in a specific manner and indicating how they are measured contributes to process improvement, particularly related to the configuration of a team prior to project execution.

E. SIMULATION SCENARIOS

Two work teams were analyzed in parallel during a short phase of a software development project to carry out the same activities. Both teams estimated 100 units of effort that need to be completed (work to be done = 100), and each team indicated that they could deliver 20 units of effort per day (estimated nominal capacity = 20). In this way, the time in which the tasks associated with those units of effort should be completed is five days (estimated time = 5). Based on historical data, around 20% of rework is generated (maturity level = 3), and when rework is identified, it is rescheduled to the same day (rescheduling time = 1). Additionally, for every ten identified problems, each team indicated that they were capable of early resolution for 2 of them.

On the other hand, the team perceives that communication is good, according to the responses to this variable from the Teamwork Quality instrument (perception of communication = 1). The leader adopted a facilitator role (strategy from the leader's role = 4). The application of the GDQS instrument allowed us to identify that one of the teams is at a more advanced stage of development (initial teamwork stage = 4: performing) while the other is at the preceding stage (initial teamwork stage = 3: norming). Each day, the team reported both the work completed and the rework identified, data from which the actual work to be done was calculated. Table X presents the burndown chart for each team, including both simulated and real data, along with the respective coefficient of determination.

TABLE X
SIMULATED DATA VS REAL DATA

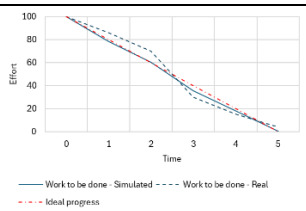
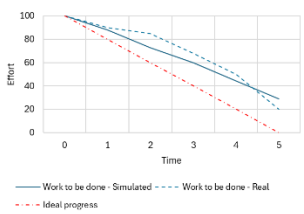
Team Stage	Burndown Chart	Coefficient of determination R^2
Performing		97,97%
Norming		94,37%

Fig. 12 shows the burndown chart and the simulated rework behavior of a team in the performing stage, maintaining the previously presented conditions. Although rework is generated, the positive perception of communication and the alignment between the strategy from the leader's role and the team's stage of development allows for the planned tasks to be completed within the estimated time. The behavior shown in this burndown chart is ideal, as indicated in Fig. 7.

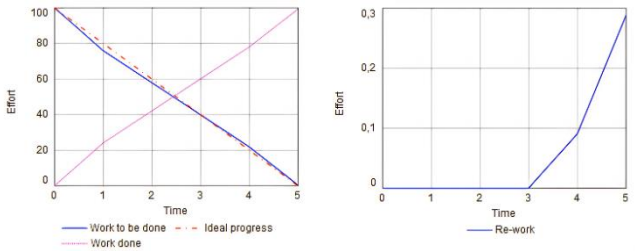
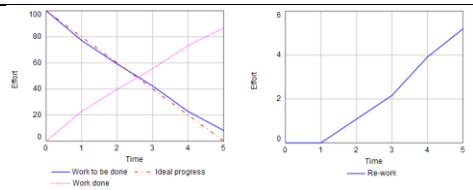
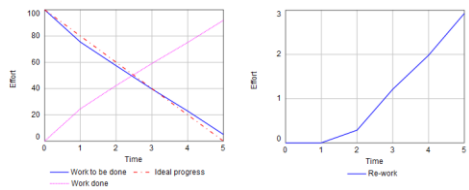


FIGURE 12. **Burndown chart and rework – Ideal case.**

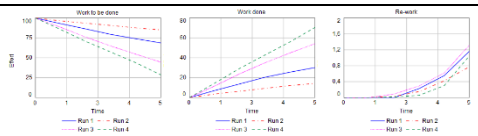
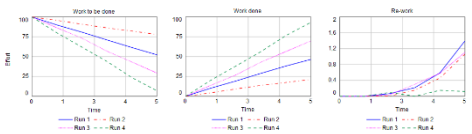
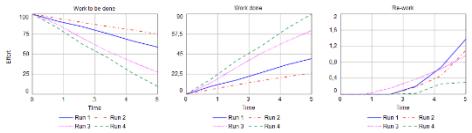
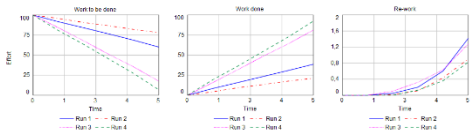
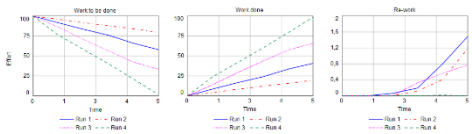
From an organizational viewpoint, the team may experience slight delays due to the maturity process being in Immature (level 1) or Project-controlled states (level 2). Although a rework percentage exceeding 25% is anticipated (which is expected to lead to late project completion), being in the performing stage, experiencing effective communication, and having a suitable strategy could mitigate the impacts of these delays on the project's delivery (Table XI).

TABLE XI
SIMULATIONS OF TEAMS WITH LOW MATURITY LEVELS

Maturity level	Burndown Chart and Rework
Level 1 - Immature	
Level 2 - Project controlled	

From the perspective of social and human factors, low progress may be due to the team not being in the "performing" stage, which represents the highest level of performance. Furthermore, the leadership strategy employed may not be the most suitable for the stage of development the team is in, or the members perceive deficiencies in communication. Table XII presents the work to be done, the work completed, and the rework, simulated based on the proposed model, incorporating social and human factors. In this analysis, the maturity level remains at level 3, and 2 out of every ten identified problems are resolved early; additionally, it is assumed that the team perceives communication well. The runs correspond to each of the stages of team development: Run 1 corresponds to forming, run 2 to storming, run 3 to norming, and run 4 to performing.

TABLE XII
SIMULATIONS RELATED TO LOW PROGRESS ASSOCIATED WITH SOCIAL AND HUMAN FACTORS

Strategy from the leader's role	Behavior of the stocks: work to be done, work done, and rework
Without a strategy	
Mentor	
Instructor	
Coach	
Facilitator	

The simulations' analyses (Table XII) reveal that the absence of a clear leadership strategy is associated with lower team performance, especially during the storming stage (run 2). That is an aspect related to the fact that teams in the early stages of formation face problems that take longer to resolve [112].

These results suggest that, even in teams that reach the performing stage, effective leadership is essential to ensure the success of the project [113]. Furthermore, although progress can often be slow, the team's performance tends to improve when there is alignment between the leader's role strategy and the stage the team is in [114]. However, this suggests that strategies can be viewed beyond a simple classification of good or bad, and it is preferable to evaluate them based on their relevance, according to the team's reality.

V. DISCUSSION

Software development can be considered a complex system because developers interact strongly with each other [45]. Throughout the execution of the project, decisions are made based on feedback [46], and the effect of the interventions made on the process is often non-linear [7], showing their results gradually [47]. Furthermore, the interest in making more strategic decisions and studying the productivity

behavior of the software development team rather than the individual developers requires a high level of aggregation, which is why SD is preferred over other paradigms, such as Agent-Based Simulation [115] or Discrete Event Simulation [55].

System Dynamics models in Software Engineering can address the complexity of strategic issues in traditional management approaches [116] and serve as a tool to support planning and control at the tactical level [46]. Additionally, it is possible to simulate the typical behavior of software development projects to address planning and control issues [117].

The schedule and delivery time of software development projects have been of interest [118], [119], [120], [121], [122], as well as resource allocation [123] and the inspection and testing process [124], [125], [126]. These proposals recognize that SD is a simulation paradigm that facilitates the management of software development projects.

Regarding non-technical factors, Caulfield and Maj (2002) indicated that including soft variables in simulation models allows for more informed sociotechnical models. From a qualitative perspective, factors such as leadership, self-management, and adaptability have been incorporated [128], [129]. Likewise, some simulation models address aspects such as openness and the ability to understand the ideas and interests of others [130], team interaction, individual behavior [131], communication overload [27], [86], [94], [132], and motivation [133]. Even though these research works consider non-technical factors, there remains a need for an approach that adequately integrates them, considering the available instruments for their measurement. In fact, having strategies to obtain data for simulating software processes remains an aspect to be addressed [134]. Regarding this gap, the contribution of this study lies in the formalization of a structured evaluation framework based on a simulation model that incorporates non-technical factors using the tools reported in the literature.

The simulation model presented in this study can capture the complex dynamics of software development team productivity over short time frames, incorporating technical, non-technical, and organizational aspects. This integrative approach allows for estimating the behavior of productivity in response to variations in aspects such as the level of organizational maturity or the stage of development in which the work team is situated. In this way, the model reveals relationships among these aspects, showing, for example, how an adaptive leadership strategy can enhance the team's work capacity. The results obtained align with previous studies that highlight the interdependence of technical and human factors [135] and, on the other hand, validate the effectiveness of the approach used [134], [136]. These considerations reinforce the applicability of the model in real-world scenarios, providing recommendations that support the management of software development teams when a project needs to be executed.

The simulation provides an analytical framework that facilitates the identification of strategic interventions in project management, allowing the adoption of practices that enhance team effectiveness [137]. In this case, the proposed simulation model highlights the importance of maintaining effective communication and adaptive leadership, especially in processes characterized by a high level of human intervention. Adjusting these non-technical aspects can mitigate internal friction and enhance productivity by strengthening teamwork [83]. In this way, the model presented in this study serves as support for informed decision-making, contributing to continuous improvement in software development.

Using simulation models based on SD is convenient to support process improvement in software development [138]. In this regard, Ramdoo and Gukhool [139] proposed a model that involves the relationship between maturity levels and fluctuations in software quality. Aligned with this approach, the model proposed in this study integrates information about the level of maturity, considering CMMI and its relationship with ISO/IEC 29110. This aspect is relevant because it facilitates the identification of improvement opportunities, promoting the implementation of best practices that are aligned with high standards in the software industry. Including ISO/IEC 29110 [140] in this type of model is the first step to support their dissemination and adoption in small information technology companies, given that these companies represent at least 70% of the sector [96].

In the field of Software Engineering, few studies use SD models and report in detail the stock and flow diagrams along with the underlying equations that support them [20]. This situation limits the replicability and validation of the models presented [137]. This study addresses this limitation and stands out from other published works by providing a detailed description of both the stock and flow diagrams as well as the equations used. This approach aims at facilitating the review, implementation, and application of the model by researchers and professionals in this field.

VI. CONCLUSIONS AND FUTURE WORK

This study presents a productivity simulation model of software development teams based on System Dynamics. The model incorporates three relevant social and human factors in its analysis: communication, leadership, and teamwork. The measurement of these factors, the establishment of causal relationships among them, and the transition from a qualitative to a quantitative approach culminating in a robust simulation model with explicit equations represent a significant contribution to empirical research in Software Engineering.

The results of the simulations provide a variety of patterns of the aggregate productivity of software development teams. Thus, the model proposed in this study represents a foundational structure for understanding the productivity behavior of these teams. This aspect contributes to team management, as it allows the identification of improvement

opportunities related to social and human factors and enables decisions aimed at strengthening team effectiveness.

In summary, key findings include:

- Leadership strategies aligned with the team's developmental status and a positive perception of communication reduce the percentage of rework.
- Teams in advanced stages of development (such as the performance stage) achieve higher levels of productivity, especially when they have adaptive leadership.
- The proposed system dynamics model provides a robust tool for analyzing and predicting productivity behaviors, facilitating strategic decisions in project management.
- The inclusion of non-technical factors in planning improves the accuracy of estimates and the design of more effective strategies in team management.

Regarding the simulation model, considering that non-technical factors are treated as input data, the methodology proposed by McLucas (2003) could be adapted to include other factors once the problem articulation stage is completed. Given that the simulation model presented in this study corresponds to an early stage of project management, it would be advisable to adapt the model for the subsequent stages of software development projects, where the work to be done may be affected by additional requirements or requests made by the user at later points after the project has started.

Furthermore, we propose to integrate other paradigms, such as agent-based simulation, as a future research direction, with the aim of providing more comprehensive analyses and exploring multiple perspectives on the productivity behavior of software development teams.

Another future line work corresponds to the design and construction of a simulation framework based on the model presented here so that software development project leaders have an environment that facilitates decision-making during the project planning stage. This line includes the design of interventions for the team related to social and human factors. This framework, in addition to being a tool for managing work teams, also contributes to the Sustainable Development Goals [142] by providing decision-making tools in the growing software industry [143], which is one of the key players in enhancing the technological capacity of industrial sectors in the context of the fourth industrial revolution [144], [145].

VII. ACKNOWLEDGMENT

This research was supported by a collaborative study between Universidad de Medellín (Colombia) and Universidad de Aysén (Chile). The authors would like to thank the Editor and the anonymous reviewers for their helpful comments. During the preparation of this work, the authors used ChatGPT to improve the manuscript's readability and language. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

REFERENCES

[1] W. Royce, *Software Project Management A unified framework*. 1998.

[2] Project Management Institute, *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)*, Séptima ed. 2021.

[3] H. Krasner, "The Cost of Poor Software Quality in the US: A 2020 Report," 2021.

[4] B. Sharma and R. Purohit, "Review of current software estimation techniques," *Communications in Computer and Information Science*, vol. 799, pp. 380–399, 2018, doi: 10.1007/978-981-10-8527-7_32.

[5] A. J. Albrecht, "Measuring application development productivity," 1979.

[6] C. A. Behrens, "Measuring the Productivity of Computer Systems Development Activities with Function Points," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 6, pp. 648–652, 1983, doi: 10.1109/TSE.1983.235429.

[7] C. Sadowski and T. Zimmermann, *Rethinking Productivity in Software Engineering*. New Yotk: Springer, 2019.

[8] G. Karner, "Resource estimation for objectory projects," *Objective Systems SF AB*, pp. 1–9, 1993.

[9] P. C. Pendharkar, "Probabilistic estimation of software size and effort," *Expert Syst Appl*, vol. 37, no. 6, pp. 4435–4440, 2010, doi: 10.1016/j.eswa.2009.11.085.

[10] C. H. C. Duarte, "Software Productivity in Practice: A Systematic Mapping Study," *Software*, vol. 1, no. 2, pp. 164–214, 2022, doi: 10.3390/software1020008.

[11] U. M. Devadas and Y. Y. Dharmapala, "Soft skills Evaluation in the Information Technology and Business Process Management Industry in Sri Lanka: Skills, Methods and Problems," *International Journal of Economics Business and Human Behaviour*, vol. 2, no. 3, 2021, doi: 10.5281/zenodo.5280309.

[12] D. T. W. Wardoyo and R. S. Dewi, "Agile Leadership Cost Estimation Model in Software Development Project (Case Study: Public Service Applications)," *Proceedings - 2023 6th International Conference on Computer and Informatics Engineering: AI Trust, Risk and Security Management (AI Trism), IC2IE 2023*, pp. 271–275, 2023, doi: 10.1109/IC2IE60547.2023.10330999.

[13] E. D. Canedo and G. A. Santos, "Factors affecting software development productivity: An empirical study," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Sep. 2019, pp. 307–316. doi: 10.1145/3350768.3352491.

[14] L. F. Capretz, F. Ahmed, and F. Q. B. da Silva, "Soft sides of software," *Inf Softw Technol*, vol. 92, pp. 92–94, 2017, doi: 10.1016/j.infsof.2017.07.011.

[15] J. A. García-García, J. G. Enríquez, M. Ruiz, C. Arévalo, and A. Jiménez-Ramírez, "Software Process Simulation Modeling: Systematic literature review," *Comput Stand Interfaces*, vol. 70, no. August 2019, p. 103425, 2020, doi: 10.1016/j.csi.2020.103425.

[16] John. D. Sterman, *Business Dynamics: systems thinking and modeling for a complex world*. Mc Graw Hill, 2000.

[17] K. Hiekata, M. T. Khatun, and M. A. Chavy-Macdonald, "System dynamics modeling to manage performance based on scope change for software development projects," in *Proceedings of the 26th ISTE International Conference on Transdisciplinary Engineering*, Tokyo, Japan, 2019, pp. 675–684. doi: 10.3233/ATDE190177.

[18] M. T. Khatun, K. Hiekata, Y. Takahashi, and I. Okada, "Design and management of software development projects under rework uncertainty: a study using system dynamics," *J Decis Syst*, vol. 00, no. 00, pp. 1–24, 2022, doi: 10.1080/12460125.2021.2023257.

[19] J. S. Aguilar-Ruiz, J. C. Riquelme, D. Rodríguez, and I. Ramos, "Generation of management rules through system dynamics and evolutionary computation," *Lecture Notes in Computer Science*, vol. 2559, pp. 615–628, 2002, doi: 10.1007/3-540-36209-6_50.

[20] E. Ferreira Franco, K. Hirama, and M. M. Carvalho, "Applying system dynamics approach in software and information system

projects: A mapping study," *Inf Softw Technol*, vol. 93, pp. 58–73, 2018, doi: 10.1016/j.infsof.2017.08.013.

[21] L. Machuca-Villegas, G. P. Gasca-Hurtado, S. Morillo Puente, and L. M. Restrepo-Tamayo, "Factores sociales y humanos que influyen en la productividad del desarrollo de software: Medición de la percepción," *Revista Ibérica de Sistemas e Tecnologias de Informação*, vol. E41, no. 02/2021, pp. 488–502, 2021.

[22] O. Filipova and R. Vilão, *Software Development From A to Z*. 2018. doi: 10.1007/978-1-4842-3945-2.

[23] A. P. Murray, *The Complete Software Project Manager*. New Jersey: Wiley, 2016.

[24] B. Baj, Ed., *Wiley Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, 2008.

[25] J. F. Dooley, *Software Development, Design and Coding*. Galesburg, Illinois, USA: Apress, 2017. doi: 10.1007/978-1-4842-3153-1.

[26] R. Pressman, *Ingeniería del Software: Un Enfoque Práctico*, Séptima ed. Mc Graw Hill, 2010.

[27] B. Hughes and M. Cotterell, *Software Proiect Management*, Fifth Edit. McGraw Hill Education, 2009.

[28] A. Villafiorita, *Introduction to Software Project Management*. CRC Press, 2014.

[29] M. Jørgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 33–53, 2007, doi: 10.1109/TSE.2007.256943.

[30] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf Softw Technol*, vol. 54, no. 1, pp. 41–59, 2012, doi: 10.1016/j.infsof.2011.09.002.

[31] N. Govil and A. Sharma, "Estimation of cost and development effort in Scrum-based software projects considering dimensional success factors," *Advances in Engineering Software*, vol. 172, no. April, p. 103209, 2022, doi: 10.1016/j.advengsoft.2022.103209.

[32] L. Machuca-Villegas, G. P. Gasca-hurtado, S. Morillo, and L. M. Restrepo-Tamayo, "Perceptions of the human and social factors that influence the productivity of software development teams in Colombia: A statistical analysis," *J Syst Softw*, vol. 192, 2022, doi: 10.1016/j.jss.2022.111408.

[33] V. Spiezia, "Jobs and skills in the digital economy," 2017. doi: 10.1787/de5b1ac4-en.

[34] Infosys, "Talent Radar How the best companies get the skills they need to thrive in the digital era," 2019.

[35] L. M. Restrepo-Tamayo and G. P. Gasca-Hurtado, "Non-technical Factors in Software Engineering Within the Context of Industry 4.0," in *New Perspectives in Software Engineering. Studies in Computational Intelligence*, J. Mejía, M. Muñoz, A. Rocha, Y. Hernández Pérez, and H. Avila-George, Eds., Springer, Cham, 2024, pp. 89–103. doi: 10.1007/978-3-031-50590-4_6.

[36] K. G. Cooper, "The Rework Cycle: Why Projects are Mismanaged," *Project Management Action*, vol. 24, no. 1, pp. 17–21, 1993.

[37] S. Bayer and D. Gann, "Balancing work: Bidding strategies and workload dynamics in a project-based professional service organisation," *Syst Dyn Rev*, vol. 22, no. 3, pp. 185–211, 2006, doi: 10.1002/sdr.344.

[38] H. Rahmandad and K. Hu, "Modeling the rework cycle: Capturing multiple defects per task," *Syst Dyn Rev*, vol. 26, no. 4, pp. 291–315, 2010, doi: 10.1002/sdr.435.

[39] Y. Jalili and D. N. Ford, "Quantifying the impacts of rework, schedule pressure, and ripple effect loops on project schedule performance," *Syst Dyn Rev*, vol. 32, no. 1, pp. 82–96, 2016, doi: 10.1002/sdr.1551.

[40] S. H. Lee and F. Peña-Mora, "Understanding and managing iterative error and change cycles in construction," *Syst Dyn Rev*, vol. 23, no. 1, pp. 35–60, 2007, doi: 10.1002/sdr.359.

[41] B. D. Owens, N. G. Leveson, and J. A. Hoffman, "Procedure rework: A dynamic process with implications for the 'rework cycle' and 'disaster dynamics,'" *Syst Dyn Rev*, vol. 27, no. 3, pp. 244–269, 2011, doi: 10.1002/sdr.464.

[42] J. M. Lyneis, K. G. Cooper, and S. A. Els, "Strategic management of complex projects: A case study using system dynamics," *Syst Dyn Rev*, vol. 17, no. 3, pp. 237–260, 2001, doi: 10.1002/sdr.213.

[43] L. F. Luna-Reyes, L. J. Black, A. M. Cresswell, and T. A. Pardo, "Knowledge sharing and trust in collaborative requirements analysis," *Syst Dyn Rev*, vol. 24, no. 3, pp. 265–297, 2008, doi: 10.1002/sdr.404.

[44] T. Walworth, M. Yearworth, L. Shrieves, and H. Sillitto, "Estimating Project Performance through a System Dynamics Learning Model," *Systems Engineering*, vol. 19, no. 4, pp. 334–350, Jul. 2016, doi: 10.1002/sys.21349.

[45] F. Ahmed, L. F. Capretz, S. Bouktif, and P. Campbell, "Soft skills and software development: A reflection from software industry," *International Journal of Information Processing and Management*, vol. 4, no. 3, pp. 171–191, 2013, doi: 10.4156/ijipm.vol4.issue3.17.

[46] Alexandre. Rodrigues and T. Williams, "System dynamics in software project management: Towards the development of a formal integrated framework," *European Journal of Information Systems*, vol. 6, no. 1, pp. 51–66, 1997, doi: 10.1057/palgrave.ejis.3000256.

[47] C. Caulfield, G. Kohli, and S. P. Maj, "Sociology in software engineering," in *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition*, 2004, pp. 12685–12697. doi: 10.18260/1-2--14027.

[48] L. M. Restrepo-Tamayo, G. P. Gasca-Hurtado, and J. Valencia-Calvo, "Characterizing Social and Human Factors in Software Development Team Productivity: A System Dynamics Approach," *IEEE Access*, vol. 12, no. 1, pp. 59739–59755, 2024, doi: 10.1109/ACCESS.2024.3388505.

[49] M. Schaffernicht, *Indagación de situaciones dinámicas mediante la dinámica de sistemas. Tomo 1: Fundamentos*. 2006.

[50] J. Aracil, *Dinámica de Sistemas*, 4th ed. Madrid, España: Isdefe, 2016.

[51] E. Pruyt, *System dynamics models for big issues: Triple Jump towards Real-World Complexity*. The Netherlands: TU Delft Library, 2013. doi: 10.1007/978-1-84882-809-4_2.

[52] C. Cadenas and W. Guaita, *Dinámica de sistemas Una metodología para la construcción de modelos de toma de decisiones en sectores agroindustriales*. Bogotá, Colombia: Editorial Politécnico Grancolombiano, 2020.

[53] P. M. Senge, *La quinta disciplina: el arte y la práctica de la organización abierta al aprendizaje*, 2° ed. Buenos Aires: Granica, 2010.

[54] A. Sarmiento-Vásquez and E. López-Sandoval, "Una comparación cualitativa de la dinámica de sistemas, la simulación de eventos discretos y la simulación basada en agentes," *Ingeniería Industrial*, no. 35, p. 27, 2017, doi: 10.26439/ing.ind2017.n035.1789.

[55] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. Mc Graw Hill, 2015. doi: 10.2307/2288169.

[56] J. W. Forrester, *Industrial Dynamics*. The M.I.T Press, 1961.

[57] L. G. Birta and G. Arbez, *Modelling and Simulation Exploring Dynamic System Behaviour*, Third Edit. Springer, 2019. doi: 10.1201/b16599-4.

[58] Y. Barlas, "Model Validation in System Dynamics," in *Proceedings of the 1994 International System Dynamics Conference*, 1994, pp. 1–10.

[59] P. Sudhakar, A. Farooq, and S. Patnaik, "Measuring productivity of software development teams," *Serbian Journal of Management*, vol. 7, no. 1, pp. 65–75, 2012, doi: 10.5937/sjm1201065s.

[60] D. Rodríguez, M. A. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012, doi: 10.1016/j.jss.2011.09.009.

[61] L. Rising and N. Janoff, "The Scrum Software Development Process for Small Teams," *IEEE Softw*, no. July/August, pp. 26–32, 2000, doi: 10.1109/52.854065.

[62] B. W. Tuckman, "Developmental sequence in small groups," *Psychol Bull*, vol. 63, no. 6, pp. 384–399, 1965, doi: 10.1037/h0022100.

[63] L. Gren, R. Torkar, and R. Feldt, "Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies," *Journal of Systems and Software*, vol. 124, pp. 104–119, 2017, doi: https://doi.org/10.1016/j.jss.2016.11.024.

[64] D. Strode, T. Dingsøyr, and Y. Lindsjorn, "A teamwork effectiveness model for agile software development," *Empir Softw Eng*, vol. 27, no. 2, pp. 1–50, 2022, doi: 10.1007/s10664-021-10115-0.

[65] B. W. Tuckman and M. A. C. Jensen, "Stages of Small-Group Development Revisited," *Group & Organization Studies*, vol. 2, no. 4, pp. 419–427, 1977.

[66] M. Rossi, *Human Resource Design Steering Human-centered Innovation within Organisations*. Springer, 2021.

[67] S. Eybers and M. J. Hattingh, "The last straw: Teaching project team dynamics to third-year students," in *Communications in Computer and Information Science*, Springer International Publishing, 2019, pp. 237–252. doi: 10.1007/978-3-030-05813-5_16.

[68] K. M. Lui and K. C. C. Chan, "Pair programming productivity: Novice-novice vs. expert-expert," *International Journal of Human Computer Studies*, vol. 64, no. 9, pp. 915–925, 2006, doi: 10.1016/j.ijhcs.2006.04.010.

[69] M. Edson and G. Metcalf, "Adaptive Capacity in Project Teams," in *Proceedings of the 58th Annual Meeting of the Isss 2014 United States*, 2014.

[70] J. F. Super, "Building innovative teams: Leadership strategies across the various stages of team development," *Bus Horiz*, vol. 63, no. 4, pp. 553–563, 2020, doi: 10.1016/j.bushor.2020.04.001.

[71] K. Nicolopoulou, M. Koštomaj, and A. Campos, "How to address group dynamics in virtual worlds," *AI Soc*, vol. 20, no. 3, pp. 351–371, 2006, doi: 10.1007/s00146-005-0027-0.

[72] M. Niever, N. Trefz, J. Heimicke, C. Hahn, and A. Albers, "Situation- and need-based method recommendation for coaching agile development teams," in *Procedia CIRP*, Elsevier B.V., 2021, pp. 512–517. doi: 10.1016/j.procir.2021.05.112.

[73] A. L. Mesquida, J. Karać, M. Jovanović, and A. Mas, "A game toolbox for process improvement in agile teams," in *Communications in Computer and Information Science*, 2017, pp. 302–309. doi: 10.1007/978-3-319-64218-5_25.

[74] S. A. Wheelan and J. M. Hochberger, "Validation Studies of the Group Development Questionnaire," *Small Group Res*, vol. 27, no. 1, pp. 143–170, 1996, doi: https://doi.org/10.1177/1046496496271007.

[75] L. Gren, C. Jaconsson, N. Rydbo, and P. Lenberg, "The Group Development Questionnaire Short (GDQS) Scales: Tiny-Yet-Effective Measures of Team/Small Group Development," 2020.

[76] D. L. Miller, "The stages of group development: A retrospective study of dynamic team processes," *Canadian Journal of Administrative Sciences*, vol. 20, no. 2, pp. 121–134, 2003, doi: 10.1111/j.1936-4490.2003.tb00698.x.

[77] R. E. Levasseur, "People skills: Optimizing team development and performance," *Interfaces (Providence)*, vol. 41, no. 2, pp. 204–208, 2011, doi: 10.1287/inte.1100.0519.

[78] Project Management Institute, *A guide to the project management body of knowledge (PMBOK guide)*. 2017.

[79] R. Hughes, "Streamlining Project Management," in *Agile Data Warehousing Project Management*, 2013, pp. 81–113. doi: 10.1016/b978-0-12-396463-2.00003-x.

[80] S. Wolpers, "The Scrum Anti-Patterns Guide," 2023.

[81] A. N. B. M. Nasir, D. F. Ali, M. K. bin Noordin, and M. S. Bin Nordin, "Technical skills and non-technical skills: predefinition concept Mohd Safarin Bin NORDIN," in *Proceedings of the IETEC'11 Conference, Kuala Lumpur, Malaysia*, 2011, pp. 1–17.

[82] M. Hoegl and H. G. Gemuenden, "Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence," *Organization Science*, vol. 12, no. 4, pp. 435–449, 2001, doi: 10.1287/orsc.12.4.435.10635.

[83] G. Marsicano, F. Q. B. Silva, C. B. Seaman, and B. G. Adaid-castro, "The Teamwork Process Antecedents (TPA) questionnaire: developing and validating a comprehensive measure for assessing antecedents of teamwork process quality," *Empir Softw Eng*, vol. 25, pp. 3928–3976, 2020.

[84] P. Hsia, C. Hsu, and D. C. Kung, "Brooks' Law Revisited: A System Dynamics Approach," in *Twenty-Third Annual International Computer Software and Applications Conference*, 1999, pp. 370–375.

[85] M. Wu and H. Yan, "Simulation in software engineering with system dynamics: A case study," *Journal of Software*, vol. 4, no. 10, pp. 1127–1135, 2009, doi: 10.4304/jsw.4.10.1127-1135.

[86] T. K. Abdel-Hamid, "The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach," *IEEE Transactions on Software Engineering*, vol. 15, no. 2, pp. 109–119, 1989, doi: 10.1109/32.21738.

[87] S. Wood, G. Michaelides, and C. Thomson, "Successful extreme programming: Fidelity to the methodology or good teamworking?," *Inf Softw Technol*, vol. 55, no. 4, pp. 660–672, 2013, doi: 10.1016/j.infsof.2012.10.002.

[88] E. Salas, D. E. Sims, and C. Shawn Burke, "Is there A 'big five' in teamwork?," *Small Group Res*, vol. 36, no. 5, pp. 555–599, 2005, doi: 10.1177/1046496405277134.

[89] M. Alefari, M. Almanei, and K. Salonitis, "A system dynamics model of employees' performance," *Sustainability*, vol. 12, no. 16, 2020, doi: 10.3390/su12166511.

[90] P. Hersey, K. H. Blanchard, and W. E. Natemeyer, "Situational Leadership, Perception, and the Impact of Power," *https://doi-org.ezproxy.unal.edu.co/10.1177/105960117900400404*, vol. 4, no. 4, pp. 418–428, Dec. 1979, doi: 10.1177/105960117900400404.

[91] P. Ciancarini, M. Missiroli, and S. Zani, *Empirical Evaluation of Agile Teamwork*, vol. 1439 CCIS. Springer International Publishing, 2021. doi: 10.1007/978-3-030-85347-1_11.

[92] C. Laporte and A. April, *Software Quality Assurance*. 2018.

[93] B. Nagaria and T. Hall, "How Software Developers Mitigate their Errors when Developing Code," *IEEE Transactions on Software Engineering*, vol. 14, no. 8, 2020, doi: 10.1109/TSE.2020.3040554.

[94] S. G. Dastidar, "Model of distributed software development using system dynamics," 2015.

[95] ISACA, "CMMI Technical Report: Performance Results," 2023.

[96] R. V. O'Connor and C. Y. Laporte, "The Evolution of the ISO / IEC 29110 Set of Standards and Guides," *International Journal of Information Technologies and Systems Approach*, vol. 10, no. 1, 2017, doi: 10.4018/IJITSA.2017010101.

[97] ISO, "ISO/IEC TR 29110-5-1-1:2012 Software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 5-1-1: Management and engineering guide: Generic profile group: Entry profile," 2012.

[98] C. Y. Laporte, R. V. O. Connor, and L. H. García Paucar, "The Implementation of ISO / IEC 29110 Software Engineering Standards and Guides in Very Small Entities," in *Evaluation of Novel Approaches to Software Engineering. ENASE 2015. Communications in Computer and Information Science*, L. A. Maciaszek and J. Filipe, Eds., Springer, Cham, 2016, pp. 162–179. doi: 10.1007/978-3-319-30243-0.

[99] J. F. Defranco and P. A. Laplante, "Review and analysis of software development team communication research," *IEEE Trans Prof Commun*, vol. 60, no. 2, pp. 165–182, 2017, doi: 10.1109/TPC.2017.2656626.

[100] D. A. Bonebright, "40 years of storming: A historical review of tuckman's model of small group development," *Human Resource Development International*, vol. 13, no. 1, pp. 111–120, 2010, doi: 10.1080/13678861003589099.

[101] C. Rosen, "Team Management," in *Guide to Software Systems Development*, 2020, ch. 4, pp. 57–77. doi: 10.1007/978-3-030-39730-2.

[102] T. C. Pfutzenreuter, E. P. de Lima, and J. R. Frega, "Building High Performance Teams," in *Communications in Computer and Information Science*, Springer International Publishing, 2021, pp. 251–264. doi: 10.1007/978-3-030-76307-7_19.

[103] M. Kokkoniemi and V. Isomöttönen, "A systematic mapping study on group work research in computing education projects,"

Journal of Systems and Software, vol. 204, p. 111795, 2023, doi: 10.1016/j.jss.2023.111795.

[104] A. Jones, "The Tuckman's model implementation, effect, and analysis & the new development of Jones LSI model on a small group," *J Manage*, vol. 6, no. 4, pp. 23–28, 2019, doi: 10.34218/jom.6.4.2019.005.

[105] P. Kaygan, "From forming to performing: team development for enhancing interdisciplinary collaboration between design and engineering students using design thinking," *Int J Technol Des Educ*, vol. 33, no. 2, pp. 457–478, 2023, doi: 10.1007/s10798-022-09736-3.

[106] S. V. Spiegler, D. Graziotin, C. Heinecke, and S. Wagner, "A Quantitative Exploration of the 9-Factor Theory: Distribution of Leadership Roles Between Scrum Master and Agile Team," in *Lecture Notes in Business Information Processing*, Springer International Publishing, 2020, pp. 162–177. doi: 10.1007/978-3-030-49392-9_11.

[107] L. Gren and M. Lindman, "What an Agile Leader Does: The Group Dynamics Perspective," in *Lecture Notes in Business Information Processing*, Springer International Publishing, 2020, pp. 178–194. doi: 10.1007/978-3-030-49392-9_12.

[108] T. Myklebust and T. Stålhane, "Agile Practices," in *Functional Safety and Proof of Compliance*, 2021, pp. 25–58. doi: 10.1007/978-3-030-86152-0_2.

[109] C. A. Schriesheim, J. B. Wu, and T. A. Scandura, "A meso measure? Examination of the levels of analysis of the Multifactor Leadership Questionnaire (MLQ)," *Leadership Quarterly*, vol. 20, no. 4, pp. 604–616, 2009, doi: 10.1016/j.leaqua.2009.04.005.

[110] M. J. Tejeda, T. A. Scandura, and R. Pillai, "The MLQ revisited Psychometric properties and recommendations," *Leadersh Q*, vol. 12, pp. 31–52, 2001.

[111] B. Bala, F. Mohamed, and K. Mohd, *System dynamics. Modelling and Simulation*. Springer Nature, 2017. doi: 10.4324/9780203112694-14.

[112] L. Gren and M. Shepperd, "Problem reports and team maturity in agile automotive software development," *Proceedings - 15th International Conference on Cooperative and Human Aspects of Software Engineering, CHASE 2022*, pp. 41–45, 2022, doi: 10.1145/3528579.3529173.

[113] T. C. Pfutzenreuter, E. P. de Lima, and J. R. Frega, "High performance teams: an investigation of the effect on self-management towards performance," *Production*, vol. 31, no. 2019, pp. 1–14, 2021, doi: 10.1590/0103-6513.20210053.

[114] A. Zirar, N. Muhammad, A. Upadhyay, A. Kumar, and J. A. Garza-Reyes, "Exploring lean team development from the Tuckman's model perspective," *Production Planning and Control*, vol. 0, no. 0, pp. 1–22, 2023, doi: 10.1080/09537287.2023.2275693.

[115] A. P. Galvão Scheidegger, T. Fernandes Pereira, M. L. Moura de Oliveira, A. Banerjee, and J. A. Barra Montevechi, "An introductory guide for hybrid simulation modelers on the primary simulation methods in industrial engineering identified through a systematic review of the literature," *Comput Ind Eng*, vol. 124, pp. 474–492, 2018, doi: 10.1016/j.cie.2018.07.046.

[116] A. Rodrigues and J. Bowers, "System dynamics in project management: A comparative analysis with traditional methods," *Syst Dyn Rev*, vol. 12, no. 2, pp. 121–139, 1996, doi: 10.1002/(sici)1099-1727(199622)12:2<121::aid-sdr99>3.0.co;2-x.

[117] D. Pfahl, N. Koval, and G. Ruhe, "An experiment for evaluating the effectiveness of using a system dynamics simulation model in software project management education," in *International Software Metrics Symposium, Proceedings*, 2001, pp. 97–109. doi: 10.1109/metric.2001.915519.

[118] Alexandre. Rodrigues and T. Williams, "System dynamics in project management: Assessing the impacts of client behaviour on project performance," *Journal of the Operational Research Society*, vol. 49, no. 1, pp. 2–15, 1998, doi: 10.1057/palgrave.jors.2600490.

[119] P. C. Das and U. R. Dhar, "A System Dynamics Approach towards Software Development Project - A Case Study," *International Advanced Research Journal in Science, Engineering and Technology*, vol. 3, no. 4, pp. 120–123, 2016, doi: 10.17148/IARJSET.2016.3426.

[120] C. Andersson, L. Karlsson, J. Nedstam, M. Host, and B. I. Nilsson, "Understanding software processes through system dynamics simulation: A case study," in *Proceedings - 9th Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, ECBS 2002*, 2002, pp. 41–48. doi: 10.1109/ECBS.2002.999821.

[121] M. T. I. Trammell, S. E. Madnick, and A. Moulton, "Using System Dynamics to Analyze the Effect of Funding Fluctuation on Software Development," *EMJ - Engineering Management Journal*, 2013, doi: 10.1080/10429247.2016.1155390.

[122] D. Pfahl, A. Al-Emran, and G. Ruhe, "A System Dynamics Simulation Model for Analyzing the Stability of Software Release Plans," *Software Process: Improvement and Practice*, vol. 12, pp. 475–490, 2007, doi: 10.1002/spip.

[123] H. Rahmandad and D. M. Weiss, "Dynamics of concurrent software development," *Syst Dyn Rev*, vol. 25, no. 3, pp. 224–249, 2009, doi: 10.1002/sdr.425.

[124] J. D. Tvedt and J. S. Collofello, "Evaluating the effectiveness of process improvements on software development cycle time via systems dynamics modeling," in *Proceedings - IEEE Computer Society's International Computer Software & Applications Conference*, 1995, pp. 318–325. doi: 10.1109/cmpsac.1995.524796.

[125] R. J. Madachy, "System dynamics modeling of an inspection-based process," in *Proceedings - International Conference on Software Engineering*, 1996, pp. 376–386. doi: 10.1109/icse.1996.493432.

[126] Z. Sahaf, V. Garousi, D. Pfahl, R. Irving, and Y. Amannejad, "When to automate software testing? decision support based on system dynamics: An industrial case study," in *ACM International Conference Proceeding Series*, 2014, pp. 149–158. doi: 10.1145/2600821.2600832.

[127] C. Caulfield and S. P. Maj, "A case for system dynamics," *Global Journal of Engineering Education*, vol. 6, no. 1, pp. 2793–2798, 2002.

[128] I. Fatema and K. Sakib, "Using Qualitative System Dynamics in the Development of an Agile Teamwork Productivity Model," *International Journal on Advances in Software*, vol. 11, no. 1 & 2, pp. 170–185, 2018.

[129] I. Fatema, "Agile Software Develoment Teamwork Productivity - A System Dynamcs Approach to Analyse the Productivity Influence Factors," University of Dhaka, 2019.

[130] F. Stallinger and P. Grünbacher, "System dynamics modelling and simulation of collaborative requirements engineering," *Journal of Systems and Software*, vol. 59, no. 3, pp. 311–321, 2001, doi: 10.1016/S0164-1212(01)00071-1.

[131] M. Alshammri, "Simulation modelling of human aspects in software project environment," in *ACM International Conference Proceeding Series*, 2015, pp. 145–146. doi: 10.1145/2811681.2824995.

[132] J. Collofello, D. Houston, I. Rus, A. Chauhan, D. M. Sycamore, and D. Smith-Daniels, "System dynamics software process simulator for staffing policies decision support," in *Proceedings of the Hawaii International Conference on System Sciences*, 1998, pp. 103–111. doi: 10.1109/hicss.1998.654764.

[133] T. K. Abdel-Hamid and S. Madnick, "Software productivity: Potential, actual, and perceived," *Syst Dyn Rev*, vol. 5, no. 2, pp. 93–113, 1989, doi: 10.1002/sdr.4260050202.

[134] B. Liu, H. Zhang, L. Dong, Z. Wang, and S. Li, "Metrics for software process simulation modeling," *Journal of Software: Evolution and Process*, no. August 2023, pp. 1–34, 2024, doi: 10.1002/smr.2676.

[135] C. Z. Kirilo *et al.*, "Organizational climate assessment using the paraconsistent decision method," *Procedia Comput Sci*, vol. 131, pp. 608–618, 2018, doi: 10.1016/j.procs.2018.04.303.

[136] J. A. García-García, J. G. Enríquez, M. Ruiz, C. Arévalo, and A. Jiménez-Ramírez, "Software Process Simulation Modeling: Systematic literature review," *Comput Stand Interfaces*, vol. 70, no. January, p. 103425, 2020, doi: 10.1016/j.csi.2020.103425.

[137] N. Tiruvengadam, A. Elizondo-Noriega, D. Gemes-Castorena, and M. G. Beruvides, "Opportunities for System Dynamics Implementation in Project Management Evaluation," in *PICMET 2022 - Portland International Conference on Management of Engineering and Technology: Technology Management and Leadership in Digital Transformation - Looking Ahead to Post-COVID Era, Proceedings*, 2022. doi: 10.23919/PICMET53225.2022.9882860.

[138] H. Zhang, B. Kitchenham, and R. Jeffery, "A framework for adopting software process simulation in CMMI organizations," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007, pp. 320–331. doi: 10.1007/978-3-540-72426-1_27.

[139] V. D. Ramdoo and O. Gukhool, "Applying System Dynamics to Software Quality Management," *International Journal of Emerging Research in Management &Technology*, vol. 6, no. 2, pp. 28–43, 2017, doi: 10.23956/ijermt/v6n2/108.

[140] *ISO/IEC FDIS 29110-5-1-2 Systems and software engineering — Life cycle profiles for very small entities (VSEs)*. 2011.

[141] A. McLucas, "Incorporating soft variables into system dynamics models: a suggested method and basis for ongoing research," *2003 System Dynamics Conference papers*, 2003.

[142] United-Nations, "Goal 9: Build resilient infrastructure, promote sustainable industrialization and foster innovation," https://www.un.org/sustainabledevelopment/infrastructure-industrialization/.

[143] CompTIA, "IT Industry Outlook 2022 Return to Strategy," 2022.

[144] M. Akşit, "The Role of Computer Science and Software Technology in Organizing Universities for Industry 4.0 and beyond," in *Proceedings of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018*, 2018, pp. 5–11. doi: 10.15439/2018F002.

[145] O. Bongomin, G. Gilibrays Ocen, E. Oyondi Nganyi, A. Musinguzi, and T. Omara, "Exponential Disruptive Technologies and the Required Skills of Industry 4.0," *Journal of Engineering (United Kingdom)*, vol. 2020, 2020, doi: 10.1155/2020/4280156.

LUZ MARCELA RESTREPO-TAMAYO received her degree in Industrial Engineering in 2009 from the University of Antioquia, and in 2014, she graduated with a Master of Statistics from the National University of Colombia. She is currently a PhD student in Engineering at the University of Medellin. His line of research is related to the improvement of productivity and process quality through data analysis. She worked in manufacturing companies and is currently an assistant professor in the Faculty of Engineering at the University of Medellin. She is currently working on the analysis of the productivity of software development teams, considering non-technical factors based on system dynamics.

GLORIA PIEDAD GASCA-HURTADO is a professor at the University of Medellín in the Faculty of Engineering. In addition, she has a background in systems engineering and a specialization in Systems Auditing. Her doctorate was completed at the Polytechnic University of Madrid, Spain, in the Department of Languages, Computer Systems and Software Engineering of the Faculty of Informatics. Her research is oriented towards educational innovation and digital transformation in engineering education, with a specific focus on gamification and enabling technologies of the Fourth Industrial Revolution. Additionally, her areas of interest include training in promising practices for software development in teams and agile methodologies, the treatment of social and human factors in software engineering, and the productivity of software development teams. At the curricular level, the research professor has experience in leading and participating in curricular innovation processes that incorporate active learning strategies.

JOHNNY VALENCIA-CALVO received his degree in Electronics Engineering in 2010, an M.Sc. in Industrial Automation from Universidad Nacional de Colombia in 2012, and his Ph.D. in 2016 in Computer Sciences and Decisions. His research experience has led him to become involved in topics related to dynamic analysis, non-linear dynamics, and system dynamics, focusing on modeling, simulation, and applications of mathematics in engineering. Professional in computer science and decision, electronics engineer from the National University of Colombia, and Master in Industrial Automation. He worked on the analysis, design, and modeling of a national electricity market to formulate new political strategies. Expert in systems dynamics and complex systems modeling.