Use of Feedback Loop Control Theory in Software Project Productivity Control: A Simulation Study*

Francisco Valdés Souto[1][0000-0001-6736-0666] and Jorge Valeriano Assem[2]

[1] National Autonomous University of Mexico
Science Faculty
CDMX, Mexico City, Mexico
fvaldes@ciencias.unam.mx
[2] Spingere SA de CV, CDMX, Mexico City, Mexico
jorge.valeriano@spingere.com.mx

Abstract. Several studies have shown that one of the main problems in software development is control of the projects to accomplish the specific constraints in terms of cost, time, and scope.

Several performance evaluation techniques have been developed from the project management perspective. However, these techniques are not often utilized in practice for software projects, and when they are utilized, they provide information to project managers that are analyzed based on their own experience defining actions that are challenging to replicate in distinct contexts.

In this paper, the feedback loop control theory, very often used in other disciplines to ensure that a certain operation of a specific process accomplishes particular behavior under certain conditions and constraints, is applied to a software development project simulation, modeling the team capacity in terms of productivity as a plant.

Keywords: COSMIC ISO 19761; Productivity control; Software functional size; Feedback loop control.

1 Introduction

Any functional size measurement standard, such as the COSMIC ISO 19761 standard [1], provides us with the functional units of a piece of software that, in combination with resource variables such as effort or cost, enable us to define productivity models that quantitatively represent the way in which an organization or team works. This essential information is derived from past projects and employed to estimate future projects.

One of the most significant and fundamental problems recognized in software development is the control of project constraints such as scope, time, and cost, being

identified as the main success factors in projects [2]. From the project management perspective have been developed some performance evaluation techniques, also known as control systems [3, 4], have been developed, helping to manage the performance of the project in terms of the main constraints: scope, cost, and time, some examples of control systems include the following: EVM (earned value management) [5] for cost, ES (earned schedule management) [5] for time or ESM (earned scope management) [5] for scope, however, these techniques provide information to project managers (PM) who analyze it and define actions subject to experience, rendering it difficult to replicate the results among projects, teams, and institutions.

In other engineering fields, the feedback loop control theory is used to ensure that the operation of a specific process, such as, mechanical, electrical, electronic, achieves a specific behavior under certain conditions and constraints in a consistent manner.

The application of this feedback loop control theory to software development projects has not found, to our knowledge, in the literature. However, there are standardized quantity measures that can utilized to measure, as in other disciplines, i.e., quantity, resources, and productivity.

This article proposes the use of feedback loop control theory in software project productivity control, bound to the environment of a simulation, allowing to observe the behavior in a fully controlled environment. Thus employing conditions gathered from a real situation are considered for simulation purposes, aiming to identify whether the theory proposed could be applied to real projects.

A Simulation Study of the application of the feedback loop control theory considering the productivity of a software development team represents the simulation of a plant. The goal is to simulate team behavior under certain constraints applying a feedback loop control system, seeking to find whether this could be applied in a real projects, enabling management improvement for software projects in an automatic way, providing the possibility of making behavior predictions or dynamic and adaptable management tools for the Project Managers.

The article is organized in the following manner: Section 2 explains the Background utilized, been the main topics described are the productivity models for software using COSMIC (ISO/IEC 19761) as a quantifiable unit of functional size, and the feedback loop control systems; Section 3 presents the simulation study developed by applying a feedback loop control system to a software team for a specific project request, and finally, the analysis of results and the conclusions of the Simulation study are presented.

2 Background

2.1 Functional size measurement with the COSMIC method.

It is noteworthy that, in this article, we are solely interested in the functional part of the software, which can be dimensioned, to date, with 5 standards (IFPUG, NESMA, FISMA, MKII, and COSMIC), among which the first four are considered from first generation and only one is recognized from the second generation (COSMIC).

In this paper, the use of COSMIC was selected because the method offers a standardized method of measuring the functional size of software in several domains, i.e., as

“business application” (or “MIS”) software, “real-time” software, “infrastructure” software, and some types of scientific software and engineering software. [1]

The COSMIC measurement method consists of applying a set of models, principles, rules, and processes to the Functional Requirements of Users (FUR) of a given software application. The result is a numerical "value of a quantity" (as defined in ISO), representing the functional size of the software. The functional size unit of the COSMIC method is 1COSMIC Function Point (CFP), which is equivalent to a movement of a group of data according to the rules of the method. Detailed information about COSMIC could be accessed at www.cosmic-sizing.org

2.2 Productivity models in software development

Mathematical models are typically constructed using data from completed projects [6]; due to this, the majority of the so-called estimation models are productivity models, representing the productivity of past projects already completed, and reflecting how a specific team or company works [7]. According to Abran [6], some of the benefits of past-project mathematical productivity models are the following:

- The efficiency of these models can be analyzed and described.
- They can be used to estimate future projects.
- Whenever the same information is entered into the estimation model, the same result will be obtained.

It is possible to say that "the productivity model represents the relationship between the two variables, that is, between the independent variable (the size of the software) and the dependent variable (the effort or cost of the finished project)" [6]. To date, only functional size can be measured in a standardized manner, in such way we can only have an independent variable that can be measured adequately, some dependent variables could be considered like effort or cost.

There are various methods to find the best line that describes the behavior between independent and dependent variables in statistics. One of the most widely used of these is the least-squares adjustment method. The latter, is a numerical analysis procedure in which, given a set of data (ordered pairs), an attempt is made to determine the continuous function that best approximates the data (regression line or line of best fit), providing a simple visual demonstration of the relationship between the independent and dependent variables.

In its purest form, it seeks to minimize the sum of squares of the ordered differences (called residuals) between the points generated by the function and the corresponding data. Once the productivity model is established by the least-squares regression method, it is necessary to evaluate the goodness of the design. It is necessary to have a measure that we can take as a reference to weigh the performance of our statistical exercise. The determination coefficient (R^2) represents the proportion of the variance (dispersion of the data around the average) of the response variable that is explained by the model; this is given by a percentage, that is, it falls between 0 and 1, this latter value is the ideal since it implies that we are explaining the dependent variance at 100% [6] [7].

The relation between functional size and effort or cost is fundamental because it is related to productivity; from a software point of view, productivity is defined as "the

ratio of the number of products delivered by a process to the number of inputs" [6]. The general formula of productivity can be written as Productivity = Q / R, where Q refers to the quantity produced (which can be measured by functional size [CFP]), and R refers to the resources (i.e., the effort that can be measured in Work-Hours [WH]) utilized to produce Q (Productivity= (Functional size)⁄Effort). It is possible to affirm, then, that if there is no way to measure the amount of software produced, "PRODUCTIVITY CANNOT BE MEASURED".

The previous productivity definition is used in this article and is focused only on the part of the Functional User Requirements of the software (FUR), dimensioned with COSMIC; we omit the part concerning the Non-Functional Requirements (NFR). However, different definitions of productivity could exists depending on the parts of the software considered (FUR, NFR, Quality, etc.) as proposed by Buglione [22].

It is important to note that, in this article, we focused on the product scope as understood by the PMBOK [2] and on what can be measured in a standardized manner. This due to that, there are several studies in the literature on the correlation of functional size with effort; thus, NFR topics are not considered.

2.3 Performance evaluation techniques from the project management perspective

Since 1987, EVM has been evolving; some improvements at various levels have arisen, for example, improvement in the formulas or new approaches derived, such as ES or ESM [5]. Earned Value Management (EVM) is the most recognized, most accepted, and best-known performance evaluation technique to assess project performance objectively, providing valuable insights [8, 9] from the general project management perspective. However, "software project managers rarely apply this powerful technique during the tracking of the software projects" [10]. An argument provided by Ferle [9] is that software projects are not trivial and states that the author forecasted that the estimating the efforts of tasks in software projects is extremely difficult, exhibiting a notable project failure rate. Independently of the performance evaluation techniques utilized, these techniques will provide information to project managers; the latter analyze the information and define actions subject to their own experience, rendering it difficult to replicate the results among projects, teams, and institutions.

2.4 Control theory

Control theory deals with the control of continuously operating dynamical systems in engineered processes and machines. The objective is to develop a control model for controlling such systems using a control action in an optimal manner without delaying or overshooting and ensuring control stability. Control theory is a subfield of mathematics, computer science, and control engineering. [11]

Feedback control systems.

The feedback control system conforms with a controller that monitors the controlled process variable (PV-System output) and compares it with the Set Point or Reference (SP-Reference). The difference between the actual and desired value of the process variable, termed the Measurement error (SP-PV error), is applied as feedback to generate a control action to take the controlled process variable to the same value as the SP. Other aspects that are also studied include controllability and observability.

This is the basis for the advanced type of automation that revolutionized manufacturing, aircraft, communications, and other industries. The feedback control system involves collecting measurements using a sensor and making intended adjustments to maintain the measured variable within an stablished range utilizing a "final control element", such as a control valve. Fundamentally, there are two types of control loops:

1. Open-loop control, and
2. Closed-loop (feedback) control.

In open-loop control, the controllers control action is independent of the "process output" (or PV). An excellent example of this is a central heating boiler controlled only by a timer, so that heat is applied for a constant time, regardless of the temperature of the building. The control action is the timed on/off switching of the boiler. The process variable is the building temperature, but neither is linked.

In closed-loop control, the control action from the controller is dependent on feedback from the process in the form of the value of the Process Variable (PV). In the boiler analogy case, a closed-loop would include a thermostat to compare the building temperature (PV) with the temperature set on the thermostat (SP), generating a controller output to maintain the building at the desired temperature by switching the boiler between turn on and turn off. A closed-loop controller has a feedback loop that ensures that the controller executes a control action to manipulate the process variable to reach the same as the SP. Thus, closed-loop controllers are also called feedback controllers or feedback control systems.

PID feedback control

PID is an acronym for Proportional-Integral-Derivative, referring to the three terms operating on the error signal to produce a control signal. [12, 13, 14]. A PID controller is a closed-loop or feedback control system widely used in control systems. A PID controller continuously calculates an error value (e(t)) as the difference between the desired setpoint and a measured process variable. It also applies a correction based on proportional, integral, and derivative terms. The theoretical understanding and application of PID date from the 1920s. They are implemented in nearly all analog control systems, originally in mechanical controllers, then using discrete electronics and later in industrial process computers. The PID controller is probably the most-used feedback control design [12, 13, 14].

Fig.1 depicts the general form of the PID controller. If u(t) is the control signal sent to the system or plant, y(t) is the measured output, r(t) is the desired output, and e(t) = r(t)-y(t) is the tracking error.

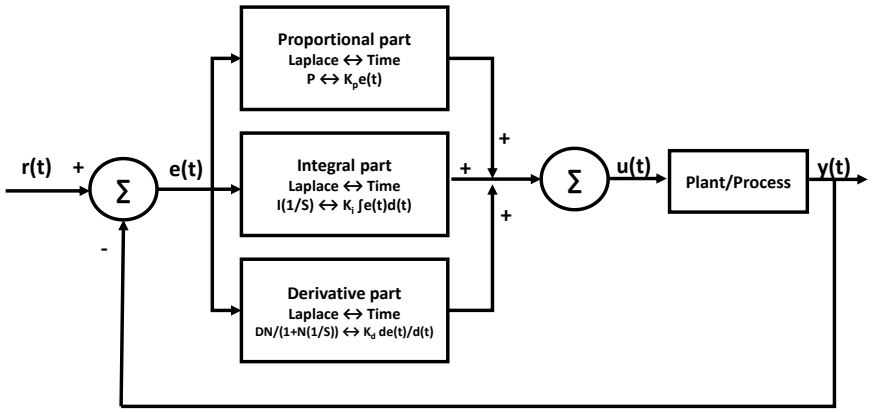


Fig. 1. PID feedback control system schema.

The desired closed-loop dynamics are obtained by adjusting the three parameters, i.e., Kp, Ki, and Kd, often iteratively by "tuning" and without specific knowledge of a plant model.

An approach to understanding PID controllers is by considering that the proportional value depends on the current error, the integral value depends on the past errors, and the derivative value is a prediction of future errors. The sum of these three actions is employed to adjust the process utilizing a control element.

The P control mode considers the product between the error signal and the proportional constant to make the steady-state error approaches zero. Nonetheless, in majority of cases, these values will only be optimal over a particular portion of the total range of control, with optimal values for each portion of the range being different. However, there is also a limit value in the proportional constant; in some cases, the system achieves higher values than desired. This phenomenon is called overshoot and needs to be eliminated as much as possible.

The control mode is intended to decrease and eliminate steady-state error, which is caused by external disturbances, and cannot be corrected by proportional control. The integral control acts when there is a deviation between the variable and the reference point, integrating this deviation in time, that is, observing the samples in past times and adding this to the proportional action.

Derivative action manifests itself when there is a change in the absolute value of the error: If the error is constant, only the proportional and integral modes operate. The error is the deviation between the measurement point and the reference value. The function of the derivative action is to keep the error to a minimum by correcting it in

proportion with the same speed occuring; in this way, the error is prevented from increasing.

PID controllers are the most well-established class of control systems. However, they cannot be used in several more complicated cases. For this situations PI or P controllers are available.

Main control feedback domain application

The applications of feedback control systems are used in numerous applications, such as automation, control processes, servo control, vector control, guidance, and stability controls in the military, cybernetics, and in intelligent control. However, to date, there are no feedback control system applications applied to the production control process during the development of software projects. It is precisely this application that will be explored in this article.

Alternative controllers to the PID

Due to the evolution of feedback control theory, several approaches have been emerging such as control Fuzzy-logic control [15,16], which is widely utilized in machine control. Alternative approaches, such as genetic algorithms [17] and artificial neural networks [18], have been investigated; these approaches have demonstrated that make it easy to mechanize tasks that are already successfully performed by humans. There are other methods proposed for PID autotuning. Based on the integral measures of the open or closed-loop step response, Multi Layer Perceptron (MLPs) are employed to supply PID parameter values for a standard PID controller [19, 20].

3 A Simulation Study: Feedback Loop Control Theory in Software Project Productivity Control

3.1 Research questions

With the simulation study detailed in this section, we intend to answer the following questions:

- Will the simulation of a feedback control system, limited by the maximal productivity achievable by the plant, be able to provide us with information on the feasibility of successfully achieving initial planning of the progress in the construction of a software project?
- If the constraint of the maximal productivity achievable by the plant is eliminated, the closed-loop control system will be able to provide us with the optimal productivity profile over time to successfully achieve the initial planning of the progress in the construction of a software project?
- What type of closed-loop controller achieves the best results?

3.2 Information from the simulation study and limitations.

The simulation study presented in this article uses real data from a development team of three programmers, who have been working as a team for the last year in developing a software system. This software system is a web development using Java programming language and a MySQL database manager.

The history of functionalities implemented by the development team during the last six months was utilized, consisting of 165 functional processes with their COSMIC functional size and the actual effort consumed in Work-Hours by each of the functional processes.

The data gathered were used to define a productivity model that represents the way that the team usually works to model the plant. The productivity model related to the teamwork indicates that average productivity is 2,233 [CFP / day per person], considering working hours of 8-man hours per day per person of the development team. With this real productivity model, which reveals average productivity per person, we have the essential elements to build a plant model for simulation purposes.

It is noteworthy that, on being a simulation study, the developed control system was not fed the real values in each period for the real project. The values considered are those calculated by MatLab software using the Simulink toolbox, in its R2019a version, which was used as a development tool to model the control system proposed in this work.

3.3 Description of the simulation study

In our very particular case, the “Plant” is a team of persons who develop the piece of software. The project administrator determines this number of persons as part of the initial planning. This number of persons translates into the maximal productivity at which the plant can work per day, that is, how many [CFP] per day the plant will be able to produce (considering the total number of persons and knowing the individual productivity per day [CFP / day per person] that one person is capable of building).

It is important to note that the term “day” refers to the “work hours” executed by the development team on a working day, that is, it refers to effort and not elapsed time. This avoids confusion between productivity and speed, in that, in this study, we are focused on productivity.

Considering the productivity available in the plant and the curve representing the planned behavior defined by the project manager in terms of the relationship in days and the functional size in CFP that must be built per day, two types of simulations will be considered:

(i) With the saturation point. The controller will be in charge of controlling the plant's productivity without exceeding the initially planned limit, thus being able, through simulation, to determine whether it will be possible to comply with the initial planning in functional size built and completion time in days with the available work equipment;

(ii) Removing the saturation point. The controller automatically calculates the productivity curve necessary to meet the project completion term and the

functionality to be built. In this case, the controller may exceed the productivity limit. This means that the number of perosns assigned to the development of the piece of software may be greater than that considered in the initial planning.
In this case, the simulation aid the administrator in making the necessary adjustments by increasing the work team in terms of the required days, since the precise productivity curve will have been obtained to achieve the plan.

The reference “SP” that will be used is a function that starts at 0 [CFP] and that models the planned behavior (baseline) of the progress in the construction of the piece of software against time. This function can be a “ramp” type, or can be any function defined according to the desired planning for the project; usually, the planned reference for this type of project follows a growth curve typically modeled by a sigmoid [2, 21]. We will explore the performance of three types of controllers, including PID, PI, and P, and determine which the best controller is.

3.4 Plant modeling

As defined previously and considering real productivity numbers of persons in terms of [CFP / day], the average productivity by a person for each day is 2,233 [CFP / day per person]. For simulation purposes, a plant with a maximum of five developers working in parallel is considered. Therefore, we consider the maximal productivity limit calculated by the controller as (2,233 [CFP / day]) * 5 [persons], rendering a maximal plant productivity of 11,165 [CFP / day], which should be the controller saturation point for the first simulation option. It is noteworthy that, in the first simulation option, the plant experiences maximal productivity (saturation limit of the controller output). In the second simulation option, this limit will be released, allowing the controller to increase the plant's productivity to the extent necessary to accurately fulfill the initial planning of functional size in the planned number of days.

The function that models the plant block in the simulation system is "Y (t) = U (t) + U (t-1)". Where the output of the plant ("Y (t)") at current time is calculated as the sum of the productivity at which the "plant" is working according to the indication of the control signal in the current time "U (t)", plus the control signal in the previous time "U (t-1)". The plant model considers that the built functionality accumulates over time; therefore, the use of a 1/S integrator is necessary (in the Laplace domain) Fig. 2.

The output of the plant comprises the [CFP] built up at the time or period “t” [days], which will become the input to be compared with the reference and for calculating the “error” that will enter into the controller. Behavior using PID, PI, and P controllers will be analyzed. The plant model, considering what was previously explained, and was modeled in MatLab, as shown in Fig. 2.

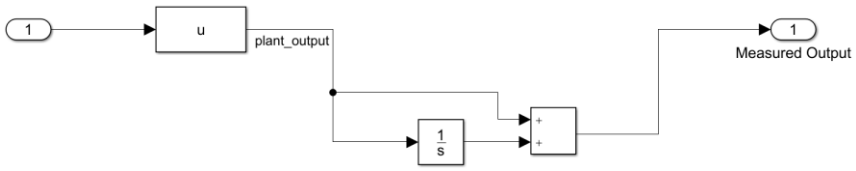


Fig. 2. Plant model

3.5 Scheme of the complete control system and initial conditions

Fig. 3. depicts the schematic of the complete system. The “Ramp” block models the entry into the system starting at time “0”, with a unit slope, where each unit of time “t” represents a working day (8 [Work-Hour]) of development.

The exit of the "Ramp" enters the block called the "Baseline Sigmoid growth curve"; which contains the model example of a baseline that involves planning to build a piece of software of 400 CFP size, moving forward at a planned rate modeled by a function known as sigmoid or growth.

It was decided to use the sigmoid growth function because it adequately models the behavior of software projects as referred to in PMBOK [2] and by other authors [5] in techniques such as Earned Scope Management (ESM). The growth function, realistically models the behavior of the rate of construction of a software project, starting smoothly at first stage and then moving on to a second stage where the fastest growth occurs and to a third stage where growth decreases and stabilizes. The constraint is to complete the project in 40 days. The code and graphic that models this dynamic in MatLab are presented in Fig. 4.

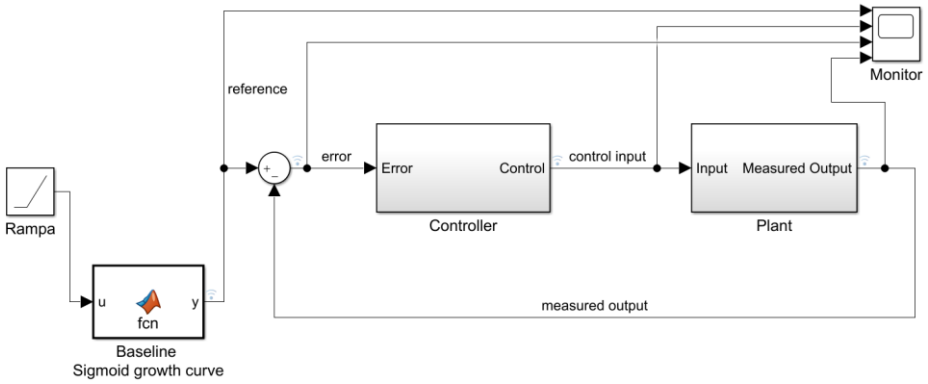
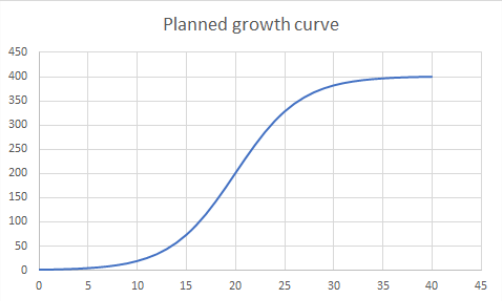


Fig. 3. Complete control system implemented in MatLab using the Simulink toolbox, in its R2019a version.




```
function y = fcn(u)
if u<=40
    y = 400/(1+exp(-1*(0.3*u-6)))
else
    y=400;
end
```

Fig. 4. Planned growth curve and MatLab code

The system error is calculated as the difference between the reference (baseline) minus the functional size in CFP constructed by the "Plant" up to the current time "t"; the error enters the controller which calculates the control input action that represents the productivity in CFP / day that the plant must achieve.

3.6 PID control system simulation

The controller was tuned using the transfer function-based tuning method, defining the constants P = 0.8095, I = 0.454, and D = -0.2141. These tuning constants were calculated in an optimized manner using the tuning tool available in the MatLab control toolbox.

Four graphs are shown in Fig. 5, all these present in the "x" axis, the number of periods simulated, starting at t=0 and finalizing in t= 50.

The first graph is related to the reference or baseline curve (r(t)), meaning the conventional way to be developed and the scope defined, considering that the "y" axis is related to the scope with a maximal value at 400 [CFP].

The second graph is related to the control input ("U (t)"), because this simulation was conducted considering a saturation point, and with the "y" axis defined with a maximum of 11.165 [CFP/Work day] which is reached in period t=14.

The third graph is related to the error ("e(t)") during the periods; in consequence, the "y" axis is defined from the minimal error to the maximal error found in the simulation. In this case, the peak of error was more than 150 [CFP] during period t=28. In this graph, the error is zero (0) approximately in period t=47. That means that the scope endsin this period, seven periods more than those defined by the constraint.

The last graph describes the measured output ("Y(t)"), during period t=40; it is possible to observe that the scope developed is lower than 400 [CFP], and for period t=47 the scope was fully developed.

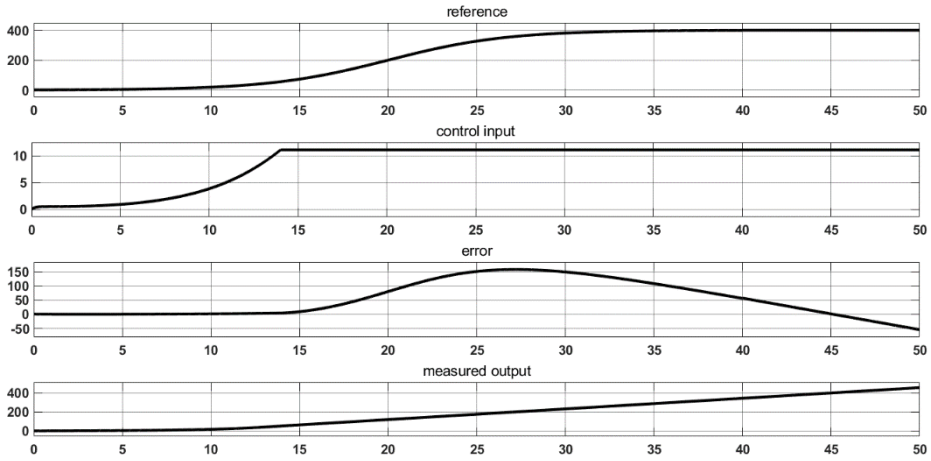


Fig. 5. PID control system simulation with the saturation point

This type of simulation helps to provide information related to the optimal time in days required to build the scope expected. Additionally, it is possible to identify whether it will be possible to comply with the initial planning, considering the specific maximal productivity for the "Plant" represented by the controller's saturation.

Fig. 6. reveals the results of the simulation, eliminating the saturation for the controller. In Fig. 6. it is possible to observe, in the second graph, the values for "y" axis moving toward reaching a productivity equal to 31 [CFP/Work day] in period t=22, generating a lower error (Graph 3) with a maximal deviation from the planning close to 7.5 [CFP] between periods t=15 and t= 20, and also in the periods t=25 to t=30. The negative error means that more scope was developed in those periods, but is also comprises an error.

With this approach, which is without a saturation point for the controller and, developing the scope (400 [CFP]) in 40 days, this is possible, as is shown in Fig. 6.

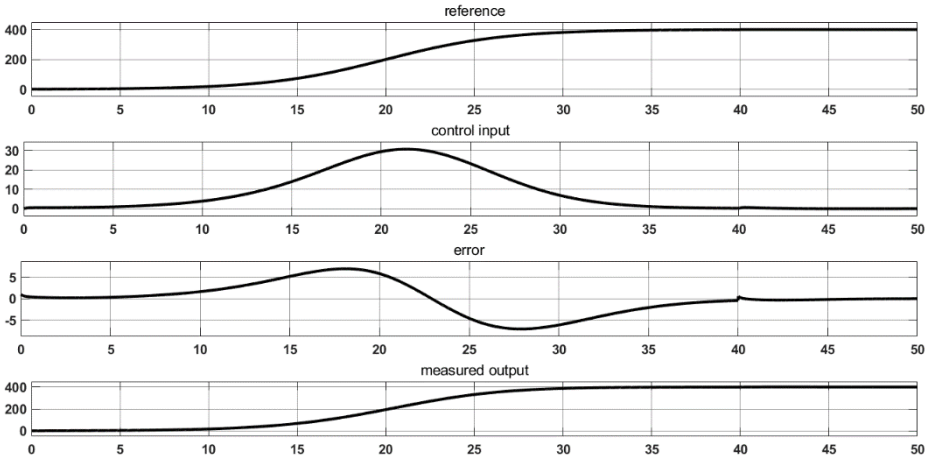


Fig. 6. PID control system simulation removing the saturation point

This simulation provides information related to the optimal productivity required to build the planned functional size in the planned time.

3.7 PI control system simulation

The controller was tuned using the transfer function-based tuning method, defining the constants P = 0, I = 2.661.

Fig. 7. presents the simulation results limiting the saturation of the controller according to the maximal productivity available in the plant. Similarly to the PID control system simulation, four graphs are shown, where it is possible to observe that the controller achieves maximal productivity on period t=14. However, it fails to build 400 CFP on the 40-day plan, requiring 47 periods, seven more than initially planned. In addition to that the maximal error is 150 [CFP] during period t=28. In this case, behavior for the PID and the PI control systems is the same, and for both, with the assumed parameters, it will not achieve the objective.

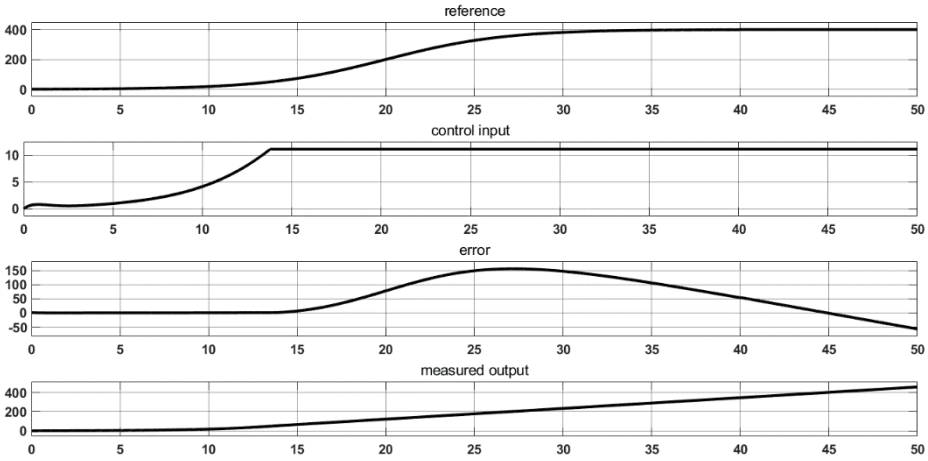


Fig. 7. PI control system simulation with the saturation point

Fig. 8. shows the simulation results, removing the saturation point for the controller, considering the graphs for the PI controller. The simulation indicates the need for a maximal productivity of about 30 [CFP/Work day] on period t=22 (Graph 2). Note that the required productivity is defined as a normal distribution. The maximal deviation error from the plan or e(t) is 1.3 [CFP] during period t=17 (Graph 3), while the error = zero is reached during period t=40.

From Graph 3 in Fig. 8, it is possible to observe that utilizing the approach without the saturation point for the controller, it is possible to develop the scope (400 [CFP]) in 40 days.

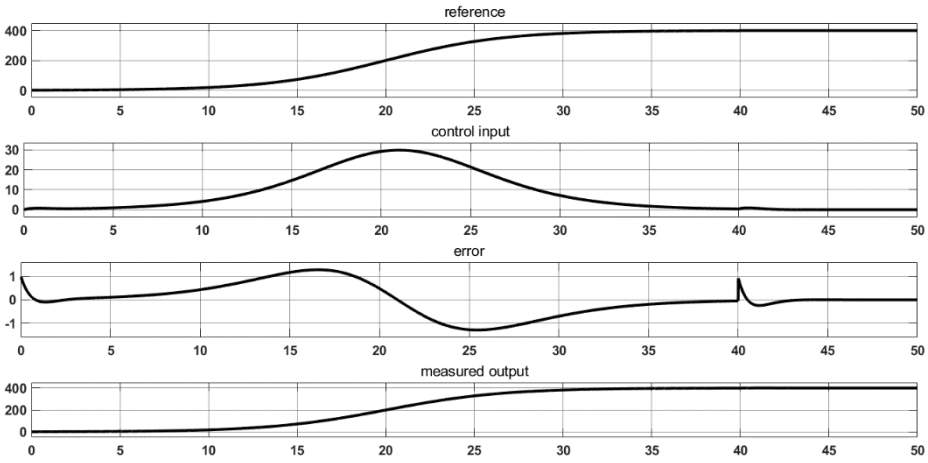


Fig. 8. PI control system simulation removing the saturation point

Note that for the PI controller; the best-tuned constants are P = 0 and I = 2.661; then, the controller has a proportional constant equal to zero, indicating that it is an integral type controller.

3.8 P control system simulation

The controller was tuned using the transfer function based tuning method, defining the constant P=0.872.

Similarly, to the previous controllers, Fig. 9. demonstrates the P controller's behavior considering a saturation point, where the controller reaches maximal productivity during period t=14 (Graph 2). However, it fails to build 400 CFP on the 40-day, requiring eight more periods for finishing to develop the full scope during period t=48. The maximal error was 175 [CFP] during period t=28 (Graph 3), representing the maximal deviation from the plan. From Graph 4, it is possible to observe that the objective to develop 400 [CFP] in 40 days is not reached.

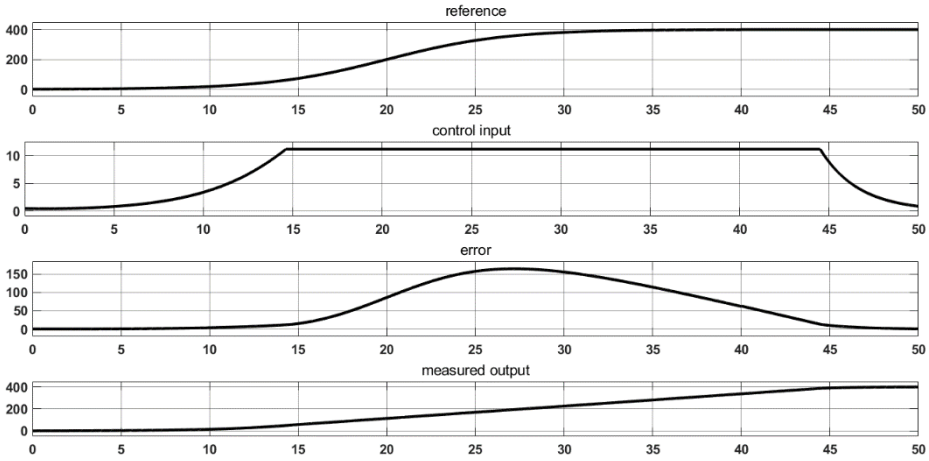


Fig. 9. P control system simulation with the saturation point set

A simulation was conducted, disabling the saturation point constraint for the P controller. In Fig. 10, it is shown that the controller requires maximal productivity near day 28 [CFP/Work day] during period t=22 (Graph 2), if the team reaches this productivity according to the simulation, the maximal error (Graph 3) will be e(t) = 32 [CFP] for period t=23, meaning maximal deviation from the plan. Note that the required productivity (Graph 2) is defined as a normal distribution similar to the error (Graph 3). The period can be a real number because the simulation is performed in continuous time, although the period must be rounded to the nearest higher integer in practice.

Because the error = 0 (zero) is reached during period t=40, the objective is accomplished.

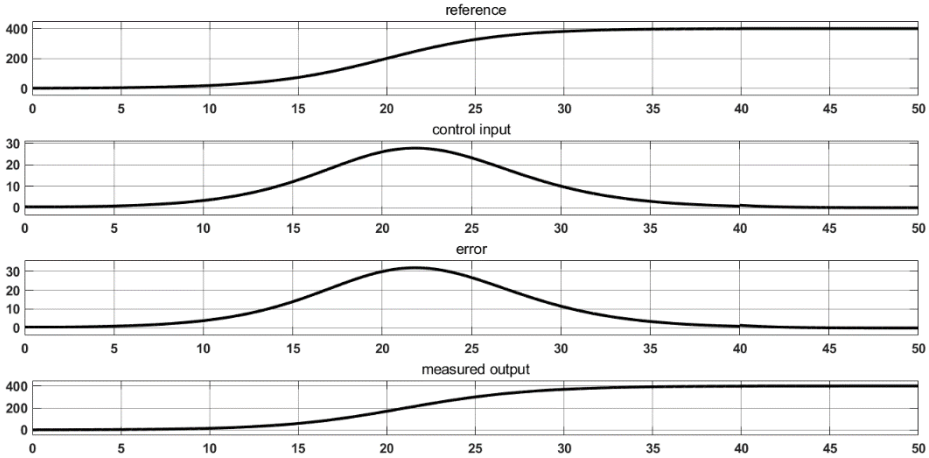


Fig. 10. P control system simulation removing the saturation point

4 Analysis of results

Table 1 depicts the comparison of the results for the simulation of the PID, PI, and P controllers. From Table 1, it is possible to observe that none of the controllers can achieve the goal of building 400 CFP in 40 days with a plant with a saturation point defined by the productivity that five persons could reach per day according to the reference database (11,165 [CFP / day]).

On the other hand, on releasing the saturation point constraint, all of the controllers achieved the objective; however, the PI controller achieved the lowest error e(t)= 1.3 CFP. This means that, with this controller, the lowest variation from the scope developed is presented. However, it is essential to note that the constant Kp of the PI controller is "zero", that it is only the action of the integral controller that is relevant for achieving the best results.

Table 1. Results comparison

Controller type	**Goal**	**Tuning constants**	**Controller Saturation ON**		**Controller Saturation OFF**	
PID	400 CFP	Kp=0.8095	Max productivity (CFP/day)	11.165	Max productivity (CFP/day)	31
	40 days	Ki=0.454	Days required	47	Days required	40
		Kd=-0.2141	Max error (CFP)	150	Max error (CFP)	7.5
			Period Max error (day)	28	Period Max error (day)	17

Contro-ller type	Goal	Tuning constants	Controller Saturation ON		Controller Saturation OFF	
PI	400 CFP	Kp=0	Max productivity (CFP/day)	11.165	Max productivity (CFP/day)	30
	40 days	Ki=2.661	Days required	47	Days required	40
			Max error (CFP)	150	Max error (CFP)	1.3
			Period Max error (day)	28	Period Max error (day)	17
P	400 CFP	Kp=0.872	Max productivity (CFP/day)	11.165	Max productivity (CFP/day)	28
	40 days		Days required	48	Days required	40
			Max error (CFP)	175	Max error (CFP)	32
			Period Max error (day)	28	Period Max error (day)	23

5 Conclusion

The simulation of a closed-loop control system limited by the maximal productivity achievable by the plant provides us with information on the feasibility of successfully achieving the initial planning of the progress in the construction of a software project. This in turn permits us to observe whether the initial planning is feasible or not. In the simulation considering a saturation point, none of the controllers can meet the 400 CFP target in 40 days with the plant productivity limit.

If the restriction of the maximal productivity achievable by the plant is eliminated, the closed-loop control system provides us with the optimal productivity profile over time to successfully achieve the initial planning. In this simulation approach all of the controllers reach the planned objective with differences between them.

From the simulations, the best controller to achieve the defined objective was the Integral type, which achieves a steady-state error of 0 [CFP] and a minimal error during the construction of 1.3 [CFP], compared to the remainder of the simulated controllers(See Table 1).

This indicate that productivity adjustment is more sensitive to good results in the controller that considers, as control action, only the history of the errors (I) and not the proportional (P) or derivative (D) part.

It is recognized that one of the greatest and fundamental problems in software development is the control of project constraints such as scope, time, and cost.

From the project management perspective, several control systems have been developed. However, the information obtained by these control systems must be analyzed by the project managers to define actions subject to their own experience, rendering it difficult to replicate the results among projects, teams, and institutions.

The application of the feedback loop control theory in other disciplines in which there is something to be controlled in order to accomplish a specific objective has been

studied for several years and present good results. This is not so for the case of the software, where this discipline has not been applied. With this simulation study, a possibility to use the feedback loop theory to build controllers that aim to ensuring control stability in software projects is proposed and analyzed, providing good expectations that could be validated in real projects.

5.1 Limitations

A constraint into applying the feedback control theory is to have measurable elements in terms of what the "Plant" will produce. In the case of the software, the plant is represented by a team that produces software; this quantitative element is defined by the functional size, which is the sole standardized measure in software to date.

It is noteworthy that, in this article, we focus on the product scope as is defined by PMBok, utilizing a specific standard for software functionality measurement. However, there are other standards that could be employed and other types of requirements, such as NFR.

5.2 Further Work

The derivative action is important when there are variations in the error. This indicates the need to study, in a later work, phenomena in a plant that could imply changes in the variations of the error, most likely due to disturbances that affect the productivity of the plant.

Once the simulation has demonstrated the feasibility, considering as feedback the real values after each evaluation period (not in a continuous manner as simulation) in a real project should be evaluated.

References

1. Common Software Measurement International Consortium (COSMIC), Measurementt Manual for ISO 19761, Part 1, Part 2, Part 3. Version 5.0, May 2020.
2. Project Management Institute, A Guide to the Project Management Body of Knowledge, PMBOK, 6th ed. 2017.
3. S. Rozenes, Multidimensional project control system, Coventry University, 2004. doi:10.1504/IJPOM.2010.031881.
4. K.A.S. Erik G. Cummings, Cost/Schedule Control Systems Criteria A Reference Guide to C/SCSC Information, Air University, 1992.
5. Francisco Valdés-Souto, "Earned Scope Management: Scope Performance Evalua-tion for Software Projects Considering People and Effort as Resources ", Proceedings 2019 7th International Conference in Software Engineering Research and Innova-tion, IEEE Computer Society CPS, IEEE Catalog Number: CFP19B19-ART, ISBN: 978-1-7281-2524-4

6. Alain Abran, “Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers”, IEEE Computer Society, 2015, ISBN 978-1-118-95408-9. Ed. Wiley, 2015
7. Valdés-Souto, F. (2016), “Creating a Historical Database for Estimation Using the EPCU Approximation Approach for COSMIC (ISO 19761)”, 4th edition of the Inter-national Conference in Software Engineering Research and Innovation (CONISOFT’16), Puebla, México, 27 - 29 Abril 2016. Puebla, Universidad Popular Autónoma de Puebla (UPAEP), IEEE CPS
8. P. Efe, O. Demirors, Applying EVM in a Software Company: Benefits and Difficul-ties, in: O. Demirors, O. Turetken (Eds.), 2013 39th Euromicro Conf. Softw. Eng. Adv. Appl., Conference Publishing Services (CPS), Santander, Spain, 2013: pp. 333–340. doi:10.1109/SEAA.2013.55.
9. M. Ferle, Implementing Earned Value Management on It, in: 19th Int. Cost Eng. Congr., Ljubljana, Slovenia, 2006
10. J. a Lukas, “Earned Value Analysis - Why it Doesn’t Work,” AACE Int. Trans., pp. 1–10, 2008
11. Atherton, Drek P (December 2014). "Almost Six Decades in Control Engineering". IEEE Control Systems Magazine. 34 (6): 103–110. doi:10.1109/MCS.2014.2359588
12. Kiam Heong Ang; Chong, G.; Yun Li (2005). "PID control system analysis, design, and technology" (PDF). IEEE Transactions on Control Systems Technology. 13 (4): 559–576. doi:10.1109/TCST.2005.847331
13. Jinghua Zhong (Spring 2006). "PID Controller Tuning: A Short Tutorial" (PDF). Archived from the original (PDF) on 2015-04-21. Retrieved 2011-04-04
14. Bequette, B. Wayne (2003). Process Control: Modeling, Design, and Simulation. Upper Saddle River, New Jersey: Prentice Hall. p. 129. ISBN 978-0-13-353640-9.
15. Zadeh, Lotfi A. (1965). "Fuzzy sets" (PDF). Information and Control. 8 (3): 338–353. doi:10.1016/S0019-9958(65)90241-X.
16. Pedrycz, Witold (1993). Fuzzy control and fuzzy systems (2 ed.). Research Studies Press Ltd.
17. Vose, Michael (1999). The Simple Genetic Algorithm: Foundations and Theory. Cambridge, MA: MIT Press. ISBN 978-0262220583.
18. Hagan, M. T., O. De Jesus, and R. Schultz, ‘Training Recurrent Networks for Filtering and Control,’ Chapter 12 in Recurrent Neural Networks: Design and Applications, L. Medsker and L.C. Jain, Eds., CRC Press, 311-340 (1999).
19. Richard E. Bellman (December 8, 2015). Adaptive Control Processes: A Guided Tour. Princeton University Press. ISBN 9781400874668.
20. Li, Y., et al. (2004) CAutoCSD - Evolutionary search and optimisation enabled computer automated control system design, Int J Automation and Computing, vol. 1, No. 1, pp. 76-88. ISSN 1751-8520
21. Tom M. Mitchell, Machine Learning, WCB-McGraw-Hill, 1997, ISBN 0-07-042807-7.
22. Luigi Buglione. Some thoughts on Productivity in ICT projects. Version 1.3. WP-2010-01. White Paper. August 23 2010.