Received: 31 August 2017 | Revised: 21 November 2017 | Accepted: 9 January 2018
DOI: 10.1002/cpe.4438

SPECIAL ISSUE PAPER

WILEY

Analyzing the requirements to implement a data analysis model for software process improvement

Jezreel Mejía | Freddy Íñiguez | Mirna Muñoz

Mathematics Research Center, Avenida Universidad 222, 98068 Zacatecas, Mexico

Correspondence
Jezreel Mejía, Mathematics Research Center, Avenida Universidad 222, 98068 Zacatecas, Mexico.
Email: jmejia@cimat.mx

Summary

Software is everywhere. From simple applications to complicated pieces of software, these software products and services facilitate our day-to-day activities, but they also need to be high-quality products in order to guarantee their reliability, security, good performance, and usability. In this context, software development organizations have highlighted that a product's quality largely depends on the quality of the processes followed to build the software. The Software Process Improvement (SPI) initiative is a set of best practices designed to improve software processes through the evaluation of the current organizational practices, competencies, and experiences. However, these SPI initiatives have presented some issues and challenges that complicate their success and preclude software organizations to get improvements. Most of these issues are related to how organizations manage, analyze, and use their own data to improve software processes. The purpose of this paper is to analyze if the three most used software repositories cover the key requirements to implement a data analysis model for implementing SPI initiatives. The results show that the three repositories analyzed as they are do not cover the complete requirements to implement a model highlighting the aspects to be improved.

KEYWORDS

big data, data analysis, information management, knowledge management, Software Process Improvement, software repositories

1 | INTRODUCTION

Nowadays, software products and services are everywhere. There are simple applications like those to listen to music, order food, or even to remember when to drink a glass of water or take a break to breathe. There are also complicated pieces of software, eg, fraud detection systems or software to guide a robot on a precise surgery. These software products, which facilitate our day-to-day life, need to be high-quality products in order to guarantee their reliability, security, good performance, and usability.[1]

In recent years, software development organizations such as large organizations, small-medium enterprises, and very small entities have focused their efforts on implementing models and standards to guide practitioners on the development of software products and services.[2] Because of this, it has been possible to identify that the quality of the products depends largely on the quality of the processes followed to build the software.[3]

In this context, the Software Process Improvement (SPI) involves the performance of a set of best practices and activities designed to improve software organizations processes through the evaluation of current practices and the way software products and services are developed, considering the practitioner's experience and competencies.[4] SPI initiatives have become an essential factor for project managers and software engineers to achieve their goals, maintain competitiveness, and provide a profitable return on investment to organizations.[2]

However, these SPI initiatives have presented a series of issues and challenges that complicate their success and preclude software organizations from making improvements, eg, lack of resources, inexperienced staff, organizational policy, and time pressure, among others.[5] In fact, most of the problems presented in recent years are related to how organizations manage, analyze, and take advantage of their own data to improve.[6] These issues complicate the extraction and management of knowledge toward a successful SPI.

As a solution, the SMART-SPI model was created. This is a data analysis model oriented to SPI, which aims to take advantage of business information to improve software processes. However, to implement this model, a key aspect that should be covered is a repository tool that allows capturing, storing, cleaning, analyzing, and visualizing software process information. Therefore, the goal of this paper is to analyze the capacity of current software repository tools to identify if they achieve requirements to implement the SMART-SPI model.

This paper is organized as follows. Section 2 presents the model background. Section 3 describes the SMART-SPI model. Section 4 shows the analysis of three of the most used current repository. Finally, Section 5 shows the conclusions and future work.

2 | BACKGROUND OF THE SMART-SPI MODEL

Current organizations are rich on data but poor on insights on how to get the best advantage from this.[7] Just 20% of business information are being used for decision making in the organization.

Therefore, the problem resides on 1) how to perform data analysis and 2) which pieces of data and information are the most appropriate to be selected for data analysis.

To develop the SPI model, there were performed steps described as follows:

- *Perform a systematic review from which a set of issues and challenges are identified*. The issues and challenges identified are (1) critical process areas, (2) business objectives and perspectives, (3) data selection and extraction, (4) data cleansing, (5) data understanding, (6) data analysis technique, and (7) suggestion of improvements.
- *Select a data analysis model*. After comparing the five big data analysis models, the most complete over the attributes comparative and the most appropriate big data model was the SMART model.
- *Select a data analysis technique*. The Mining Software Repositories (MSR) techniques were selected because our previous work[6] had emphasized that it is the most used in the analysis of information for SPI.
- *Select an SPI model*. There was selected a methodology for a gradual implementation of improvements focused on reducing resistance to change. The (MIGME-RRC by its Spanish acronym) methodology has evolved and has been proved over different environments such as in the works of Marr[8] and Muñoz,[9] and provides a set of activities to implement SPI initiatives, which encompass the formalization of business objectives, the identification of internal and external best practices, and how to execute the improvements.
- *Select an assessment model for critical process areas identification*. Performing a search over some scientific databases, it was found that the method described in the Software Process Assessment model in ISO/IEC 15504 successfully uses some aspects of the business, eg, business needs and goals, risks, and assessment results, among others, to identify critical software process areas.

3 | SMART-SPI MODEL DESCRIPTION

The SMART-SPI model encompasses a total of 8 phases; each of them integrates a set of activities and recommended tools, which range from the identification of business objectives to the implementation of improvements.

Its purpose is to provide guidance to software practitioners in the design and definition of SPI initiatives, which help to build high-quality software products and services.

One of the key features of SMART-SPI is its adaptability to the current way of building software and to the methodology and practices used by practitioners. This adaptability enables practitioners to keep those good practices that are already implemented by the organization and to adopt new ones based on the recommended methodologies and techniques.

In order to provide a full sight of the SMART-SPI model, Table 1 shows the general structure of the model, using process notation such as phases, activities, inputs, and outputs.

Besides, a milestone diagram of SMART-SPI is presented in Figure 1 and then described.

- *S phase:* This phase encompasses two milestones: 1) to create a strategy for the SPI initiative, formalizing and using the business goals and 2) to identify the internal best practices by the application of questionnaires.
- *M phase:* Three milestones are intended to be accomplished during the development of the M phase: 1) to create a fulfillment matrix, which emphasizes that business objectives are addressed with some specific practices, 2) to identify the most significant process areas for the business in order to have the greatest impact when applying the SPI initiative, and 3) to provide details about the datasets by using a data sheet.
- *A phase:* Milestones for this phase are: 1) to provide some guidelines to help practitioners in the data cleansing and storage process, 2) to provide some guidelines for the selection of the most appropriate data analysis technique, and 3) to analyze process information using the selected data analysis technique.

TABLE 1 SMART-SPI model description

Phase	Activity	Input	Output
S	S.1 Identify current organizational situation		Current situational context
	S.2 Identify business goals and perspectives		SMART strategy board
	S.3 Provide from two to five questions for each panel in the strategy board		SMART strategy board
	S.4 Obtain formalized business goals	SMART strategy board	Formalized business goals
	S.5 Perform interviews		Tacit knowledge extraction
	S.6 Analyze interviews' information	Tacit knowledge SMART strategy board	Practices diagrams
	S.7 Identify generic practices	Practices diagrams	Generic practices
	S.8 Analyze formal process documentation	Formal process documentation	List of unadopted practices
	S.9 Trace formal process documentation and generic practices	Generic practices Formal process documentation	Internal best practices
M	M.1 Coverage analysis	Business goals Internal best practices	List of indicators
	M.2 Fulfillment analysis	Business goals List of indicators	Fulfillment matrix
	M.3 Prioritize indicators	Business goals List of indicators	Prioritized indicators
	M.4 Identify critical process areas	CMMI process areas Business goals Process assessments results	Critical software process areas
	M.5 Understand formats of data		
	M.6 Identify datasets for SMART questions	SMART strategy board	SMART data sheet
A	A.1 Identify kind of data	SMART data sheet	List of types of data
	A.2 Collect process information	SMART data sheet	
	A.3 Select the most appropriate data analysis techniques	Process information	Data analysis technique and algorithms list
	A.4 Apply data analysis techniques to database	Process information Data analysis technique and algorithms list	Data models
R	R.1 Analyze results	Data models	Validated data models
	R.2 Select the most appropriate data visualization technique	Data models	Dataviz graphs
	R.3 Communicate results	Dataviz graphs	
T	T.1 Fulfillment matrix review	Fulfillment matrix	Prioritized list of practices to be improved
	T.2 Select models and standards to be analyzed	SMART strategy board	Most used models and standards list
	T.3 Choose a reference model	Most used models and standards list	Reference model
	T.4 Choose the processes	Fulfillment matrix	List of processes to be analyzed
	T.5 Establish detail level for analysis	Most used models and standards list	Detail level for analysis
	T.6 Create a correspondence template	Detail level for analysis	Correspondence template
	T.7 Identify similarities between models and standards		Correspondence template updated
	T.8 Establish the reference multi-model environment	Correspondence template	Multi-model environment
	T.9 New business opportunities		New business-opportunities document
S'	S'.1 Fulfillment matrix update and review	Fulfillment matrix	Fulfillment matrix updated
	S'.2 Supervise initial business goals	SMART strategy board	
P	P.1 Perform postmortem questionnaires		Postmortem information
	P.2 Create business decisions document	Postmortem information	Business decisions document
I	I.1 Analyze resistance-to-change factors		Resistance to chance factors and risks table Cause-effect diagrams
	I.2 Select best external practices	Fulfillment matrix	External best practices
	I.3 Integrate best practices		New software processes
	I.4 Select pilot projects		Pilot project list
	I.5 Start up the SPI initiative	New software processes	Start-up plan New software processes material

- *R phase:* For the R phase, two milestones have been defined: 1) to provide some guidelines that help practitioners to select the most appropriate data visualizations tools according to the type of available data and 2) to provide some guidelines to communicate the results of the data analysis process in a way that most of the stakeholders can understand.
- *T phase:* The unique milestones for this phase is to identify external best practices from a multi-model environment in order to have some practices from other methodologies, models, and standards for software development that help to fulfill the current business goals.

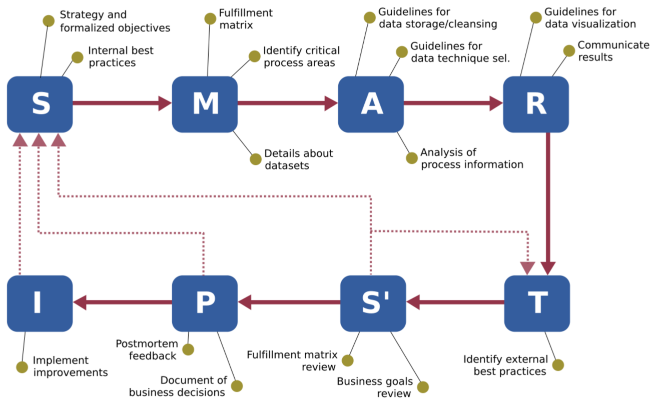
Strategy and formalized objectives
Internal best practices
Fulfillment matrix
Identify critical process areas
Guidelines for data storage/cleansing
Guidelines for data technique sel.
Guidelines for data visualization
Communicate results
S
M
A
R
Details about datasets
Analysis of process information
I
P
S'
T
Implement improvements
Postmortem feedback
Document of business decisions
Fulfillment matrix review
Business goals review
Identify external best practices

FIGURE 1 SMART-SPI milestones

- *S' phase:* Two milestones have been identified for this phase: 1) to update and review the fulfillment matrix and 2) to evaluate the accomplishment of business goals.
- *P phase:* P phase's milestones are: 1) to perform some postmortem questionnaires in order to obtain feedback for the SPI initiative and 2) to create a document of business decisions that help the organization to achieve its mission.
- *I phase:* The only milestone in this phase is to implement the SPI initiative by executing the internal and external best practices that have been identified.

4 | REQUIREMENTS TO IMPLEMENT THE SMART-SPI MODEL

This section presents a list of requirements that help guarantee the implementation of the SMART-SPI model in software development organizations.

- *High-level support*. In order to facilitate the implementation of SMART-SPI, this must be supported by the CEO to project managers and developers, so that efforts are focused on the same direction and any problem can be easily solved.
- *Communicate the plan*. It is highly recommended to communicate the plan along the different levels of the organizations; thus, the purpose of SMART-SPI is well understood, and its activities are implemented correctly like the application of questionnaires to identify internal best practices.
- *Reduce resistance-to-change attitudes*. Occasionally, process monitoring and assessment can be perceived as a threat by stakeholders as a way to extract knowledge from stakeholders, so that, once captured, they will lose value for the organization and may be fired. The recommendation to avoid these attitudes is to communicate the purpose and vision of the organization; thus, every employee would understand the real need for the SPI initiative implementation.
- *Many ways to do the same*. SMART-SPI is a reference model that provides activities to implement SPI initiatives. However, it is possible to adapt or change these activities for better ones or activities more comfortable for practitioners, but it is important to remember the purpose and milestones of SMART-SPI through each phase.
- *Adaptation, not to force implementation*. One of the most important features of SMART-SPI is that it does not intend to force practitioners to follow practices that do not correspond to the way they develop software. In other words, practices and activities from SMART-SPI are intended to be adapted according to the capacity of the software organizations.

From a data analysis perspective, the SMART-SPI model provides some guidelines for creating clean well-structured software process information databases to be able to apply some data analytics and to get insights to improve software processes by the software development organizations such as large organizations, small-medium enterprises, and very small entities that are facing the implementation of quality models and standards. The following is a main requirement to perform a data analysis and implement the model in a correct way.

- *Software repository*. Given that the SMART-SPI model integrates mining software repository techniques to analyze software process information that relies on software repositories to manage their process information, eg, project plans, requirements specification, time and cost estimation, risks, schedules, and tasks assignment, among others. The software repository is a key requirement to perform the model in a correct way. Therefore, it is necessary to establish a software repository that manages information or data related to software processes.

The next subsection describes the software repositories and how the analyses were performed in order to identify the type of information and functionalities related to software processes information that a software repository should cover.

4.1 | Software repositories

Nowadays, most of the software development companies are using software repositories to manage their process information, eg, project plans, requirements specifications, time and cost estimations, risks, schedules, and tasks assignment, among others. Therefore, this section implements an analysis of functionalities to evaluate the capacity of current software repository tools to manage process information.

The purpose of this analysis is to determine if there is a software repository tool that allows the data analysis and can be integrated into the SMART-SPI model to facilitate the storage, cleansing, analysis, and visualization of the software process information, ie, to facilitate the implementation of SM**AR**T-SPI's phases **A** and **R**. In order to do this, a comparison of the functionalities and management of information related to software processes of three of the most used software repositories tools (GitHub, GitLab, and Bitbucket) was performed.

According to the work of Hong,[10] given their functionalities and collaboration capabilities, GitHub, Bitbucket, and GitLab are considered as the most popular software repositories nowadays. Only GitHub has reported 24 million users across 200 countries with more than 67 million repositories in its Octoverse report this year.[11] Because of this, these three software repository tools have been selected in the comparison for this research.

Developing software products and services is a creative task that, for its success, requires developers to collaborate among them and have a good control of their resources. Software developers can manually manage the code, versions, and features of software products, but nowadays, it is highly recommended to use a Source Code Management (SCM) system. An SCM is a tool that facilitates the control of the code and the resolution of defects and errors and monitoring and control of some activities for a software project development.[12]

Software repositories may contain useful information, which is typically analyzed to develop new tools and solutions that support the maintenance of software products, their progress and evolution, facilitate reusability, and validate novel ideas.[13] In the present day, there is a vast quantity of these tools, like GitHub, Bitbucket, SourceForge, Launchpad, Assembla, CodePlex, and Savannah, among others[10]; they all offer different functionalities and features to control software projects information. Three of these tools have been selected for this analysis of functionalities, regarding its popularity in the software development field as well as personal usage and experience. They are explained in the following sections.

4.1.1 | GitHub

GitHub is a platform developed by GitHub, Inc., (originally named Logical Awesome) launched in 2008, with the main purpose of helping to develop software products collaboratively by using the Git version control system.[14] Given its easy-to-use and collaborative characteristics, GitHub is one of the most used tools to host Git repositories, with more than 38 million projects.[15]

This SCM tool lets practitioner store software product codes and provides tools to collaborate with other developers of the same project or public collaborators. It is worthy to mention some of the GitHub remarkable features:

- *Issue tracking*. GitHub integrates an issue tracker, which announces bugs and errors to software developers when they appear. It is also possible to see the progress of issues and manage them into specific categories by using labels and milestones.
- *Reuse of code*. Using an Elasticsearch cluster, software developers can easily find portions of code and reuse them in other projects.
- *Monitoring and dashboards*. GitHub provides a variety of graphs and charts, which help visualize the work across a specific project. In addition, it is possible to see more details about the work flow of a certain project by using the Activity Dashboard.

4.1.2 | GitLab

Launched in 2012 by Dmitriy Zaporozhets and Valery Sizov,[15] GitLab is a free software SCM that allows software developers to make changes to the tool and adapt the solution to their specific needs. The following list comprises some of the remarkable functionalities of GitLab:

- *Powerful APIs*. An Application Programming Interface (API) is a set of resources like tools and communication protocols to link software components when building software products and services. GitLab provides a simple and powerful API, where each of its components is described in a dedicated location on the Internet.
- *Issue tracking and board*. The GitLab issue tracker is a place where issues are displayed in separated columns, regarding the labels software developers have established for specific issues. This tool lets practitioners easily visualize the progress of a software project and identify problems like bottlenecks or critical bugs.
- *Time tracking*. Time estimation is a common activity that practitioners perform for every software project. This activity is usually performed on a third-party application. However, GitLab provides an own time tracking system, where initial estimations for a specific issue are established by

using the "/estimate" tag. Meanwhile, the "/spend" tag is used every time software developers work on that specific issue. When closing an issue, the total time spent is displayed.
- *Activity stream*. In order to let software developers and leaders know the progress of a software project, the activity stream is a functionality within a GitLab repository, which displays information about the activity of a specific repository in the dashboard view. This activity information may encompass up to six kinds of content: projects in collaboration, starred projects, project's activity, starred project's activity, groups, and to-dos. This is a useful feature, which allows practitioners to focus their efforts on those critical tasks.
- *Cycle analytics*. Finally, this functionality is a big differentiator of GitLab from GitHub and Bitbucket. Cycle analytics displays statistics, projections, and estimations for software repositories, taking data from projects. This information is around issues, developing time (planned versus real time for implementing an issue), coding tasks (estimated time to fulfill a specific merge), testing and continuous integration tasks, and statistics for review, staging, and production/deployment tasks.

4.1.3 | Bitbucket

This SCM tool was launched in 2008, by an Australian startup. It integrates a set of services and functionalities to manage software project information. Moreover, it is also possible to connect with some other functionalities and services from Atlassian, which is the organization that acquired Bitbucket in 2010. Its target customers are large organizations.[15] As well as GitHub and GitLab, Bitbucket is a powerful tool, which integrates the most common functionalities to manage a software project. However, the following list encompasses some of the remarkable functionalities of Bitbucket:

- *Git Large File Storage*. This functionality is a new implementation of Bitbucket, which enables practitioners to store and preview big files in software repositories, unlike GitHub or GitLab, which do not support files bigger than 100 MB.
- *JIRA Software integration*. As part of an Atlassian product, Bitbucket has the ability to integrate other products and services from Atlassian. Bitbucket has a direct integration with JIRA, which is a development tool where it is possible to create a Scrum or Kanban board and display agile reports.
- *third-party integrations*. Another useful functionality of Bitbucket is the capacity to integrate third-party services, not only those services of Atlassian but other services that provide wider possibilities for software practitioners and project leaders when developing software products.

4.2 | Software repositories analysis

For this section, the first part encompasses the identification of software process information on the software repositories. Meanwhile, the second part of the comparison evaluates the process information management capacity of the selected tools.

4.2.1 | Software process information on software repositories

The evolution of current software repositories tools has been oriented to fulfill the needs of software developers and organizations during software products and services development. Therefore, each of the software repository tools provides specific functionalities and capabilities to store information of different types. Then, in order to identify the type of information and functionalities related to software processes, Table 2 shows a comparison of the main functionalities and storage capacity of GitHub, GitLab, and Bitbucket. Those functionalities that are related to software process information are marked in grey color.

As shown in Table 2, from a total of 30 functionalities, just 9 of them are related to software process information.

However, in order to select a software repository tool to be integrated into the SMART-SPI model, it is necessary to compare these results with the information that a regular software development organization can generate during the development of a software product or service. This activity is shown in the following section.

4.2.2 | Process information management capacity of software repositories

CMMI-DEV is a model that provides guidance to practitioners in the development of software products and services by promoting the implementation of best practices from government and industry.[16]

Because of the wide adoption of MMI-DEV, this model has been selected to represent the activities of a regular software development organization, specifically, process areas of the maturity level 2 because this maturity level is considered as the first level in which an organization can be certified. The processes areas and Specific Practices (SP) assigned to this maturity level are[16]:

1. Configuration Management
   - SP 1.1 Identify Configuration Items
   - SP 1.2 Establish a Configuration Management System

TABLE 2 Software repositories comparison matrix

Comparative Functionalities	Software Repositories		
	GitLab	Bitbucket	GitHub
Access level	✓	✓	✓
Activity	✓	✗	✗
Branching	✓	✓	✓
Charts and graphs	✓	✗	✓
Clone repository	✓	✓	✓
Collaborators	✓	✓	✓
Commits	✓	✓	✓
Compare changes, branches	✓	✗	✓
Cycle analytics	✓	✗	✗
Delete repository	✓	✓	✓
Deployment and services	✓	✗	✓
Fork repository	✓	✓	✓
Landing pages	✓	✓	✓
Issue tracking	✓	✓	✓
Issues	✓	✓	✓
Labels and tags	✓	✗	✓
Language	✓	✓	✓
License	✓	✗	✓
Mailing lists	✗	✓	✗
Merge	✓	✓	✓
Milestones	✓	✗	✓
Pull requests	✓	✓	✓
Repository information	✓	✓	✓
Snippets	✓	✗	✗
Stars	✓	✗	✓
Time tracking	✓	✗	✗
Transfer repository	✓	✓	✓
Watch	✗	✗	✓
Webhooks	✗	✓	✓
Wiki	✓	✓	✓

- SP 1.3 Create or Release Baselines
- SP 2.1 Track Changes Requests
- SP 2.2 Control Configuration Items
- SP 3.1 Establish Configuration Management Records
- SP 3.2 Perform Configuration Audits

2. Measurement and Analysis

- SP 1.1 Establish Measurement Objectives
- SP 1.2 Specify Measures
- SP 1.3 Specify Data Collection and Storage Procedures
- SP 1.4 Specify Analysis Procedures
- SP 2.1 Obtain Measurement Data
- SP 2.2 Analyze Measurement Data
- SP 2.3 Store Data and Results
- SP 2.4 Communicate Results

3. Project Monitoring and Control

- SP 1.1 Monitoring Planning Parameters
- SP 1.2 Monitor Commitments
- SP 1.3 Monitor Project Risks
- SP 1.4 Monitor Data Management
- SP 1.5 Monitor Stakeholder Involvement
- SP 1.6 Conduct Progress Reviews

- SP 1.7 Conduct Milestones Reviews
- SP 2.1 Analyze Issues
- SP 2.2 Take Corrective Action
- SP 2.3 Manage Corrective Actions

4. Project Planning

- SP 1.1 Estimate the scope of the project
- SP 1.2 Establish Estimates of Work Product and Task Attributes
- SP 1.3 Define Project Lifecycle Phases
- SP 1.4 Estimate Effort and Cost
- SP 2.1 Establish the Budget and Schedule
- SP 2.2 Identify Project Risks
- SP 2.3 Plan Data Management
- SP 2.4 Plan the Project's Resources
- SP 2.5 Plan Needed Knowledge and Skills
- SP 2.6 Plan Stakeholder Involvement
- SP 2.7 Establish the Project Plan
- SP 3.1 Review Plans that Affect the Project
- SP 3.2 Reconcile Work and Resource Levels
- SP 3.3 Obtain Plan Commitment

5. Process and Product Quality Assurance Requirements Management

- SP 1.1 Objectively Evaluate Processes
- SP 1.2 Objectively Evaluate Work Products
- SP 2.1 Communicate and Resolve Noncompliance Issues
- SP 2.2 Establish Records

6. Requirements Management

- SP 1.1 Understand Requirements
- SP 1.2 Obtain Commitment to Requirements
- SP 1.3 Manage Requirement Changes
- SP 1.4 Maintain Bidirectional Traceability of Requirements
- SP 1.5 Ensure Alignment Between Project Work and Requirements

7. Supplier Agreement Management

- SP 1.1 Determine Acquisition Type
- SP 1.2 Select Suppliers
- SP 1.3 Establish Supplier Agreements
- SP 2.1 Execute the Supplier Agreement
- SP 2.2 Accept the Acquire Product
- SP 2.3 Ensure Transaction of Products

Therefore, to identify if the software repository tools are appropriate to be used to manage the process information, Table 3 compares the process information from specific practices and work products of the maturity level 2 of CMMI-DEV in which the software repository functionalities can support process information (specific tools are shown in parenthesis). For this paper, from the total of 54 practices, Table 3 shows those practices which are supported by at least one of the software repository functionalities.

For instance, practice *1.1 Establish Measurement Objectives* from the Measurement and Analysis (MA) process area is supported by the milestones and labels and tags functionalities, both of these ones are provided by GitLab and GitHub.

From the comparison presented in Table 3, it is concluded that the software repositories tools cover the following process areas.

Partially covered:

- Configuration Management: 2 out 7 practices.
- Measurement and Analysis: 2 out 8 practices.
- Project Monitoring and Control: 6 out 10 practices.

- Project Planning: 4 out 14 practices.
- Requirements Management: 3 out 5 practices.

Not covered:

- Process and Product Quality Assurance: 0 out 4 practices.
- Supplier Agreement Management: 0 out 6 practices.

TABLE 3 Maturity level 2 of CMMI process areas coverage by software repositories

Process area	Specific practice	Example Work Products	Covered By: GitLab = GL Bitbucket = B GitHub = GH
Configuration Management (CM)	SP 1.3 Create or Release Baselines	• Baselines • Description of baselines	Milestones: (GL, GH) Labels and tags: (GL, GH)
	SP 2.1 Track Changes Requests	• Change request	Issues: (GL, B, GH) Issue tracking: (GL, B, GH)
Measurement and Analysis (MA)	SP 1.1 Establish Measurement Objectives	• Measurement objectives	Milestones: (GL, GH) Labels and tags: (GL, GH)
	SP 2.4 Communicate Results	• Delivered reports and related analysis reports	Charts and graphs: (GL, GH)
Project Monitoring and Control (PMC)	SP 1.1 Monitoring Planning Parameters	• Records of project performance • Records of significant deviations • Cost performance reports	Cycle analytics: (GL)
	SP 1.2 Monitor Commitments	• Records of commitment reviews	Issue tracking: (GL, B, GH)
	SP 1.3 Monitor Project Risks	• Records of project risk monitoring	Labels and tags: (GL, GH)
	SP 1.5 Monitor Stakeholder Involvement	• Records of stakeholder involvement	Activity: (GL) Issue tracking: (GL, B, GH) Commits: (GL, B, GH)
	SP 1.7 Conduct Milestones Reviews	• Documented milestones review results	Milestones: (GL, GH)
	SP 2.1 Analyze Issues	• List of issues requiring corrective actions	Issue tracking: (GL, B, GH) Issues: (GL, B, GH)
Project Planning (PP)	SP 1.1 Estimate the scope of the project	• Task descriptions • Work package descriptions • WBS	Issue Tracking: (GL, B, GH) Issues: (GL, B, GH) Milestones: (GL, GH)
	SP 1.3 Define Project Lifecycle Phases	• Project lifecycle phases	Milestones: (GL, GL) Labels and Tags: (GL, GH)
	SP 1.4 Estimate Effort and Cost	• Estimation rationale • Project effort estimates • Project cost estimates	Cycle analytics: (GL)
	SP 2.7 Establish Project Plan	• Overall project plan	Milestones: (GL, GH)

TABLE 3 (Continued) Maturity level 2 of CMMI process areas coverage by software repositories

Process area	Specific	Example Work Products	Covered By:
Requirements Management (REQM)	SP 1.1 Understand Requirements	• List of criteria for distinguishing appropriate requirements providers • Criteria for evaluation and acceptance of requirements • Results of analyses against criteria • A set of approved requirements	Issues: (GL, B, GH)
	SP 1.3 Manage Requirements Changes	• Requirements change request • Requirements change impact records • Requirements status • Requirements database	Issue tracking: (GL, B, GH) Issue: (GL, B, GH)
	SP 1.4 Maintain Bidirectional Traceability of Requirements	• Requirements database traceability matrix • Requirements database • Requirements tracking system	Issue tracking: (GL, B, GH)

As a result of this analysis, for the development of a tool to allow manage and perform data analysis regarding to software processes information, it must integrate functionalities at least of process areas related to project management established in the CMMI-DEV model[16]: *"The Basic Project Management process areas address the activities related to establishing and maintaining the project plan, establishing and maintaining commitments, monitoring progress against the plan, taking corrective action, and managing supplier agreements."*

From 7 process areas assigned in maturity level 2, 4 process areas are related to project management. Therefore, the tool must cover as functionality 35 specific practices regarding to the follow process areas:

- Project Monitoring and Control: 10 practices.
- Project Planning: 14 practices.
- Requirements Management: 5 practices.
- Supplier Agreement Management: 6 practices.

5 | CONCLUSIONS

Software Process Improvement (SPI) initiatives help practitioners to improve the quality of practices used to develop software products and services. It has been proved that when the quality of the processes is improved, the quality of software products is also positively affected.

However, SPI initiatives have presented some issues and challenges around the management of data and information, which prevent obtaining improvements.

Based on the issues and challenges software development organizations are facing, a data analysis model for SPI has been proposed in order to provide a solution to each one of the issues and challenges, having, as a result, the SMART-SPI model, which integrates activities to perform SPI initiatives, taking advantage of business information to support the software process improvement.

An analysis of functionalities was performed to evaluate the capacity of software repositories tools to manage process information to perform data analysis and to be able to integrate them into the SMART-SPI model to help its implementation. From the comparison presented in Table 3, it is concluded that from a total of 54 practices, the software repositories tools can cover the following practices by themselves:

- GitLab covers 17 out 54 practices (31%);
- Bitbucket covers 8 out of 54 practices (15%);
- GitHub covers 15 out of 54 practices (28%).

That means that the current functionality and storage capacity of the most complete software repository tool (GitLab) covers only 17 practices (31%) of process information a regular software development organization can generate during software development.

As future work, based on the results of the analysis of functionalities, a further step on this research has been set, which encourages the development of a tool to manage the software process information that can be integrated into the SMART-SPI model to facilitate the capture, storage, cleansing, analysis, and visualization of software process information. Moreover, we are working on prioritizing which are the priority practices and their work products. In addition, the case study is being prepared to analyze an organization that has a level 2 certification of CMMI-DEV.

ORCID

Freddy Íñiguez http://orcid.org/0000-0002-3848-3648
Mirna Muñoz https://orcid.org/0000-0001-8537-2695

REFERENCES

1. O'Regan G. Introduction to software process improvement. *J Chem Inf Model*. 2013;53(9):1689-1699.
2. Kuhrmann M, Konopka C, Nellemann P, Diebold P, Münch J. Software process improvement: where is the evidence? Initial findings from a systematic mapping study. Paper presented at: Proceedings of the 2015 International Conference on software and System Process; 2015; Tallinn, Estonia.
3. Mejia J, Muñoz E, Muñoz M. Reinforcing the applicability of multi-model environments for software process improvement using knowledge management. *Sci Comput Program*. 2016;121:3-15.
4. Chugh M, Chugh N, Punia DK. Evaluation and analysis of knowledge management best practices in software process improvement: a multicase experience. Paper presented at: 2nd International Conference on Advances in Computing and Communication Engineering (ICACCE); 2015; Dehradun, India.
5. Khan A, Keung J. Systematic review of success factors and barriers for software process improvement in global software development. *IET Software*. 2016;10(5):125-35.
6. Mejía J, Iñiguez F, Muñoz M. Data analysis for software process improvement: a systematic literature review. Paper presented at: Recent Advances in Information Systems and Technologies WorldCIST 2017; 2017; Madeira, Portugal.
7. Marr B. *Big Data: using SMART Big Data, Analytics and Metrics to Make Better Decisions and Improve Performance*. Chichester, UK: Wiley; 2015.
8. Muñoz M, Mejía J, Gasca G. A methodology for establishing multi-model environments in order to improve organizational software processes. *J Softw Eng Knowl Eng*. 2014;24:909-33.
9. Muñoz E, Muñoz M, Capón E, Mejía J. Knowledge management in process improvement and vest practices sharing. *IEEE Lat Am Trans*. 2014;12:469–74.
10. Chue N. Choosing a repository for your software project. Software Sustainability Institute. 2017. https://www.software.ac.uk/resources/guides/choosing-repository-your-software-project
11. The state of the Octoverse. GitHub Octoverse; 2017. https://octoverse.github.com
12. Li S, Tsukiji H, Takano K. Analysis of software developer activity on a distributed version control system. Paper presented at: 30th International Conference on Advanced Information Networking and Applications Workshops; 2016; Crans-Montana, Switzerland.
13. Chaturvedi K, Singh V, Singh P. Tools in mining software repositories. Paper presented at: 13th International Conference on Computational Science and Its Applications; 2013; Ho Chi Minh City, Vietnam.
14. Castillo L. *Conociendo GitHub Documentation. Release 0.1*. 2017. https://media.readthedocs.org/pdf/conociendogithub/latest/conociendogithub.pdf
15. Medium Corporation: Github vs Bitbucket vs GitLab vs Coding. Repository management services compared. 2016. https://medium.com/flow-ci/github-vs-bitbucket-vs-gitlab-vs-coding-7cf2b43888a1
16. CMMI for development, version 1.3. Improving processes for developing better products and services. Bedford, Massachusetts: Software Engineering Institute; 2010.

How to cite this article: Mejía J, Íñiguez F, Muñoz M. Analyzing the requirements to implement a data analysis model for software process improvement. *Concurrency Computat Pract Exper*. 2019;31:e4438. https://doi.org/10.1002/cpe.4438