Determinación de la Calidad de Producto, Persona y Proceso en entornos de desarrollo con GitHub: un estudio sistemático de la literatura

Esteban Jorge Alonso[1], Gabriela Robiolo[1]

[1] Universidad Austral, Facultad de Ingeniería, (1629) Pilar, Provincia de Buenos Aires, Argentina
{estebanalonso, grobiolo}@austral.edu.ar
https://www.austral.edu.ar/ingenieria/

Resumen. La comparación entre costo, tiempo y alcance de un proyecto de desarrollo de software y el resultado final, ha brindado una manera simple, aunque sesgada de medir la eficiencia y eficacia de un equipo de proyecto. Consideramos necesario comprender que algunas características de cada desarrollador más los resultados de un proceso creativo del equipo de trabajo dentro del ciclo de vida del Producto, incide directamente en la calidad de éste. Proponemos entonces considerar la calidad de las 3Ps: Producto, Persona y Proceso como una manera holística de determinar la calidad total en la Ingeniería de Software. Pretendemos encontrar métricas objetivas para las 3Ps cuya fuente de datos sea GitHub. De un estudio sistemático de la literatura realizado, los resultados muestran que los estudios previos se centraron en la definición de métricas calculables desde metadata de GitHub para proyectos de código abierto y sin foco en las 3Ps.

1 Introducción

La evolución en el desarrollo de software, tendiendo a materializarse en proyectos cada vez más complejos y eventualmente soportados por herramientas automatizadas, en muchos casos se realiza en ambientes distribuidos [39], donde colaboran muchas personas, particularmente desde principios de 2020 cuando la mayoría del mundo ha tenido que sufrir diferentes niveles de estrictas cuarentenas . Esto provocaría un aumento en la cantidad de datos relacionados con el Producto (artefacto de software) y el proceso (de desarrollo) disponibles. Es posible así investigar la posibilidad de extraer información de los sistemas de control de versiones, toda vez que éstos han sido adoptados para una gran cantidad de organizaciones y empresas. A 2016 se midió la cantidad de proyectos que contenía GitHub en 35 millones [27].

Desde una perspectiva empresarial, un proyecto es un esfuerzo de tiempo limitado destinado a llevar a una organización de un nivel de desempeño medido a un nivel más alto. Para determinar si hemos logrado el objetivo perseguido, necesitamos buenos y adecuados métodos de medición objetiva. Si medimos las cosas incorrectas, entonces nuestro enfoque y atención se desviarán de lo importante. La Triple Restricción clásica (Triple Constraint), como parámetro para medir el éxito de un proyecto, no resulta del

todo adecuada ya que no nos permite medir el éxito alcanzado con respecto a la oportunidad [10]. Particularmente visto desde una perspectiva de Ingeniería de Software, si el resultado de un proyecto de desarrollo es un Producto (de software), independientemente de la gestión del tiempo, el costo y el alcance, es de mucho valor atender la calidad del Producto, del Proceso de desarrollo en sí (no la tarea "Desarrollar") y de la o las personas que participaron en el mismo con un rol de Desarrollador.

El contenido de la metadata de GitHub es vasto, creciente en el tiempo y podría ser enriquecido mediante políticas que rijan su uso. Aún en expansión, los datos que actualmente se puedan recuperar de este repositorio podrían transformarse en métricas de calidad para los 3 ejes estudiados, de acuerdo con la siguiente especificación:

1. Calidad de Persona. Este concepto ha estado más asociado a evaluaciones de desempeño (como política de desarrollo profesional de las empresas) que a un proceso objetivo de recolección de datos sobre el desempeño personal. De hecho, está más asociado a tareas de áreas como Capital Humano o Recursos Humanos.

2. Calidad de Producto. La calidad de producto suele estar asociado al volumen de issues o problemas detectados por usuarios y no tanto del resultado de una medición objetiva sobre la base de objetivos y métricas de calidad esperadas.

3. Calidad de Proceso. Sobre el proceso de desarrollo, existe una tendencia a vincular el proceso con el proyecto de desarrollo el cual está vinculado a la práctica de Project Management en donde se busca que el resultado del proyecto (artefacto o producto entregable del desarrollo) haya estado dentro del "triple constraint".

Para objetivar una eventual medición de la calidad desde estas tres perspectivas, se cree que los equipos que trabajan en instalaciones con plataformas de "Concurrent Versions System" o CVS (particularmente GitHub) generan una base de metadata explotable para responder las preguntas de investigación de este trabajo. [11]

El presente trabajo describe un estudio sistemático de la literatura con la finalidad de mapear las métricas de Calidad relacionadas a las Personas, Producto y Proceso aplicadas o factibles de ser aplicadas en un contexto de GitHub. Deliberadamente hemos dejado fuera del foco de estudio a toda publicación que se enfoque en métricas de calidad pero que no puedan basarse en el uso de metadata de GitHub. Hemos elegido GitHub debido a la amplia adopción de este CVS dentro de la zona geográfica en donde se realizó el estudio, lo que facilitará un futuro trabajo de campo en una corporación. Con esto identificamos las siguientes necesidades:

1. Objetivar el desempeño o calidad de un desarrollador basándose en el registro de su actividad en el repositorio de metadata de GitHub y no solamente en el criterio de quienes supervisan las tareas (RQ1)

2. Buscar huellas de las acciones deliberadas de los desarrolladores en GitHub que permitan determinar la calidad del producto, como comentarios, criterios explícitos, problemas resueltos, volumen de código y otros elementos (RQ2)

3. Independizar la metodología de desarrollo y la calidad de producto del resultado de la gestión del proyecto, extrayendo de GitHub elementos que permitan medir la calidad del proceso, dentro del ciclo de vida de un producto de software (RQ3)

4. Categorizar y tipificar los resultados de las preguntas anteriores obteniendo una visión integral de Calidad de las 3Ps (RQ4)

Por lo tanto, se plantean las siguientes preguntas de investigación (RQ):

RQ1: ¿Cómo se mide la Calidad de cada desarrollador (Persona) de en un equipo de trabajo (team) dentro de un entorno de desarrollo soportado por GitHub?

RQ2: ¿Cómo se mide la calidad del Producto (aplicativo o paquete de software) desarrollado por un equipo de trabajo dentro de un entorno de desarrollo soportado por GitHub?

RQ3: ¿Cómo se mide la calidad del Proceso de desarrollo en equipos soportados por GitHub?

RQ4: ¿Cuáles son las métricas de 3Ps (Persona – Proceso - Producto) aplicadas en la industria y que se puedan calcular con los datos contenidos en GitHub?

Este estudio sistemático es un primer paso de un objetivo más ambicioso que es la construcción de un modelo de calidad para la industria, basado en la medición automática de métricas de Persona, Producto, Proceso.

El artículo se organiza de la siguiente forma: introducimos el contexto de la investigación tanto desde el punto de vista teórico como su aplicación práctica (Sección 2), planteamos la metodología de investigación (Sección 3), los resultados (Sección 4) y el correspondiente análisis de estos (Sección 5). Finalizamos con una conclusión y un planteo de trabajos futuros (Sección 6).

2 Contexto de la investigación

Algunas prácticas como la Programación Interactiva con Trabajo Pendiente o la Programación a Demanda requieren de una enfoque ágil, reactivo y tendiente a mejores resultados progresivos. Para objetivar este concepto, es necesario basarse en datos más exactos y que resulten de la actividad de desarrollo dentro de un entorno tecnológico, sin interpretaciones o consideraciones subjetivas e/o incompletas.
El desarrollo de software colaborativo ha impulsado la construcción de herramientas que permitan alojar en forma eficiente y confiable las versiones de aplicaciones, cuando éstas tienen un gran número de archivos de código fuente, de tal forma que facilite el desarrollo de las aplicaciones y su posterior mantenimiento. Ejemplo de este tipo de herramientas es GitHub la cual está desarrollada con el controlador de versiones Git, o los conocidos “Concurrent Version System” (CVS). Este tipo de herramientas de código abierto están coleccionando una gran cantidad de datos relativos a las personas involucradas en el desarrollo, al código que alojan o al proceso de desarrollo en que estas personas se encuentran trabajando. Surge como una consecuencia natural la posibilidad de analizar los datos registrados y utilizarlos en la toma de decisiones [1]. Otra tendencia a destacar es que la ciencia de datos se está utilizando como una herramienta introspectiva para evaluar la productividad y la calidad del software de la organización [2].

Se ha acuñado el término “software analytics” el cual es entendido como la capacidad de permitir a los profesionales de software realizar exploraciones y análisis de datos para obtener información perspicaz y útil para tareas basadas en datos sobre software y servicios. La idea de software analytics es aprovechar cantidades potencialmente grandes de datos para obtener información real y procesable sobre el ciclo de vida del software [3].
El análisis de software tiene como objetivo obtener información perspicaz y útil de los artefactos de software que ayudan a los profesionales a realizar tareas relacionadas con el desarrollo, los sistemas y el uso de software [4].
Sin embargo, en los proyectos de desarrollo de software, las personas (desarrolladores, evaluadores, analistas) son el pilar más importante, pero muy difíciles de modelar. Es inevitable que nos movamos a un modelo que considere Producto, Proceso y Personas (3Ps), donde los repositorios de códigos analizados tradicionalmente deben vincularse con otros repositorios orientados al cliente, como registros de operaciones, transcripciones de llamadas de soporte al cliente, publicaciones de blog, reseñas de videos, etc. [4]
En gran medida, el análisis se refiere a qué proyectos de software pueden aprender de sí mismos. El análisis de software es un análisis de datos de software para gerentes e ingenieros de software con el objetivo de capacitar a los individuos y equipos de desarrollo de software para obtener y compartir información de sus datos para tomar mejores decisiones con el fin de mejorar en las 3 dimensiones “P”. El avance que se ha realizado es la comprensión de que en lugar de buscar modelos globales, el enfoque ahora está en los métodos locales: un modelo global no cubriría todos los proyectos de software. [1]
La práctica de análisis de software aplicado a las 3Ps podría entenderse bien en organizaciones con muchos equipos de desarrollo de software en paralelo, en donde la falta de un entorno de trabajo físico común desafía la tarea de control de proyectos e integración, a la vez que exige un gran trabajo de recopilación de datos no masivos pero oportunos, variables y de difícil interpretación, dado que cuanto más detallado es el registro de eventos en los repositorios de GitHub, mayor valor oculto podríamos extraer. A esto podríamos sumar la dificultad de trabajar con un mismo estándar y reglas de calidad sin prohibir la creatividad y talento personal de los desarrolladores. Es así como resulta imprescindible conocer una organización desde su estrategia corporativa, misión y visión, políticas de desarrollo, estándares, procesos, roles y capacidades de las personas, así como atributos de los productos de software que desarrollarán, con el fin de diseñar un modelo de calidad que responda a las expectativas organizacionales pero que su construcción y población sea factible a través de la ingesta de datos basada en el contenido de GitHub.

3 Metodología de trabajo

La revisión sistemática fue llevada a cabo por los autores de este artículo y estuvo basada en las pautas, recomendaciones y procedimientos propuestas por B. Kitchenham particularmente destinadas a la Ingeniería de Software [5]. Esta a su vez se basó en el Manual Cochrane de revisiones sistemáticas [6]. Se describe a continuación el protocolo de revisión sistemática que se determinó para esta investigación. El proceso de

investigación está determinado por la aplicación de los siguientes pasos: Criterios de Búsqueda o Elegibilidad, Fuentes de Búsqueda para Identificación de Estudios, Criterios de Selección de Estudios, Método de Selección de Estudios, Evaluación de la calidad de las publicaciones, Estrategia para la extracción de datos.

A continuación se describen los pasos requeridos para la construcción de términos o argumentos de búsqueda: a. se definió el marco teórico que incluye los términos relacionados con métricas de calidad y productividad asociados a Personas / Procesos / Productos y el concepto de metadata asociado a Git o GitHub como CVS; b. se utilizaron palabras de investigación para encontrar términos principales, intervenciones en la población y resultados; c. para los términos encontrados se identificaron palabras alternativas y/o sinónimos; d. se identificaron las palabras clave en diferentes artículos y publicaciones dentro del conjunto determinado en paso anterior; e. se construyeron argumentos de búsqueda usando aperadores booleanos de conjunción (And, Or); f. se identificaron publicaciones, resultados o estudios sistemáticos sobre temas similares a este estudio cuya lista, ya depurada luego de una primera lectura, aparece en link2.

La cadena de búsqueda genérica que refleja la búsqueda realizada es la siguiente: TS=(”Application Quality Metrics”) OR TS=(”Calidad de Software”) OR TS=(CVS) OR (TS=(CVS) AND TS=(Developer)) OR TS=(“Developer Quality”) OR TS=(“Developer Quality Metrics”) OR TS=(“Development Quality Metrics”) OR (TS=(Development Process Quality”) AND TS=(GitHub)) OR TS=(”GitHub Metadata”) OR TS=(”Software Development Metrics”) OR (TS=(Métricas) AND TS=(Calidad) AND TS=(Software)) OR TS=(“Product Quality”) OR TS=(“Software Metrics”) OR TS=(“Software Quality Metrics”).

En cuanto a las fuentes de información, se realizaron búsquedas en diferentes bases de datos, seleccionando artículos relevantes en la ventana de tiempo desde 2014 a 2021 en el ámbito regional, nacional e internacional. La ventana de tiempo a analizar fue definida sobre la base de la evolución que ha tenido GitHub en su uso y aceptación por las comunidades de desarrolladores. Hasta 2014 no se han realizado estudios que describan la calidad y propiedades de los datos disponibles en GitHub [7].

En la búsqueda, se incluyeron las bases de datos o bibliotecas digitales detalladas en Google Scholar (URL: Scholar.google.com), Science Direct (URL: sciencedirect.com), SpringerLink (URL: link.springer.com), IEEE Xplore (URL: ieeexplore.ieee.org) y ACM (URL: dl.acm.org).

Se seleccionaron los estudios teniendo en cuenta criterios de inclusión y exclusión con el fin de considerar solo aquellos que sean relevantes para las preguntas de investigación.

El criterio de selección consideró todos los artículos /reportes /tesis /capítulos de libros relacionados de manera directa con la temática, en idioma inglés. Los estudios considerados son descriptivos, experimentales, cualitativos, y/o revisiones sistemáticas. Se preseleccionaron todos los artículos y publicaciones que corresponden a la Calidad y/o Productividad de Persona/Producto/Proceso dentro de equipos de desarrollo de software, estén estos equipos en un entorno de trabajo soportados por CVS o no. Se preseleccionaron todos los artículos y publicaciones que traten sobre metadata y/o estructura interna de los sistemas de CVS como GitHub.

El proceso de selección de estudios implicó: a) la lectura del título, resumen y palabras clave (contraste con los criterios de inclusión); b) la lectura del título, resumen, introducción y conclusiones (si cumple con los criterios); c) la lectura del artículo completo. Si cumple criterios sobre las dos temáticas a abordar: Métricas de Calidad y Desarrollo soportado por GitHub; c) la lectura parcial del artículo, cuando cumple criterios sobre una sola de las temáticas a abordar.

Estrategia para la extracción de datos. Una vez realizada la selección de estudios, se completó una lista de toda la literatura considerada con detalle de los motivos de inclusión / exclusión. La cantidad de artículos que cumplieron al menos un criterio de selección fue 74; de estos, 34 fueron los que se consideraron como referencias válidas para el estudio sistemático. Se procedió luego al registro de las métricas descritas en los trabajos y a un ordenamiento y clasificación de estas, a la vez que se descartaron métricas repetidas. De un total de 235 métricas seleccionadas en primera instancia, luego de la revisión y selección final excluyeron 27, quedando las 208 métricas.

4. Resultados

La lista de métricas completa que surgió del estudio sistemático de la literatura se puede consultar en el siguiente link. En el presente artículo mostramos, a modo de resumen, algunas identificadas y clasificadas para facilitar el análisis de éstas.

Del análisis de contenido de la literatura preseleccionada (34 artículos) solo 19 artículos contenían detalles de métricas específicas, algunas de las cuales se repetían con el mismo o distinto nombre. Estos trabajos corresponden a las siguientes referencias: [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26]. Una dificultad encontrada durante el análisis fue la taxonomía y tipología de las métricas propuestas por los autores, las cuales se ajustaban a objetivos distintos al del presente trabajo; por este motivo o bien por tratarse el artículo con otro foco más que proponer métricas, 15 fuentes no aportaron al objetivo de esta investigación.

De las métricas que fueron halladas y consideradas en la bibliografía consultada, se seleccionaron las que están relacionadas con conceptos de Calidad de Persona, Proceso y/ o Producto en contextos de GitHub. De éstas hemos extractados las que a nuestro juicio podrían tener valores determinados mediante una extracción y eventual cálculo posterior de datos crudos existentes en el repositorio de GitHub.

Con los pasos anteriores realizados estamos en condiciones de responder las preguntas de investigación como sigue:

RQ1: ¿Cómo se mide la Calidad de cada desarrollador (Persona) de en un equipo de trabajo (team) dentro de un entorno de desarrollo soportado por GitHub?

Las métricas propuestas por los autores consultados se basan en el registro e individualización de los aportes de cada desarrollador que deja rastros en los repositorios de GitHub. Los resultados corresponden al análisis de algunos datos recuperables de la

metadata toda vez que una persona registra cambios o nuevos aportes a un desarrollo (Commit). Así, es posible determinar la cantidad de líneas de código, el tiempo que un desarrollador participa en un proyecto, el tiempo que un desarrollador tarda en cerrar un problema registrado (issue), la cantidad de aportes sin errores, su relación de cantidad con otros desarrolladores del mismo proyecto y proyectos en los que participa cada persona. Otras métricas propuestas muestran la relación entre la Persona y la Organización en donde trabaja, tales como la complejidad organizacional y la interacción social entre desarrolladores. Se determinaron 17 (diecisiete) métricas relacionadas con la Calidad de Persona.

Tabla 1. Métricas de Calidad de Persona en Contextos CVS

Contribution	Experience	Ownership	Quality
Skill Communication	Unresolved Items	# Branches	# Closed Issues
# Closed Pull Req.	# Commits	# Concurrent Project by person	# Edited Lines
Social Interaction between Developers	Number of Open Pull Request	Organizational Complexity	
# Open Issues	# Open Edited Issues		

RQ2: ¿Cómo se mide la calidad del Producto (aplicativo o paquete de software) desarrollado por un equipo de trabajo dentro de un entorno de desarrollo soportado por GitHub?

De las métricas propuestas por los autores se han individualizado 136 (ciento treinta y seis). Los autores consultados han asociado algunas métricas de acuerdo con categorías (distintas entre autores) que permiten establecer diferentes aspectos de la calidad de un Producto de Software como Arquitectura, Confiabilidad, Herencia, Testing, Acoplamiento, Clases, Mantenibilidad, Tamaño. Algunas métricas mencionadas por los autores no están claramente tipificadas o vinculadas a la algún aspecto Calidad. , tal es el caso de Popularidad, el Consumo de Energía, Volatilidad de Requerimientos, etc.

Desde el punto de vista Arquitectura. las dos métricas encontradas definen la complejidad arquitectónica. Una de ellas, Costo de Propagación [13] determina la proporción de archivos directa o indirectamente vinculados entre sí (por ejemplo, cuando se realizan llamadas de unos a otros). La segunda métrica, Core Size, tiene relación con la interdependencia de los componentes entre sí; estas dos métricas están fuertemente vinculadas a la densidad de defectos, la productividad del desarrollador y su retención [13].

Las métricas encontradas sobre la Confiabilidad del Producto están relacionadas con los eventuales problemas, defectos, arreglos, problemas reportados por uso y líneas de prueba de código. Estas métricas se pueden ver en la Tabla 2.

Tabla 2. Métricas de Confiabilidad de Productos en Contextos CVS

# Bugs Reports	# Defects	# Fixes
# Forks	# Opened Issues	# Test Lines of Code

Las métricas sobre Herencia pueden indicar problemas de abstracción y mantenimiento de software; tal es el caso de las que se propone en la Tabla 3.

Tabla 3. Métricas de Herencia de Productos en Contextos CVS

# Overriden Methods	Specialization Index	Depth of Inheritance Tree	# Ancestors
# Children	# Descendants	# Parents	

Existen autores que han categorizado métricas de calidad de producto basados en resultados de pruebas Unitarias, de Integración, de Regresión [21]. La perspectiva de algunos investigadores podría relacionar la deuda técnica (Number of TD) con la Mantenibilidad [20]. Varios autores han determinado la importancia de la Popularidad de un producto (particularmente para desarrollo de código abierto) como la cantidad de estrellas asignadas por los colaboradores [9], [10], [12], [11]. Ver lista completa en link.

Existe un conjunto de métricas de calidad de producto de software el cual es el resultado de una investigación sistemática previa reciente [21] (ver lista completa en link). Dicha investigación expone los resultados basados en el hallazgo / definición de métricas de Producto, independientemente de que se trate o no sobre productos desarrollados en entornos GitHub. Sin embargo, los valores de algunas de ellas podrían ser calculados extrayendo datos / metadatos de un sistema de control de versiones. Estas métricas, que no han sido categorizadas en las tablas previamente detalladas, se muestran en link.

La interdependencia entre módulos de software en cuanto a la forma y el nivel determina el acoplamiento como una medida de qué tan cercanamente están conectadas dos rutinas o módulos. Un bajo acoplamiento implica independencia por lo que resulta una buena métrica para determinar la calidad de Producto en cuanto al mantenimiento, reutilización de código, propagación de errores y minimización de esfuerzos de modificaciones al tener que cambiar un solo módulo. Estas métricas son: abstractness, coupling (Afferent, between object classes, between object classes inverse, between objects, efferent), Instability, normalize distance from mail sequence. Ver tabla en link.

Algunos autores han asociado métricas a la mantenibilidad. Estás métricas miden los comentarios y/o líneas de documentación que se agregan al código, elementos que facilitan el entendimiento del código y consecuentemente su mantenimiento eficaz y eficiente en el insumo del tiempo. Ejemplos de estas métricas son: API documentation, code of comment, comment ratio, # undocumented API, entre otras.

Los autores consultados asocian a la complejidad el tamaño del Producto: cuanto más líneas de código, métodos, atributos, interfases, paquetes, parámetros y otros recursos de programación, mayor es la complejidad asociada. Algunas de estas métricas son: comment density, logical lines of code, nesting level, # attributes, # Interfaces, entre otros. Ver tabla completa en link.

Se definieron métricas vinculadas con la calidad de Producto de Software, sin una clara subclasificación por parte de los autores consultados. Por ejemplo: Attribute complexity, average coupling factor, commit entropy, commit frequency, customer perception calculate, customer satisfaction, entre otras. Ver tabla completa en link

RQ3: ¿Cómo se mide la calidad del Proceso de desarrollo en equipos soportados por GitHub?

El proceso de desarrollo está vinculado fuertemente a la metodología utilizada y a la naturaleza del Producto a construir. Así es fácil pensar en grandes diferencias entre desarrollo colaborativo de software abierto / libre y los desarrollos de productos propios de una empresa u organización. Otra consideración importante es la plataforma y/o lenguaje utilizado para desarrollar. De todas maneras se han identificado 55 (cincuenta y cinco) métricas relacionadas por los autores a la Calidad del Proceso de desarrollo (ver lista completa en link). Estás métricas pueden referirse a la gestión del proyecto midiendo la actividad de los desarrolladores en cuanto a las líneas agregadas, modificadas, cambiadas (tanto en total, el máximo, el promedio), los problemas y su frecuencia [18], [8], [14]. Ejemplos de estas métricas son: average lines added / changed / deleted, issue frequency, minor contributor count, owner's experience, core contribution, quality of support, entre otras. La lista completa puede verse en link.

La colaboración en momento de Proceso de Desarrollo fue mencionada en tres artículos relacionando la Contribución Central [8], la eficiencia de desempeño de los desarrolladores [15] y la calidad de soporte como medida de Apoyo al proceso [12]. La lista de métricas categorizadas como de Colaboración aparece detallada en link.

Los procesos de desarrollo considerados por los autores consultados corresponden en su gran mayoría a desarrollos de software abiertos cuya gestión de proyecto es radicalmente distinta a los desarrollos bajo una misma área dentro de una organización. Sin embargo, la forma de calificar métricas sobre la confiabilidad, eficiencia de desempeño, usabilidad, mantenibilidad, satisfacción de usuario y la comunidad involucrada pueden tener una misma raíz dentro de los repositorios de los CVS utilizados. [10], [11], [12], [14], [15], [16], [21], [23]. En las tablas que figuran en link se muestran las métricas de proceso sin una clasificación específica tales como: CI flag, continuous integration, # attentive people, # commits per bug report, # pull request accepted, pase-based defect removal pattern, entre otras.

RQ4: ¿Cuáles son las métricas de 3Ps (Persona – Proceso - Producto) aplicadas en la industria y que se puedan calcular con los datos contenidos en GitHub?

Si bien Bird et al. [41] plantearon la necesidad de tres dimensiones (Persona, Producto Proceso), no se han encontrado estudios que propongan métricas y modo de gestionarlas considerando las 3Ps en un ambiente de desarrollo dentro de un ámbito industrial.

La calidad total en Ingeniería de Software podría gestionarse a través de la determinación de valores en las métricas definidas en las preguntas de investigación anteriores. Lo que podría ser más complejo es interrelacionar las tres distintas agrupaciones. La gestión de las métricas tiene real valor cuando permiten tomar decisiones o acciones concretas para alcanzar y/o mantener un nivel de calidad mínimo necesario para satisfacer las aspiraciones de las empresas. Algunas de las acciones esperadas pueden ser la asignación de recursos adecuados, conformación de equipos de trabajo, interacción entre los desarrolladores. Es difícil analizar tal volumen de métricas en un entorno cambiante permanentemente en paralelo con la gestión propia del proyecto, la cual intenta mantener el resultado dentro de un tiempo, costo y alcance predefinido [41].

De un análisis integral de todas las métricas seleccionadas vimos que es posible encontrar sinónimos (los cuales fueron depurados) pero también una visión distinta de la categorización y/o tipificación de las métricas según el foco analítico del autor. Un ejemplo claro fue la interpretación de la Cantidad de Líneas Totales. Desde un punto de vista Producto puede darnos una medida de Volumen o Tamaño; desde la perspectiva de Proceso y relacionándola con el tiempo nos da una medida del Ciclo de Vida de Desarrollo de Software, la Metodología y otros aspectos que se desprende de estos: Efectividad y Eficiencia del Proceso, la Evaluación de Costos, entre otros.

5. Analisis de los resultados

El resultado de revisión sistemática de la literatura corresponde a una lista de métricas tipificadas según el elemento de las 3Ps al que se refiere. Como vimos, algunas métricas pueden ser utilizadas para medir la calidad de más de una dimensión por lo que deberán homogeneizarse al momento de catalogarlas con una visión holística considerando las 3Ps.

Al momento de registrar las métricas seleccionadas, se analizaron y definieron categorías en las que podrían agruparse de acuerdo con el criterio de los autores.

Si bien las preguntas no resultan complicadas de entender y planificar acciones para conseguir responderlas, igualmente simple es entender que es necesario contar con una fuente de datos objetivos que refleje con precisión la realidad en cuanto a la calificación de la calidad, de la manera que una organización y sus gestores la entiende.

Es justamente esta visión personalizada de cada empresa la que nos limita para construir un modelo general; sin embargo, creemos posible diseñar este modelo basado sobre algunas métricas formuladas y mencionadas en este trabajo, pero buscando que sus valores se basen en el cálculo sobre datos existentes en el repositorio de metadata de GitHub.

Para obtener un modelo más ajustado a la realidad se hace necesario un trabajo de campo previo sobre la base de entrevistas con responsables de empresas de desarrollo de productos de software, con el fin de determinar cuáles son sus inquietudes en esta materia. En base a las respuestas de los desarrolladores y considerando la recopilación realizada en este estudio se harán la propuesta de un modelo de calidad para cada empresa.

La búsqueda de la excelencia en el desarrollo de productos dentro de la Ingeniería de Software se ha perseguido desde los inicios de la práctica, sin embargo es necesario articular las necesidades reales de grandes empresas, la definición de métricas que reflejen con precisión y de manera objetiva esas necesidades conjuntamente con la disponibilidad de datos fuente reales existentes en GitHub que puedan ser explotados con técnicas y herramientas adecuadas para el control de la calidad de las 3Ps.

Algunos investigadores han publicado trabajos asociados a alguna de estas inquietudes por separado (preguntas de investigación), pero a la fecha, no se han encontrado publicaciones que traten este tema como un todo; es decir, la medición de la calidad para las 3Ps en la industria. Aun así, es interesante rescatar los esfuerzos individuales

para tomarlos como base para la investigación y plantear algunas métricas que son de amplia aceptación.

6. Conclusiones

Esta revisión sistemática de la literatura permitió catalogar un conjunto de 209 (doscientas doce) métricas asociadas a la calidad de alguno de los tres ejes previstos: Persona (18), Proceso (55), Producto (136), en entornos de desarrollo donde se trabaja con CVS. La categorización de estas métricas se relaciona con la visión particular de cada investigador o autor pero resulta interesante encontrar algunas métricas que han sido analizadas desde más de una perspectiva, siendo todas válidas.

El proceso dentro de un proyecto de desarrollo de un producto de software relaciona el esfuerzo y efectividad de desarrolladores con el producto resultante. Así, los tres ejes estudiados están íntimamente relacionados por lo que justifica el impacto de algunas métricas en más de un eje (doble categorización). Sin embargo, en las fuentes consultadas, no se encontró ningún trabajo que aborde la problemática de las 3Ps dentro de una misma instalación.

Este trabajo será considerado como la base de un proyecto futuro que busca definir un modelo de calidad de Persona, Proceso y Producto, basado en métricas calculadas automáticamente tomando como fuente objetiva de datos, la metadata de un CVS en una empresa.

Limitaciones y amenazas de validez. Somos conscientes de que nuestra investigación puede tener algunas limitaciones. Como los resultados se basan en investigaciones previas realizadas sobre contextos específicos (código abierto), la generalización de algunas métricas y definiciones encontradas podrían no representar a empresas privadas. Es por esto que completaremos este trabajo con una encuesta a empresas. Los argumentos de búsqueda podrían tener sinónimos desconocidos al momento de la investigación por lo que podría limitar la cantidad de artículos seleccionados. Para evitar esto, posteriormente a la búsqueda por palabras clave, se incluyeron otras palabras equivalentes y su traducción al español.

Referencias

[1] Menzies T, Zimmermann T. Software Analytics: So What? IEEE Softw 2013;30:31–7. https://doi.org/10.1109/MS.2013.86.

[2] Kim S, Whitehead, EJ, Zhang Y. Classifying Software Changes: Clean or Buggy? IIEEE Trans Software Eng 2008;34:181–96. https://doi.org/10.1109/TSE.2007.70773.

[3] Abdellatif TM, Capretz LF, Ho D. Software Analytics to Software Practice: A Systematic Literature Review. 2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering, Florence, Italy: IEEE; 2015, p. 30–6. https://doi.org/10.1109/BIGDSE.2015.14.

[4] Zhang L, Zou Y, Xie B. A scalable crawler framework for FLOSS data. Proceedings of the 5th Asia-Pacific Symposium on Internetware, Changsha China: ACM; 2013, p. 1–7. https://doi.org/10.1145/2532443.2532454.
[5] Kitchenham B. Procedures for Performing Systematic Reviews. Keele University Technical Report TR/SE-0401 2004:33.
[6] Higgins JPT, Thomas J, Chandler J, Cumpston M, Li T, Page MJ, et al., editors. Cochrane Handbook for Systematic Reviews of Interventions. Cochrane Handbook for Systematic Reviews of Interventions. 1st ed., Wiley; 2019. https://doi.org/10.1002/9781119536604.fmatter.
[7] Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D. The promises and perils of mining GitHub. Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014, Hyderabad, India: ACM Press; 2014, p. 92–101. https://doi.org/10.1145/2597073.2597074.
[8] Munaiah N, Kroh S, Cabrey C, Nagappan M. Curating GitHub for engineered software projects. Empir Software Eng 2017;22:3219–53. https://doi.org/10.1007/s10664-017-9512-6.
[9] Cosentino V, Canovas Izquierdo JL, Cabot J. A Systematic Mapping Study of Software Development With GitHub. IEEE Access 2017;5:7173–92. https://doi.org/10.1109/ACCESS.2017.2682323.
[10] Jarczyk O, Jaroszewicz S, Wierzbicki A, Pawlak K, Jankowski-Lorek M. Surgical teams on GitHub: Modeling performance of GitHub project development processes. Information and Software Technology 2018;100:32–46. https://doi.org/10.1016/j.infsof.2018.03.010.
[11] Vasilescu B, Yu Y, Wang H, Devanbu P, Filkov V. Quality and productivity outcomes relating to continuous integration in GitHub. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Bergamo Italy: ACM; 2015, p. 805–16. https://doi.org/10.1145/2786805.2786850.
[12] Jarczyk O, Gruszka B, Jaroszewicz S, Bukowski L, Wierzbicki A. GitHub Projects. Quality Analysis of Open-Source Software. In: Aiello LM, McFarland D, editors. Social Informatics, vol. 8851, Cham: Springer International Publishing; 2014, p. 80–94. https://doi.org/10.1007/978-3-319-13734-6_6.
[13] Ludwig J, Xu S, Webber F. Compiling static software metrics for reliability and maintainability from GitHub repositories. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB: IEEE; 2017, p. 5–9. https://doi.org/10.1109/SMC.2017.8122569.
[14] McDonald N, Goggins S. Performance and participation in open source software on GitHub. CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13, Paris, France: ACM Press; 2013, p. 139. https://doi.org/10.1145/2468356.2468382.
[15] Gyimesi P, Gyimesi G, Tóth Z, Ferenc R. Characterization of Source Code Defects by Data Mining Conducted on GitHub. In: Gervasi O, Murgante B, Misra S, Gavrilova ML, Rocha AMAC, Torre C, et al., editors. Computational Science and Its Applications -- ICCSA 2015, vol. 9159, Cham: Springer International Publishing; 2015, p. 47–62. https://doi.org/10.1007/978-3-319-21413-9_4.
[16] Liao Z, He D, Chen Z, Fan X, Zhang Y, Liu S. Exploring the Characteristics of Issue-Related Behaviors in GitHub Using Visualization Techniques. IEEE Access 2018;6:24003–15. https://doi.org/10.1109/ACCESS.2018.2810295.

[17] Qiu Y, Zhang W, Zou W, Liu J, Liu Q. An Empirical Study of Developer Quality. 2015 IEEE International Conference on Software Quality, Reliability and Security - Companion, Vancouver, BC, Canada: IEEE; 2015, p. 202–9. https://doi.org/10.1109/QRS-C.2015.33.
[18] Wu Y, Yang Y, Zhao Y, Lu H, Zhou Y, Xu B. The Influence of Developer Quality on Software Fault-Proneness Prediction. 2014 Eighth International Conference on Software Security and Reliability, San Francisco, California, USA: IEEE; 2014, p. 11–9. https://doi.org/10.1109/SERE.2014.14.
[19] de Bassi PR, Wanderley GMP, Banali PH, Paraiso EC. Measuring Developers' Contribution in Source Code using Quality Metrics. 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), 2018, p. 39–44. https://doi.org/10.1109/CSCWD.2018.8465320.
[20] Salamea MJ, Farré C. Influence of Developer Factors on Code Quality: A Data Study. 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2019, p. 120–5. https://doi.org/10.1109/QRS-C.2019.00035.
[21] Colakoglu FN, Yazici A, Mishra A. Software Product Quality Metrics: A Systematic Mapping Study. IEEE Access 2021;9:44647–70. https://doi.org/10.1109/ACCESS.2021.3054730.
[22] Calikli G, Bener A. An algorithmic approach to missing data problem in modeling human aspects in software development. Proceedings of the 9th International Conference on Predictive Models in Software Engineering, Baltimore Maryland USA: ACM; 2013, p. 1–10. https://doi.org/10.1145/2499393.2499398.
[23] Rashid J, Mahmood T, Nisar MW. A Study on Software Metrics and its Impact on Software Quality n.d.;24:14.
[24] Belachew EB. Analysis of Software Quality Using Software Metrics. IJCSA 2018;8:11–20. https://doi.org/10.5121/ijcsa.2018.8502.
[25] Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D. An in-depth study of the promises and perils of mining GitHub. Empir Software Eng 2016;21:2035–71. https://doi.org/10.1007/s10664-015-9393-5.
[26] Qureshi MRJ, Qureshi WA. Evaluation of the Design Metric to Reduce the Number of Defects in Software Development. IJITCS 2012;4:9–17. https://doi.org/10.5815/ijitcs.2012.04.02.
[27] Cosentino V, Luis J, Cabot J. Findings from GitHub: methods, datasets and limitations. Proceedings of the 13th International Conference on Mining Software Repositories, Austin Texas: ACM; 2016, p. 137–41. https://doi.org/10.1145/2901739.2901776.
[28] Casalnuovo C, Suchak Y, Ray B, Rubio-González C. GitcProc: a tool for processing and classifying GitHub commits. Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara CA USA: ACM; 2017, p. 396–9. https://doi.org/10.1145/3092703.3098230.
[29] Gousios G, Vasilescu B, Serebrenik A, Zaidman A. Lean GHTorrent: GitHub data on demand. Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014, Hyderabad, India: ACM Press; 2014, p. 384–7. https://doi.org/10.1145/2597073.2597126.
[30] Biazzini M, Baudry B. "May the fork be with you": novel metrics to analyze collaboration on GitHub. Proceedings of the 5th International Workshop on

Emerging Trends in Software Metrics - WETSoM 2014, Hyderabad, India: ACM Press; 2014, p. 37–43. https://doi.org/10.1145/2593868.2593875.
[31] Borle NC, Feghhi M, Stroulia E, Greiner R, Hindle A. Analyzing the effects of test driven development in GitHub. Empir Software Eng 2018;23:1931–58. https://doi.org/10.1007/s10664-017-9576-3.
[32] Prana GAA, Treude C, Thung F, Atapattu T, Lo D. Categorizing the Content of GitHub README Files. Empir Software Eng 2019;24:1296–327. https://doi.org/10.1007/s10664-018-9660-3.
[33] Bissyande TF, Lo D, Jiang L, Reveillere L, Klein J, Traon YL. Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub. 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), Pasadena, CA, USA: IEEE; 2013, p. 188–97. https://doi.org/10.1109/ISSRE.2013.6698918.
[34] Kenett RS. Implementing SCRUM using Business Process Management and Pattern Analysis Methodologies. DRMJ 2013;2:29–48. https://doi.org/10.17708/DRMJ.2013.v02n02a03.
[35] Tosun A, Bener A, Turhan B, Menzies T. Practical considerations in deploying statistical methods for defect prediction: A case study within the Turkish telecommunications industry. Information and Software Technology 2010;52:1242–57. https://doi.org/10.1016/j.infsof.2010.06.006.
[36] Mladenova T. Software Quality Metrics – Research, Analysis and Recommendation. 2020 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria: IEEE; 2020, p. 1–5. https://doi.org/10.1109/ICAI50593.2020.9311361.
[37] Chatziasimidis F, Stamelos I. Data collection and analysis of GitHub repositories and users. 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), Corfu, Greece: IEEE; 2015, p. 1–6. https://doi.org/10.1109/IISA.2015.7388026.
[38] Farah G, Tejada JS, Correal D. OpenHub: a scalable architecture for the analysis of software quality attributes. Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014, Hyderabad, India: ACM Press; 2014, p. 420–3. https://doi.org/10.1145/2597073.2597135.
[39] Rodriguez-Bustos C, Aponte J. How Distributed Version Control Systems impact open source software projects. 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), Zurich: IEEE; 2012, p. 36–9. https://doi.org/10.1109/MSR.2012.6224297.
[40] Posnett D, D'Souza R, Devanbu P, Filkov V. Dual ecological measures of focus in software development. 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA: IEEE; 2013, p. 452–61. https://doi.org/10.1109/ICSE.2013.6606591.
[41] Bird J, Menzies T, Zimmermann T, editors. The art and science of analyzing software data. Waltham, Massachusetts: Morgan Kaufmann; 2015.