Un Conjunto de Métricas e Indicadores para el Abordaje de la Productividad, Confiabilidad y Mantenibilidad en un Software de Gestión

A Set of Metrics and Indicators for the Approach to Productivity, Reliability and Maintainability in a Management Software

Presentación: 13 y 14 de septiembre de 2023

Santiago Castillo Elías
Departamento de Informática, Facultad de Ciencias Físico-Matemáticas y Naturales. Universidad Nacional de San Luis.
santicastilloelias@gmail.com

Carlos Salgado
Departamento de Informática – Facultad de Ciencias Físico-Matemáticas y Naturales – Universidad Nacional de San Luis
csalgado@unsl.edu.ar

Mario Peralta
Departamento de Informática – Facultad de Ciencias Físico-Matemáticas y Naturales – Universidad Nacional de San Luis
mperalta@unsl.edu.ar

Alberto Sánchez
Departamento de Informática – Facultad de Ciencias Físico-Matemáticas y Naturales – Universidad Nacional de San Luis
alfanego@unsl.edu.ar

Corina Abdelahad
Departamento de Informática – Facultad de Ciencias Físico-Matemáticas y Naturales – Universidad Nacional de San Luis
corina.abde@gmail.com

Javier Saldarini
Facultad Regional San Francisco – Universidad Tecnológica Nacional
saldarinijavier@gmail.com

Claudio Carrizo
Facultad Regional San Francisco – Universidad Tecnológica Nacional
cjcarrizo77@gmail.com

Resumen

Los modelos de calidad junto con las métricas e indicadores pueden, por ejemplo, ser utilizados por desarrolladores, adquirientes, personal de aseguramiento y de control de la calidad y evaluadores independientes, particularmente aquellos responsables de especificar y evaluar la calidad del producto de software. También se explicita que el alcance de la aplicación de los modelos de calidad incluye el apoyo de la especificación y la evaluación de software y los sistemas informáticos intensivos de software desde perspectivas diferentes de los asociados con su adquisición, requisitos, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento y control de la calidad y auditoría.

En base a lo anterior, se ha definido un modelo de calidad para poder estudiar, analizar y evaluar las características como la mantenibilidad, productividad, confiabilidad que permitan a la gerencia y líderes de proyectos tomar decisión sobre el software de gestión de la empresa basado en las características y subcaracterísticas que son deseables o esperables por los usuarios del producto software en base a las necesidades definidas en el modelo de calidad propuesto.

Palabras clave: Métricas e Indicadores, Modelo de Calidad, Productividad, Eficiencia, Toma de Decisión.

Abstract

Quality models together with metrics and indicators may, for example, be used by developers, acquirers, quality assurance and quality control personnel, and independent evaluators, particularly those responsible for specifying and evaluating software product quality. It is also made explicit that the scope of application of the quality models includes the support of the specification and evaluation of software and software-intensive computing systems from perspectives other than those associated with their acquisition, requirements, development, use, evaluation, support, maintenance, quality assurance and control, and auditing.

Based on the above, a quality model has been defined to be able to study, analyze and evaluate characteristics such as maintainability, productivity, reliability that allow management and project leaders to make decisions about the company's management software based on the characteristics and sub-characteristics that are desirable or expected by the users of the software product based on the needs defined in the proposed quality model.

Keywords: Metrics and Indicators, Quality Model, Productivity, Efficiency, Decision Making.

Introducción

Uno de los componentes principales de los sistemas informáticos lo constituye el software, y la calidad de éste tendrá influencia directa en el sistema que lo contiene.

La calidad del software es presentada en la literatura a través de distintas definiciones, una de ella, por ejemplo, es la expresada en (Pressman 2021), donde a la calidad de software se la define como el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados, y de las características implícitas que se espera de todo software desarrollado profesionalmente.

También vemos que en ISO/IEC 25000 (ISO/IEC 2014) se la define como el grado en que el producto software satisface las necesidades expresadas o implícitas, cuando es usado bajo condiciones determinadas.

Existen distintos enfoques de la calidad del software, éstos pueden ser, Calidad a nivel proceso, Calidad a nivel de producto y Calidad en uso, para cada uno de estos enfoques existen distintos tipos de modelo de calidad de software que se pueden aplicar, según se especifica en (Callejas-Cuervo, Alarcón-Aldana et al. 2017).

Los Modelos de Calidad (MC), son instrumentos o artefactos específicamente diseñados y construidos para soportar la evaluación y selección de componentes de software. Permiten la definición estructurada de criterios de evaluación, la especificación de requerimientos, la descripción de componentes en relación a ellos y la identificación de desajustes de manera sistemática facilitando el proceso de evaluación y selección del software (Villalta and Carvallo 2015).

Este trabajo se centró en la definición de un modelo de calidad basado en la norma ISO/IEC 25000 (ISO 2015) para la mejora de los distintos procesos de negocio del sistema ChessERP, enfocado a distribuidoras de productos de consumo masivos como alimentos, bebidas, productos de cuidado personal, productos de limpieza, entre otros.

El sistema está adaptado para abordar los desafíos específicos de la industria de productos de consumo masivo, como la gestión de inventario, la gestión de stocks, la gestión de cadenas de suministro complejas, la administración de promociones y precios, la gestión de rutas de distribución, la gestión de múltiples fuerzas de ventas, la gestión de contabilidad incluido caja, bancos y cuentas corrientes, entre otros aspectos clave. Algunas características y funcionalidades del sistema ChessERP incluyen:

1. Gestión de la cadena de suministro: Facilita el seguimiento y control de los procesos de abastecimiento de materias primas, producción, logística y distribución de productos, permitiendo una gestión eficiente de la cadena de suministro.

2. Gestión de stocks: Consiste en planificar, organizar y supervisar de manera eficiente las existencias de mercancías o productos para garantizar que estén disponibles en el momento y lugar adecuados.
3. Gestión de promociones y precios: Permite administrar y controlar las promociones, descuentos y precios especiales aplicados a los productos de consumo masivo.
4. Gestión de rutas de distribución: Facilita la planificación y optimización de las rutas de distribución, asegurando una entrega eficiente y puntual de los productos.
5. Gestión de múltiples fuerzas de ventas: Administración y coordinación de equipos de ventas que operan en diferentes ubicaciones geográficas, segmentos de mercado o canales de distribución.

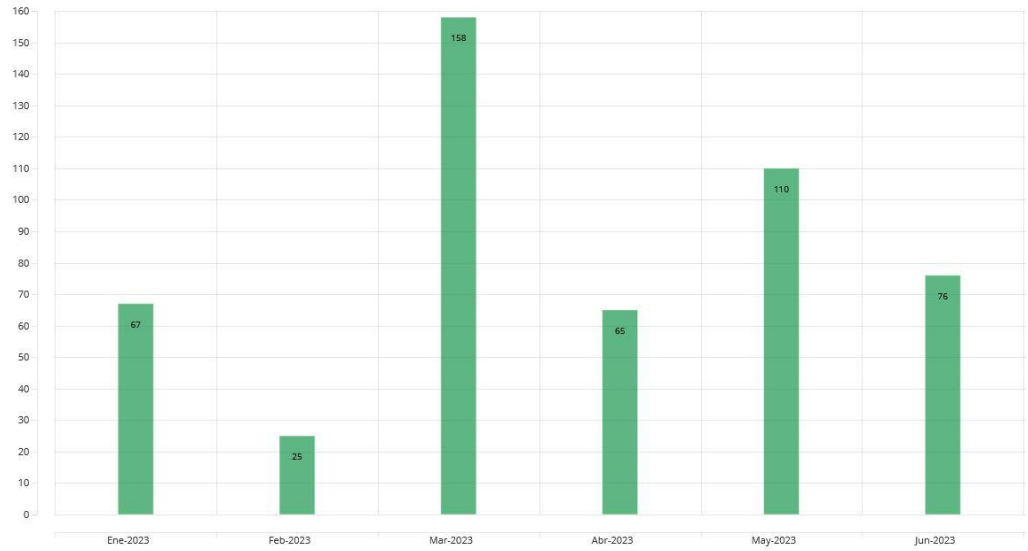
Modelo Propuesto

El modelo de calidad propuesto se centra en tres aspectos fundamentales: Productividad, Confiabilidad y Mantenibilidad. A continuación, se presentan ejemplos de cómo se definen estas características, así como algunas métricas relacionadas aplicadas en un caso de estudio:

Productividad: se hace referencia al uso de distintas metodologías, herramientas, etc. para medir la capacidad de un equipo de trabajo en un determinado contexto.

1. Métrica: *Evolución del número de tareas finalizadas durante un periodo de tiempo predefinido*: El objetivo es medir la cantidad de tareas que el equipo entregó en un período de tiempo predefinido. En el caso de estudio el equipo de trabajo utiliza metodologías ágiles por lo que las tareas o User Stories están estimadas con User Stories Points que miden la complejidad en base a un determinado pívot que se utiliza como referencia, el pívot utilizado por el equipo es un campo de texto simple y sin validaciones que equivale a un User Story Point y el periodo de tiempo predefinido o sprint es de un mes. En este caso se considera que una tarea está finalizada cuando la misma ya concluyó tanto las etapas de desarrollo como de testing unitario y funcional del mismo.

Resultados obtenidos:

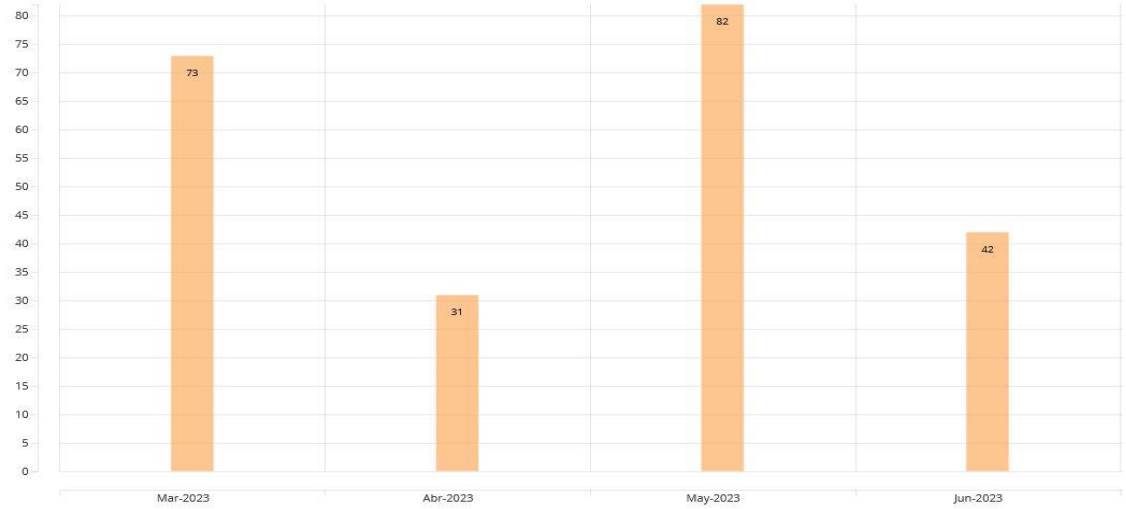


Durante el mes de marzo se finalizaron el mayor número de User Story Points de los meses analizados, seguido por el mes de mayo. Esto se debe a que en esos meses se finalizó con muchas User Story del mes anterior (febrero y abril) que se contabilizan como parte del mes en curso. Esto puede sugerir que en cada mes se debería desarrollar menos User Story Points para que no ocurra el carry over que se observa en marzo y en mayo.

Confiabilidad: se refiere a la capacidad de un sistema, producto o servicio para funcionar de manera predecible y sin problemas durante un período de tiempo determinado, cumpliendo con las expectativas y necesidades de los usuarios

1. Métrica: *Evolución en el número de bugs presentes en las tareas desarrolladas durante un periodo de tiempo preestablecido*: el objetivo es medir la cantidad de errores reportados por un equipo de QA sobre las tareas o User Story que aún no han sido liberadas en ambientes productivos durante un periodo de tiempo preestablecido. En este caso de estudio, como ya se mencionó, se utiliza como período de tiempo predefinido o sprint un mes.

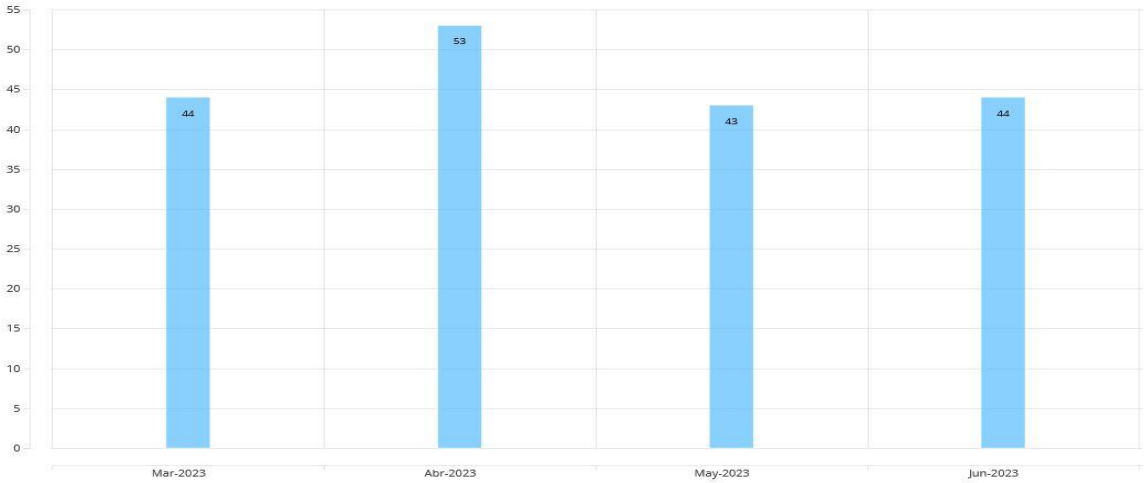
Resultados obtenidos:



Durante los meses de marzo y mayo se observa la mayor cantidad de bugs reportados sobre User Story que aún no están en ambientes productivos, esto se debe a que en esos meses fueron los que mayor cantidad de User Story Points se entregaron. Esto puede sugerir que en cada mes se debería desarrollar menos User Story Points para que no ocurra lo que se observa en los meses de marzo y mayo y se obtenga un número más estable.

2. Métrica: *Evolución en el número de bugs reportados por clientes en ambientes productivos*: el objetivo es medir la cantidad de errores reportados por los clientes sobre tareas o user story que han sido liberadas en ambientes productivos durante un periodo de tiempo preestablecido.

Resultados obtenidos:



Se observa un número estable de bugs reportados por clientes en ambientes productivos con un pico en el mes de abril, esto ocurre porque en ese mes se liberó una versión nueva del sistema. El escenario ideal es reducir el número de bugs reportados por un mismo número de clientes, pero esto es difícilmente realizable ya que el sistema evoluciona y agrega nuevas funcionalidades y características.

Mantenibilidad: se refiere a la facilidad realizar cambios, reparaciones, mejoras o adaptaciones en un sistema

1. Métrica: *Acoplamiento*: Mide la dependencia entre módulos. En este caso de estudio se analiza un módulo Front-End desarrollado utilizando el framework Angular, el mismo utiliza como lenguaje JavaScript.

Resultados obtenidos:

Uno de los módulos analizados, fue clientesModule, se analizó su dependencia con otros módulos del sistema contando la cantidad de llamadas a otros módulos del sistema en cada componente del módulo clientesModule:

- alias.component : 19 llamadas
- conjuntos.component: 0 llamadas
- fuerza-de-ventas.component: 0 llamadas
- jerarquia-de-marketing.component: 0 llamadas
- listas-de-precios.component: 0 llamadas
- segmentos-de-marketing.component: 0 llamadas
- clientes-formulario.component: 85 llamadas
- clientes-listado.component: 117 llamadas

Total, de 221 llamadas

Se observó un alto acoplamiento en las componentes de clientes-formulario.component y clientes-listado.component esto se debe a que dichas componentes son las que contienen la mayor cantidad de lógica. Las llamadas son a componentes presentes en otros módulos que realizan distintos procesamientos de datos. Se debe reducir al mínimo posible dichas llamadas ya que un acoplamiento alto significa que los módulos están fuertemente interconectados, lo que puede dificultar la modificación y la reutilización del código.

2. Métrica: *Cohesión*: Evalúa la relación entre las partes de un módulo. En este caso de estudio como ya se mencionó se analizó un módulo Front-End desarrollado utilizando el framework Angular, el mismo utiliza como lenguaje JavaScript.

Resultados obtenidos:

El módulo analizado es ***clientesModule***, el mismo está compuesto por ocho componentes, se analizaron las llamadas entre las componentes del módulo.

La componente ***jerarquia-de-marketing.component*** es llamada desde la componente segmentos-de-***marketing.component***. Las componentes ***alias.component***, ***conjuntos.component***, ***fuerza-de-ventas.component*** y ***fuerza-de-ventas.component*** son llamadas desde la componente ***clientes-formulario.component*** y esta componente es llamada desde ***clientes-listado.component***.

La componente ***clientes-formulario.component*** es la que llama al mayor número de componentes del módulo ya que es la componente que mayor lógica tiene del módulo. Dicha componente es un formulario compuesto por solapas, donde cada solapa representa una componente distinta que tiene su propio diseño y lógica por lo que posee una alta cohesión. Esto indica que las partes del módulo están fuertemente relacionadas y trabajan juntas para lograr un objetivo común. Es una propiedad importante ya que facilita el mantenimiento por la modularidad y también facilita la comprensión del código.

Conclusiones

En nuestro mundo de sobrecarga de información, el contexto determina la productividad. El modelo de calidad propuesto, junto con las métricas e indicadores, proporciona el contexto de las ventas, el marketing o las finanzas en la empresa, de modo que permite enfocarse en el problema principal, y en base a ello tomar decisiones. Para lograr tales parámetros se hace necesario conseguir un software eficiente y mantenible, es decir, que sea adaptable y ágil a las necesidades de la organización. Para ello, es necesario pensar en torno a distintos tipos de usuarios, principalmente consumidores y luego extender a los usuarios empresariales. Sin embargo, el negocio es colaborativo y se realiza en equipos. Por lo tanto, el software debe trabajar en un equipo que debe ser ágil y proactivo.

Así, en este trabajo se presenta un modelo de calidad centrado en tres aspectos fundamentales para toda empresa u organización: Productividad, Confiabilidad y Mantenibilidad, de manera que ayuden a la mejora de los procesos de negocio. En particular se instanció el modelo con su aplicación sobre el sistema de gestión ChessERP, obteniendo resultados satisfactorios y mejorando el proceso en general. Sugiriéndose, por ejemplo, en el caso de la productividad, que en cada mes se debería desarrollar menos User Story Points para que no ocurra el carry over para el siguiente sprint.

El implementar un software que cumplan las características de productividad, eficiencia y mantenibilidad es clave para mejorar no solo el rendimiento de los procesos de negocio, sino la rentabilidad de toda la firma en general. La definición de un modelo de calidad junto con un conjunto de métricas e indicadores es una herramienta que ayuda a manejar información más precisa, agilizar los procesos y tomar mejores decisiones estratégicas.

Cabe aclarar que, si bien el modelo se analizó sobre el sistema ChessERP que está orientado a distribuidores de consumo masivo, el modelo tiene la flexibilidad necesaria como para ser aplicado en otros contextos en los que se necesite analizar características como Seguridad, Satisfacción del Producto, Eficiencia y Flexibilidad utilizando las métricas definidas en el modelo propuesto.

Referencias

Callejas-Cuervo, M., A. C. Alarcón-Aldana, et al. (2017). "Modelos de calidad del software, un estado del arte." Entramado 13.

ISO (2015). ISO/IEC 25000. SQuaRE - System and Software Quality Requirements and Evaluation.

ISO/IEC (2014). ISO/IEC 25000. “SQuaRE - System and Software Quality Requirements and Evaluation”. http://iso25000.com.

Pressman, R. (2021). Ingeniería del Software: un enfoque práctico.

Villalta, A. and J. P. Carvallo (2015). " Modelos de calidad de software: Una revisión sistemática de la literatura." Maskana, CEDIA 2015.