JCSA 2022

V EDICIÓN | 24 Y 25 DE NOVIEMBRE

Jornadas de Calidad de Software y Agilidad

- Libro de Actas -

UNIVERSIDAD TECNOLÓGICA NACIONAL

AJEA
Actas de Jornadas y Eventos Académicos de UTN

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL RESISTENCIA

UNIVERSIDAD NACIONAL DE MISIONES

FaCENA - UNNE

Jornadas de Calidad de Software y Agilidad / Marcela Arias ... [et al.] ; contribuciones de Verónica A. Bollati ... [et al.] ; compilación de Verónica A. Bollati ; César J. Acuña ; Noelia Pinto ; coordinación general de César J. Acuña ; Verónica A. Bollati ; Noelia Pinto. - 1a ed. - Ciudad Autónoma de Buenos Aires : Universidad Tecnológica Nacional, 2023.
Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-950-42-0224-0

1. Software. I. Arias, Marcela. II. Bollati, Verónica A., colab. III. Acuña, César J., comp. IV. Pinto, Noelia, comp.
CDD 005.1207

V Jornadas de Calidad de Software y Agilidad

(UTN-UNaM-UNNE)

24 y 25 de Noviembre de 2022

Sede Virtual de la Facultad Regional Resistencia

Actas
Publicado en mayo de 2023

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

RESUMEN

Las Jornadas de Calidad de Software y Agilidad (JCSA) inician en el año 2017, bajo el nombre original de Jornadas de Calidad de Software, con la intención de difundir avances significativos en el campo de la Ingeniería del Software y, en particular, conceptos, métodos y herramientas que contribuyan a la Calidad de Software. Brindando un foro común donde la industria del software y las universidades puedan intercambiar experiencias, exponer necesidades y fomentar acciones conjuntas en el área. En el año 2021, las jornadas amplían su espectro, incorporando temas relacionados a la agilidad, e incluyendo la recepción de trabajos científicos y experiencias industriales.

En esta VI edición, las jornadas nos vuelven a encontrar en el escenario virtual de la Facultad Regional Resistencia donde tuvo lugar la exposición de trabajos científicos y experiencias emergentes de investigación, Talleres, Conferencias y Presentación de Experiencias Industriales, con la participación de integrantes de la Industria, Universidad y Estado.

Además de las actividades centrales de las jornadas, que se realizó de manera virtual, se realizaron, en total, cuatro talleres presenciales distribuidos en cada una de las sedes que participaron en la organización.

La conferencia inaugural denominada “Ciclo de vida del conocimiento en el desarrollo de software para la transformación digital”, estuvo a cargo de Leandro Antonelli (Grupo LIFIA - UNLP). Las actividades propuestas estuvieron destinadas a ingenieros y licenciados en sistemas, estudiantes y docentes de estas especialidades, profesionales y empresarios del sector del software y servicios informáticos, así como también público interesado en la temática.

En total participaron más de 200 personas en las actividades programadas, lo que evidencia el interés que estos temas suscitan en los destinatarios.

The JCSA (Jornadas de Calidad de Software) began in 2017, under the original name of “Jornadas de Calidad de Software”, with the intention of communicating and divulging significant advances in the field of Software Engineering, particularly concepts, methods and tools which contribute to Software Quality. Providing a common forum where the software industry and universities can exchange experiences, needs, and foster joint actions in the field. In 2001, the conference broadened its scope, incorporating topics related to Agility, and including the reception of papers and industry experiences.

In the VI edition, the conference once again finds us in a virtual setting, from the Facultad Regional Resistencia, where the exposition of scientific papers, emerging research experiences, workshops, conferences and presentations took place, with the participation of members of the industry, University, and public corporations. In addition to the main activities of the conference, which where carried out virtually, a total of four on-site workshops were held, distributed in each of the participating venues.

The keynote conference entitled “Ciclo de vida del conocimiento en el desarrollo de software para la transformación digital”, was given by Leandro Antonelli (LIFIA Group - UNLP). The proposed activities were aimed at Systems engineers and graduates, students and teachers of these fields, as well as professionals and business leaders in the Software and IT services industry sector, and also to the general public interested in the subject.

In total, more than 200 people participated in the scheduled activities, demonstrating the interest that these topics generate among the participants.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Temas de Interés

- Calidad del Producto y Proceso de Software
- Calidad Interna y Externa del Software
- Testing de Software
- Verificación y Validación
- Modelos y Métricas de Calidad de Software
- Calidad de Datos
- User Experience - Diseño Centrado en el Usuario
- Herramientas, Aplicaciones Industriales y Estudios Empíricos
- Gestión de la Calidad de Software
- Enseñanza de Calidad de Software
- Métodos y Prácticas Ágiles
- Ingeniería del Software Continúa (Ejemplo: DevOps, Agile, etc)
- Ingeniería de Requisitos en Métodos Ágiles
- Peopleware y Management 3.0
- Estimación y Planificación Ágil
- Anomalías de Código – Refactoring

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Comité Organizador

FRRe – UTN

Dr. César J. Acuña

Dra. Verónica Bollati

Dra. Noelia Pinto

FCEQyN - UNaM

Dr. Horacio D. Kuna

Dr. Eduardo Zamudio

Esp. Alice Rambo

FaCENA – UNNE

Mgter. Gladys Noemí Dapozo

Dr. Emanuel Irrazábal

Lic. Ma. de los Ángeles Ferraro

Comité Científico

Mag. Liliana Cuenca Pletsch (CINApTIC - UTN - FRRe)

Dra. Verónica Bollati (CINApTIC - UTN - FRRe / CONICET)

Dr. César J. Acuña (CINApTIC - UTN - FRRe)

Dra. Noelia Pinto (CINApTIC - UTN - FRRe)

Esp. Gabriela Tomaselli (CINApTIC - UTN - FRRe)

Ing. Nicolas Tortosa (CINApTIC - UTN - FRRe)

Ing. Valeria Sandobal Verón (GIESIN- UTN - FRRe)

Ing. Germán Gaona (CINApTIC - UTN - FRRe)

Dr. Horacio Leone (INGAR - UTN - FRSF)

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Dr. Silvio Gonnet (INGAR - UTN - FRSF)

Dr. Gustavo Rossi (LIFIA - UNLP)

Dra. Alejandra Garrido (LIFIA - UNLP)

Dr. Andrés Rodriguez (LIFIA - UNLP)

Dr. Marcelo Estayno (UNSAM)

Dr. Luis Olsina (GIDIS - UNLPam)

Dra. Luciana Ballejos (INGAR - UTN - FRSF)

Mgter. Gladys Dapozo (UNNE)

Dr. Emanuel Irrazabal (UNNE)

Dr. Rubén Bernal (UNNE)

Dr. David la Red Martínez (UNNE)

Dra. Sonia Mariño (UNNE)

Mgter. Mónica Tugnarelli (UNER)

Dra. Gabriela Arévalo (UNQui)

Dra. Luciana Roldan (INGAR - UTN - FRSF)

Dra. Milagros Gutierrez (UTN - FRSF)

Dra. Mariel Alejandra Ale (UTN - FRSF)

Dra. Elsa Estevez (UNSur/CONICET)

Dra. Alicia Mon (UNLaM)

Dra. Nancy Ganz (IIDII-FCEQyN-UNaM)

Mgter.Ing.Alice Rambo (IIDII-FCEQyN-UNaM)

Ing. Selva Nieves Ivaniszyn (FCEQyN-UNaM)

Lic. Sergio Caballero (FCEQyN-UNaM)

Lic. Martín Rey (IIDII-FCEQyN-UNaM)

Dr.Eduardo Zamudio (IIDII-FCEQyN-UNaM))

Dr.Horacio Kuna (IIDII-FCEQyN-UNaM)

Dr. Diego Godoy (UGD)

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Ing. Edgardo Belloni (UGD)

Mgter. Cristina Greiner (UNNE)

Dra. María Fernanda Golobisky (UTN-FRStaFe)

Master Ariel Passini (UNLP)

Mgter. Pablo Thomas (UNLP)

Dr. Fernando Emmanuel Frati (UNDEC)

Dra. Marcela Genero Bocco (UCM)

Dr. Jorge Andrés Diaz Pace (SISTAN)

Dr. Nazareno Aguirre (UNRC)

Ing. Rubén Castaño (FCEQyN-UNaM)

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Índice

Keynotes

Talleres

Trabajos

Keynotes

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Ciclo de vida del conocimiento en el desarrollo de software para la transformación digital.

Abstract

El desarrollo de software es una de las actividades más complejas de la Ingeniería. Se trabaja con un producto intangible, por lo cual se necesitan muchos modelos para visualizarlo. Se están continuamente desarrollando nuevas herramientas para poder desarrollarlo. Y está conformado por un alto contenido intelectual. Es decir, el software incorpora el conocimiento del dominio para ayudar a los usuarios del dominio a realizar tareas. Algunos autores definen lo definen como: "conocimiento empaquetado". Este conocimiento se debe gestionar adecuadamente, dado que omisiones o errores, causan que el software no sea útil. Por otra parte, es necesario encontrar el balance justo, para que los modelos previos al código capturen el conocimiento, pero no requieran de un esfuerzo mayor que el que demanda la propia codificación. La transformación digital agrega un nivel más de complejidad. Ya que la misma significa que un negocio u organización actualiza su forma de trabajo, por lo cual, es necesario relevar los procesos, adecuarlos y describirlos correctamente. En esta charla se tratarán algunos aspectos claves como así también algunos artefactos para llevar adelante la gestión del conocimiento en el desarrollo de software para la transformación digital.

Disertante:

Leandro Antonelli (Grupo LIFIA - UNLP)

Short Bio:

Doctor en Ciencias Informáticas UNLP
Areas de trabajo:
(i) Ingenieria de requerimientos
(ii) Gestión de proyectos
Publicaciones: mas de 70 publicaciones en conferencias y revistas
Certificaciones de la industria
(i) Project Management Profesional (PMP) del Project Management Institute (PMI)
(ii) Certificacion Scrummaster de la Agile Alliance
Docente: UNLP, UAI, UNPA, UNRN
Dicto cursos en el Consejo Profesional de Ciencias Informaticas de la Prov de BsAs CPCIBA.
Miembro de la Direccion Informatica de Fiscalia de Estado de la Provincia de Buenos Aires.
Investigador del Centro de Investigacion LIFIA, Facultad de Informatica, UNLP.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

SancorSalud y su Comunidad de Testing.

Abstract

El área de tecnología en SancorSalud tiene una Comunidad de Testing que evoluciona constantemente no sólo en prácticas ágiles sino además en las prácticas propias de testing y te lo queremos contar.

Disertante

Gustavo Terrera (Grupo Sancor Seguros)

Bio

Líder de Comunidad de Testing en SancorSalud, apasionado del testing
https://www.linkedin.com/in/gustavoterrera/

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

¿Dora la exploradora? ¡No! Las métricas de DORA y su relevancia para la calidad en el desarrollo de software.

Abstract:

Desde hace 8 años y con el aporte de más de 33.000 profesionales, el Informe anual sobre el estado de DevOps de DevOps Research and Assessment (DORA) a cargo de la Doctora Nicole Forsgren, Google, es la investigación más rigurosa y de mayor duración sobre DevOps.

Descubra cómo las cuatro métricas de DORA, pueden mejorar la calidad de los procesos de desarrollo de software. Las dos primeras métricas hacen referencia a la velocidad del equipo de desarrollo: deployment frequency (frecuencia de despliegue) y lead time for changes (tiempo de ejecución para cambios). Las otras dos hacen referencia a la estabilidad del sistema: change failure rate (Tasa de despliegues fallidos en producción) y time to restore service (tiempo para restaurar el servicio).

Al disponer de las métricas de DORA y centrando los esfuerzos en mejorarlas, los equipos de desarrollo pueden reflexionar y actuar para mejorar su performance, entregar valor antes y de la mejor manera.

Disertante:

Diana Salinas (Globant)

Short Bio:

Ingeniera en Sistemas de Información, UTN FRRe.
Especialista en Ingeniería Gerencial

Certificaciones de la industria:
(i) Certified Scrum Master
(ii) Certified SAFe® 5 Agilist
(iii) Management 3.0

Delivery Manager at Globant
Scrum Master at LATAM Airlines, Globant contractor.

Talleres

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Taller 1

Desarrollo de microservicios con Phyton.

En el taller se darán los fundamentos prácticos de un microservicio en Phyton con un ejemplo real que se irá construyendo y mejorando. Se hará foco en la estructura y en las ventajas de este tipo de estrategia de diseño y programación.

Dictante / ShortBio: Ivan Sambrana

Ivan Sambrana es Licenciado en Sistemas de Información y docente-investigador de la Universidad Nacional del Nordeste. Es maestrando en la Maestría en Tecnologías de la Información (UNNE-UNaM) y lidera equipos de desarrollo en diferentes tecnologías: PHP, .Net y Phyton.

Público Objetivo / Conocimientos previos: alumnos avanzados o graduados de carreras informáticas.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Taller 2

Flaky Test: descubrirlo y corregirlo.

El taller aborda los principales problemas causados por las pruebas inestables, sus posibles causas y soluciones con ejemplos en código fuente java de algunos de los casos mas frecuentes.

Dictante / ShortBio: Pablo García Solanellas

Pablo García Solanellas es Analista Programador y auxiliar docente - investigador de la Universidad Nacional del Nordeste. Es un fanático emprendedor de los desarrollos utilizando buenas prácticas de Ingeniería de Software, especialmente las pruebas y el desarrollo continuo de software. Ha creado, gestiona y mantiene varios sistemas con tecnología Java y clientes de Argentina y el exterior.

Público Objetivo / Conocimientos previos: alumnos avanzados o graduados de carreras informáticas.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Taller 3

Escribiendo escenarios con Gherkin

Revisamos con ejemplos el desarrollo dirigido a comportamiento BDD (Behavior Driven Development), en lenguaje Gherkin, basado en 'Given-When-Then'.

Dictante: Amadeo García (Globant)

Dictante / ShortBio: Amadeo Garcia (Globant)

Público Objetivo / Conocimientos previos: alumnos avanzados o graduados de carreras informáticas.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Taller 4

Incorporando Calidad de código en el día a día

Este taller busca presentar herramientas para analizar y mejorar la calidad del código de un equipo de desarrollo en su día a día. Se presentarán las bases del tema y herramientas de soporte que se pueden emplear. Además, se van a proponer estrategias de integración con algunas herramientas de desarrollo habituales. Temas: Conceptos de calidad de código. Estándares de codificación. Herramientas de soporte. Uso en un flujo de CI/CD.

Dictante / ShortBio: Mgter. Sergio Caballero
FCEQyN - UNaM. Docente e investigador en el área de ingeniería de software y auditoría informática por la FCEQyN - UNaM. Se desarrolla en el ámbito profesional en puestos de gestión de procesos de desarrollo de software y despliegue de soluciones en el sector de la salud y afines.

Mgter. Martín Rey | FCEQyN - UNaM. Docente e investigador en las áreas de ingeniería de software y ciencia de datos por la FCEQyN - UNaM. Integrante del equipo de la Dirección de Tecnologías para la Gestión de la FCE - UNaM.

Dr. Eduardo Zamudio | FCEQyN - UNaM. Docente Investigador, miembro del Instituto de Investigación Desarrollo e Innovación en Informática (IIDII) de la Facultad de Ciencias Exactas, Químicas y Naturales de la Universidad Nacional de Misiones (FCEQyN, UNaM).Miembro del Comité académico del Doctorado en Informática (UNNE, UNaM, UTN-FRRe) Egresado del Doctorado en Ciencias de la Computación en la Universidad Nacional del Centro de la Provincia de Buenos Aires, financiado por una beca para Áreas de Vacancia Geográfica de CONICET. Actualmente participo en proyectos de investigación y dirijo tesis de grado y posgrado en el área de las Ciencias de la Computación, principalmente en análisis y predicción de datos, mediante técnicas de Procesamiento de Lenguaje Natural y Aprendizaje Automático.

Público Objetivo / Conocimientos previos: Alumnos avanzados o graduados de carreras informáticas.

Trabajos

JCSA 2022

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

El Manifiesto Ágil y Agile 2: Un estudio comparativo

Marcela Arias[1], Germán Gaona[2], Gabriela Tomaselli[2], Nicolás Tortosa[2]

[1] GIESIN, [2]CInApTIC – Facultad Regional Resistencia – Universidad Tecnológica Nacional
{arimarcela, germanxgaona, gabriela.tomaselli, nicotortosa}@gmail.com

Resumen. En 2001 un grupo de expertos se reunió con el objetivo de discutir de qué manera se podría mejorar la producción de software; allí nació el Manifiesto Ágil, que tiene en la actualidad una amplia adopción en la industria y ha dado origen a muchas prácticas y *frameworks*. Sin embargo, han transcurrido más de 20 años desde entonces, y se hizo necesaria una revisión de los valores y principios establecidos en el Manifiesto; surge así una nueva propuesta denominada Agile 2. En este trabajo se realiza un análisis comparativo de ambos enfoques a fin de determinar qué valores y principios siguen vigentes, cuáles se modificaron y si algunos fueron descartados.

Palabras clave: Agilidad; Manifiesto ágil; Agile 2; post-agilismo; ingeniería de software.

1 Introducción

La Agilidad es un concepto amplio que abarca un conjunto diverso de enfoques y creencias, surgido como reacción a las metodologías tradicionales de desarrollo de software, cuyo representante más emblemático es el modelo de ciclo de vida en cascada. Además de una respuesta efectiva al cambio, la Agilidad, promueve estructuras de equipo que facilitan la comunicación, pone énfasis en la entrega rápida de software funcional, e incorpora al cliente en el equipo de desarrollo [1].

La Agilidad no es nueva [2], si bien su origen se formaliza en el año 2001 a partir de la declaración del Manifiesto Ágil (MA) [3], en este se consensuaron valores y principios comunes a filosofías, métodos y prácticas preexistentes, utilizadas en la industria desde la década de 1990, tales como: Scrum [4], métodos Crystal [5], Dynamic Systems Development Method - DSDM [6], Extreme Programming - XP [7], Feature Driven Development - FDD [8] y Adaptive Software Development - ASD [9].

El MA permitió llegar a un entendimiento acerca del significado de la Agilidad, aumentando la tasa de éxitos en comparación con los enfoques tradicionales [10], se crearon nuevas propuestas y se actualizaron algunas existentes; lo que pudo parecer positivo, resultó en una nueva fragmentación con algunas soluciones altamente prescriptivas (enfocadas en escalar la Agilidad). Esto, sumado a la visión dogmática de

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

ciertas comunidades hizo perder el foco en la esencia de los valores y principios originales, como señala Hervouet [11].

Con críticas al rumbo que había tomado la Agilidad, surge el post-agilismo, la idea de reflexionar sobre el significado original del MA, algunas de ellas formuladas por sus propios creadores, como el caso de Thomas [12], Jeffries [13], o Cockburn [14].

En este contexto aparece Agile 2 (A2) [15], con otro conjunto de valores y principios, aunque con cierta raigambre en aquellos originales, que brindan algunas ideas para solucionar los inconvenientes que se presentan en el MA (o más bien con su aplicación); en el presente trabajo se efectúa el análisis de esta nueva propuesta y la comparativa con la versión original del manifiesto.

El resto del trabajo se estructura de la siguiente manera: en la Sección 2 se describen los antecedentes, en la Sección 3 se presenta la metodología seguida, en la Sección 4 se presentan los resultados obtenidos, y en la última Sección se realiza la discusión y se mencionan las conclusiones y trabajos futuros relacionados.

2 Antecedentes

Existen numerosas opiniones en la industria a favor de un cambio en la Agilidad; para los más extremistas “la Agilidad ha muerto” [12], para otros, esta debe evolucionar, o simplemente ya ha evolucionado y resulta necesario poner de manifiesto dichos cambios, que en la mayoría de los casos proponen una vuelta a sus inicios [16]. Se exponen a continuación algunos de estos enfoques.

2.1 Agile is Dead

En los últimos años, algunos profesionales involucrados en el movimiento ágil contradicen la simplicidad y adaptabilidad originalmente postuladas, proponiendo el uso de prácticas altamente prescriptivas. Como reacción a esto, en marzo de 2014, Dave Thomas (uno de los 17 firmantes originales del MA) proclamó: “Agile is Dead (Long Live Agility)” [12].

Ágil (o *Agile* en el sentido propuesto por el autor) pasó de ser un conjunto de valores y principios compartidos a una palabra usada en exceso que se ha aplicado incorrectamente en distintos ámbitos, es decir, los programadores, equipos o herramientas no son ágiles per se, sino que usan la Agilidad para la consecución de sus objetivos.

2.2 Heart of Agile

En su artículo de 2016 [14] Alistair Cockburn propone volver a las raíces de la agilidad, enfocándose en cuatro palabras claves: Colaborar, Entregar, Reflexionar y Mejorar, que en su conjunto las denomina “The Heart of Agile” o el Corazón de la Agilidad. El autor la considera una herramienta de enfoque “simple, suficiente y expandible” acerca de la esencia de la agilidad, con el propósito de retornar a los valores y principios del MA.

Cockburn se basa en un concepto proveniente de las artes marciales japonesas conocido como “Shu-Ha-Ri” (obedecer, liberarse y trascender), que describe brevemente la progresión del aprendizaje de nuevas habilidades. Una vez recorrido este

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

camino, menciona que es conveniente retornar a la esencia, corazón o “Kokoro” de esas prácticas, de allí el nombre “Agile no kokoro” o “Corazón de la Agilidad”.

2.3 Modern Agile

Otra de las iniciativas que aportan a la evolución de la agilidad es la de Joshua Kerievsky [17], quien en 2016 propone un enfoque “más ligero” que apunta a obtener resultados de modo más simple, seguro y rápido, conocido con el nombre de “Modern Agile” o Agilidad Moderna.

En oposición a la agilidad tradicional, oprimida en una maraña de herramientas, *frameworks* y certificaciones cuestionables que producen más burocracia que resultados, Modern Agile no tiene roles, responsabilidades ni prácticas consagradas; por el contrario, se reduce a tan solo cuatro principios rectores de enunciación simple: “Haz que las personas sean geniales”, “Haz de la seguridad un prerrequisito”, “Experimenta y aprende rápido”, y “Entrega valor continuamente”.

2.4 Rethinking Agile

En el año 2018 Klaus Leopold plantea los motivos por los cuales los equipos ágiles no se condicen con la agilidad empresarial. En [18] narra su experiencia con una empresa cuyo objetivo era prepararse para incorporar la Agilidad en su operatoria diaria.

Según su perspectiva, una verdadera agilidad empresarial se integra de estrategia, operaciones, desarrollo y entrega que trabajan en estrecha colaboración, sobre todo con el mismo fin. Tiene como objetivo transformar positivamente las métricas empresariales.

Leopold aconseja que el primer equipo ágil tiene que ser la alta gerencia, quien debe reflexionar sobre la agilidad empresarial, su significado, los problemas a tratar y su papel en el proceso.

3 Metodología

La necesidad del análisis comparativo llevado a cabo en este trabajo radica en la reciente aparición de la iniciativa A2 y en la ausencia de estudios académicos que relacionen las propuestas realizadas por esta con las planteadas en el MA.

El primer interrogante a resolver es determinar qué valores y principios de A2 se asocian con aquellos presentes en el MA y, en segundo término, de qué naturaleza son dichas asociaciones.

Para efectuar el análisis, se adoptaron como unidad comparativa, los valores y principios de cada enfoque, aprovechando la similitud en el formato de ambas propuestas. Esto significa que, para cada valor expresado en A2 se buscaron las relaciones con los valores y principios especificados en el MA; de igual manera se procederá con cada principio de A2, considerando el agrupamiento por categorías efectuado en dicha propuesta. La naturaleza de cada relación expresa cierta cercanía de los conceptos tratados en cada par de elementos comparados, ya sea por concordancia o variación.

Dado que en el MA existen 4 valores y 12 principios, y A2 cuenta con 6 valores y 43 principios, para facilitar la comparativa y poder referenciar cada elemento de manera

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

unívoca, se decidió etiquetarlos; un compendio completo de ellos se encuentra disponible en línea[1]. Cabe aclarar que en la mayoría de los casos se tomó la traducción presente en los sitios de ambas iniciativas, en tanto que los principios de A2 debieron ser traducidos por no existir una versión oficial en español.

En virtud de que el análisis fue meramente conceptual, como resultado de este se buscó representar y sintetizar las conexiones halladas mediante gráficos que las muestran en forma de enlaces entre nodos conteniendo los valores y principios involucrados; asimismo, esto permitió obtener una retroalimentación que sirvió para reevaluar el análisis efectuado.

4 Resultados

A continuación se presentan los resultados obtenidos al aplicar el análisis comparativo descripto anteriormente, discriminados por valores y luego por principios de A2.

4.1 Valores de Agile 2 y su relación con valores y principios del MA

En A2, se presentan seis valores que están enunciados en el sitio web oficial [15] (no así en el libro que acompaña a la iniciativa [19]) en formato de conceptos contrapuestos, declarando: “Valoramos todas estas cosas, y nos esforzamos por equilibrarlas o combinarlas en cada situación.”.

Este enfoque se asemeja al MA, que cuenta con cuatro valores vinculados originalmente con el desarrollo de software, que promueven la consideración de uno por sobre otro. Aquí cabe la reflexión acerca de la utilización del término over en la enunciación de valores del MA, que dio origen a malos entendidos e ideas erróneas, tal lo analizado por Ozkan en [20]. Sin embargo, Highsmith (firmante del MA) expresa que es necesario un equilibrio entre los ítems de la derecha y la izquierda (personas y procesos, software funcional y documentación, colaboración y contratos, respuesta al cambio y planificación), y a fin de permitir a cada organización, equipo o individuo hallar su propio punto medio, se delimitaron los extremos: “Si empezamos tratando de encontrar el punto medio, nunca lo haremos” [21]. La búsqueda del equilibrio adecuado es coincidente con uno de los *insights* de A2: “Los extremos no suelen funcionar en la mayoría de las situaciones”.

Por su parte, en A2 los valores vienen acompañados de una explicación que ayuda a comprender el pensamiento de sus autores, lo que surge como una crítica directa al MA, que dejaba a libre interpretación del lector el significado de estos, al no incorporar ningún tipo de descripción asociada.

Reflexión y prescripción (A2V1). Este valor apunta al curso de acción a tomar para lograr un objetivo, es decir, adaptarse de acuerdo al contexto (conociéndolo primero) o a utilizar un framework, método o práctica de manera estricta, sin considerar las circunstancias particulares. En el mismo sentido, los creadores del MA llaman a usar el criterio propio al momento de aplicar las ideas enunciadas por ellos; sin embargo, este

[1] https://www.frre.utn.edu.ar/cinaptic/clean/files/get/item/12652.pdf

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

nuevo valor surge como respuesta al pensamiento binario [20] o al dogmatismo en la agilidad en general [22] y de algunos *frameworks* en particular.

En definitiva, cada método a utilizar dependerá de la situación; no hay ninguna metodología o *framework* que deba utilizarse sin adaptaciones. Si bien este valor nace como reacción a aplicaciones radicales de la agilidad, es posible resaltar que uno de los principios del MA establece que "A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia" (MAP12).

Resultados y salidas (A2V2). Aquí se establece la diferencia entre los resultados finales (*outcomes*), alineados con los objetivos empresariales, de las salidas (*outputs*) que se obtienen de un proceso. Los primeros aportan valor al negocio y hacen al fin último de una organización, las segundas son el fruto directo de la ejecución de una actividad o un proceso, del que se esperan artefactos o entregables concretos y que no necesariamente contribuyen de manera directa al objetivo final. Tradicionalmente el éxito de los proyectos se juzga en base a sus salidas siguiendo el modelo "Entradas-Proceso-Salidas" (IPO, por sus siglas en inglés) sin embargo en [23], proponen el modelo "Entradas Transformadas en Resultados" (ITO, por sus siglas en inglés), que se vale de la utilización de las salidas para lograr los resultados o efectos esperados.

El MA establece que prefiere "Software funcionando sobre documentación extensiva" (MAV2), A2 agrega que el software funcionando (salida), no es prueba de valor, que solo los resultados esperados por el negocio lo son. En el mismo sentido, el MA enuncia "Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor" (MAP1), y "El software funcionando es la medida principal de progreso" (MAP7), poniendo el énfasis en la satisfacción del cliente, centrándose en la entrega de "software con valor"; de este modo, se prescriben los tipos de objetivos que deben tener los equipos y lo que constituye el éxito del proyecto [24]. Por último, se considera que puede existir una relación débil con: "Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible" (MAP3), ya que este principio se refiere al tipo de salida (que es el software funcional) pero haciendo hincapié en la frecuencia con que se entrega.

Individuos y equipos (A2V3). La comunidad ágil ha puesto mucho énfasis en el equipo, en cuanto a las cualidades con las que debería contar y las prácticas asociadas a estos, dejando de lado a las personas que lo componen, a pesar de que el MA enuncia que prefiere a "Individuos e interacciones sobre procesos y herramientas" (MAV1). En A2 se enfatiza la idea de que ambos, tanto el individuo como el equipo, son importantes y se debe encontrar un balance entre ellos, vinculado con su bienestar e intereses.

En A2 se critica que algunas prácticas ágiles (equipos autoorganizados, reuniones diarias, comunicación cara a cara, oficinas abiertas, etc.) fueron diseñadas considerando un equipo homogéneo de ciertas características, ignorando los rasgos personales (en A2 se refieren a la extroversión, pero se podría extender a otras dimensiones del "Big Five") [25], la experiencia de cada integrante de forma individual (o de forma colectiva) o la diversidad cultural [26], [27].

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Aquí también A2 reacciona ante las consecuencias de malas interpretaciones y aplicaciones de los principios del MA, que expresan: “Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo” (MAP5), poniendo en un plano de igualdad en cuanto a importancia al individuo y al equipo.

Conocimiento del negocio y conocimiento técnico (A2V4). En las empresas modernas, principalmente en aquellas de base tecnológica, debería existir un trabajo mancomunado entre las personas encargadas del negocio y aquellas encargadas del área técnica, se debería contar con un conocimiento holístico de la empresa, de sus procesos y la plataforma tecnológica que les da soporte. Esto no significa que todos deberían ser expertos en todo, sino que deberían interesarse en lo que los demás están haciendo. La importancia que este valor otorga al conocimiento técnico permite relacionarlo débilmente con el principio “La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad” (MAP9).

Este conocimiento holístico implica que los altos mandos deben saber no sólo *qué* se está construyendo, sino *cómo* se está construyendo el producto o servicio, es decir la tecnología de desarrollo es parte de la estrategia del negocio, por lo que el equipo de desarrollo no debe ser tomado como un simple implementador subordinado, sin brindarle el tiempo ni el espacio para explotar el potencial de innovación existente. Esto se vincula fuertemente con “Los empresarios y los desarrolladores deben trabajar juntos diariamente durante todo el proyecto” (MAP4), y también existe una vinculación leve con “Colaboración con el cliente sobre negociación contractual” (MAV3), ya que según [28] las experiencias y habilidades de cada parte permitirían cambiar de rumbo si fuese necesario.

Empoderamiento individual y buen liderazgo (A2V5). Según los autores de A2, el liderazgo es la cuestión más importante de todas y que en general fue soslayada por la comunidad ágil, al llevar al extremo la autoorganización de los equipos planteada en “Las mejores arquitecturas, requisitos y diseños surgen de equipos autoorganizados” (MAP11). Se pasó de poseer un liderazgo centralizado (quizás tóxico), a no tener ningún tipo de autoridad formal; en estas situaciones pueden surgir liderazgos naturales que incluso serían más contraproducentes para el trabajo cotidiano o la consecución de los resultados esperados.

Un buen líder debería poseer autoridad y responsabilidad para tomar decisiones en situaciones cruciales, estar abierto a ideas, empoderar y guiar al equipo y a sus miembros, ayudando a lograr cierta autonomía y crecimiento personal. De acuerdo con [29], al intentar que un equipo sea autoorganizado se presentan barreras a nivel organizacional y a nivel de equipo. En [30] además manifiestan que existe una falta de claridad en el rol del líder de proyecto, afectando al equipo que tiene que lidiar con estas cuestiones. Por su parte, en [31] distinguen entre equipos autoorganizados y autogestionados, el MA menciona a los primeros y no necesariamente requiere de los segundos, estos últimos además, podrían decidir qué producto construir, a qué mercados ingresar, etc., cuestiones que exceden a las responsabilidades de un equipo autoorganizado. Estudios empíricos sobre equipos de desarrollo ágil [24], han hallado

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

poca evidencia de equipos autogestionados, y resulta necesaria más investigación acerca de si la autogestión conduce a un mayor rendimiento del equipo.

Adaptabilidad y planificación (A2V6). Uno de los valores más fuertes dentro del MA es la “Respuesta ante el cambio sobre seguir un plan” (MAV4), dado que los cambios son inevitables, se opta por aceptarlos y gestionarlos adecuadamente dentro de cualquier enfoque ágil. El MA prefiere el cambio por sobre el plan, es decir, planificar solo lo necesario, de hecho la mayoría de las prácticas ágiles poseen algún tipo de técnica de planificación acotada en el tiempo [32]. Lo que se desalienta son los planes rígidos, detallados y a largo plazo que tienen grandes probabilidades de no cumplirse. En este sentido A2 promueve planes lo suficientemente claros para obtener una visión del proyecto, pero con la posibilidad de ser modificados, dirigidos principalmente por resultados y no por tareas.

Como consecuencia de interpretaciones erróneas de “Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente” (MAP2) se desestimó la importancia de la planificación; aunque el MA no decía "no planificar", su afirmación de que "Responder al cambio" resulta más valioso que seguir un plan derivó incluso en no planificar más allá de la iteración actual. En contraposición surge: “Toda iniciativa exitosa requiere tanto una visión o un objetivo como un plan flexible, dirigible y orientado a los resultados” (A2P1.1); aquí los autores de A2 resaltan la necesidad de establecer una visión hacia la que trabajar, y comprender la estrategia global para poder alinearse con ella, remarcando que el modelo de liderazgo conocido como "Mission command" adopta este principio.

La **Figura 1** resume las relaciones previamente descriptas que surgieron a partir del análisis comparativo de los valores en A2 y los valores y principios del MA.

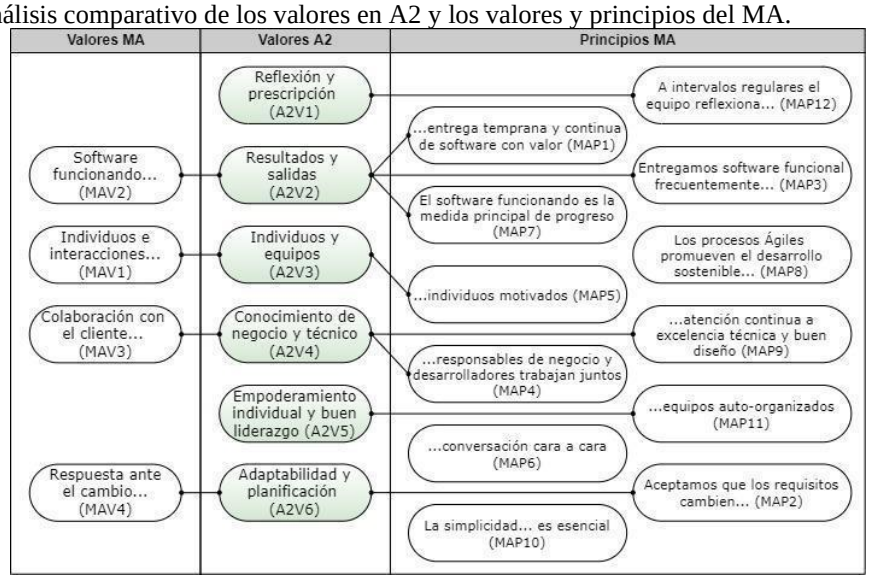

Figura 1. Relaciones entre valores de A2 y valores/principios de MA (Elaboración propia)

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

4.2 Principios de Agile 2 y su relación con valores y principios del MA

A diferencia del MA, A2 posee una extensa descripción de sus principios incluyendo los problemas detectados y los *insights* correspondientes a los que se arribaron (por consenso total o parcial) luego de efectuar un análisis retrospectivo sobre el estado de la Agilidad. Los 43 principios postulados en A2 se encuentran clasificados en 10 categorías; aquí se expone un análisis sintético basado en estas, un análisis pormenorizado de los principios será presentado en trabajos futuros.

Las categorías: (1) Planificación, transición y transformación; (2) Producto, cartera de clientes y *stakeholders*; (4) *Frameworks* y metodologías; (5) Dimensión técnica y fluidez técnica; (6) Individualidad vs. equipo; (7) Equipo vs. organización; (8) Mejora continua; y (9) Enfoque; son las que mayor cantidad de relaciones presentan con el MA, siempre desde una perspectiva diferente, y con un mayor grado de detalle; varios de estos aspectos fueron mencionados en el análisis de los valores. Cabe aclarar que dentro de estas categorías existen algunos principios que no pudieron ser vinculados con el manifiesto.

A pesar del consenso existente en cuanto al valor estratégico de los datos, en el MA no hay referencia alguna a ellos, por ende, la categoría (3) Datos incorporada en A2 no presenta antecedentes en el MA.

La categoría a la que se otorga más importancia en A2, es la (10) Liderazgo, sosteniendo que la comunidad ágil ha simplificado demasiado este concepto en favor de la conformación de equipos autoorganizados. Se incluyeron aquí numerosos principios con diferentes aristas que no fueron contempladas en el MA.

5 Discusión, conclusiones y trabajos futuros

De acuerdo con el análisis comparativo efectuado, se aprecia que A2 conforma una perspectiva diferente acerca de la Agilidad, no pretendiendo reemplazar al MA, sino buscando aportar mayor detalle a sus valores y principios, e incorporando nuevos, con lo que se corrobora la intención planteada por sus autores, la de “pivotar” sobre el manifiesto original. Esto es, no lo contradice, más bien ofrece algunas alternativas a ideas ya existentes: valora a los equipos y su autonomía, pero también resalta las individualidades; rescata la importancia de la plataforma tecnológica, particularmente de cómo se construyen y entregan los productos o servicios y su aporte directo a los resultados esperados a nivel empresarial; enfatiza que la planificación se torna imprescindible para alinear los esfuerzos con la visión del negocio, con un alto nivel de adaptabilidad ante el cambio.

Entre los nuevos valores propuestos, se refuerza el balance entre la reflexión y la prescripción, es decir la utilización estricta de ciertos *frameworks* o prácticas, cuestión no tratada en el manifiesto original, y que dio lugar a malas interpretaciones en la comunidad ágil.

El aporte central de A2 puede resumirse en uno de sus valores “Empoderamiento individual y buen liderazgo”, jerarquizando por un lado la consideración de las individualidades, que aun estando presentes entre los valores del MA, en la práctica

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

fueron en muchos casos dejadas de lado en favor de los equipos, y por otro lado enfatizando al liderazgo como imprescindible para lograr los objetivos deseados.

Una de las incorporaciones destacables entre los principios de A2 (que no fueron tratados en profundidad en este trabajo) es la utilización de los datos como un activo estratégico, cuya importancia para la toma de decisiones en las organizaciones modernas, creció exponencialmente en las últimas décadas.

Una característica general presente en A2 desde su concepción es la autodescripción, esto es: a cada valor o principio enunciado lo acompaña una explicación detallada del significado otorgado por sus autores, buscando de esta manera disminuir la posibilidad de malas interpretaciones de terceros; esto contrasta con el MA, que se limita a listar cada una de sus ideas, sin existir siquiera versiones aclaratorias posteriores que puedan considerarse "oficiales". Tales explicaciones dieron como resultado una propuesta extensa, y un número considerablemente grande de principios, que pueden tornarse inmanejables, y en algunos casos repetitivos.

El trabajo desarrollado es de índole conceptual, con lo que sus aportes se circunscriben mayormente al análisis comparativo y el aporte de referencias académicas a algunas de las afirmaciones realizadas en A2, pues al igual que el manifiesto original, se basa fuertemente en la experiencia práctica de sus creadores, incluyendo escasas o nulas evidencias científicas.

Respecto a los posibles trabajos futuros, se pretende realizar un análisis pormenorizado de los principios y continuar en la búsqueda de evidencia científica que les dé soporte. Por otra parte, sería de interés para la industria, la revisión bajo esta nueva óptica de las prácticas ágiles existentes de uso más extendido.

Finalmente, A2 es una iniciativa que surge en el post-agilismo y al igual que otras de este mismo período queda por ver si será ampliamente aceptada por la comunidad ágil al igual que el manifiesto original.

Referencias

1. Pressman, R.S. (2010). *Ingeniería del Software. Un enfoque práctico*. (7ª ed.). Mc Graw Hill.
2. Rigby, D. K., Sutherland, J., & Takeuchi, H. (2016). The secret history of agile innovation. *Harvard Business Review*, *4*.
3. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). *Manifesto for agile software development*. https://agilemanifesto.org/
4. Schwaber, K. (1997). SCRUM development process. En *Business object design and implementation* (pp. 117-134). Springer, London.
5. Cockburn, A. (2004). *Crystal clear: A human-powered methodology for small teams: A human-powered methodology for small teams.* Pearson Education.
6. Stapleton, J. (1999). DSDM: Dynamic systems development method. En *Proceedings Technology of Object-Oriented Languages and Systems. TOOLS 29 (Cat. No. PR00275)* (pp. 406-406). IEEE.
7. Wells, D. (2013). *Extreme Programming: A gentle introduction.* http://www.extremeprogramming.org/
8. Coad, P., Lefebvre, E. & De Luca, J. (1999). *Java modelling In Color With UML: Enterprise Components and Process*. Chapter 6. Prentice Hall International.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

9. Highsmith, J.A. (2000). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House Publishing.
10. Domínguez, J. (2009). The curious case of the chaos report 2009. *Project Smart*, 1-16.
11. Hervouet, P. (2018). Heart of Agile. *Lean and Agile ME Summit 2018.* https://www.youtube.com/watch?v=MarsFSnCC-k
12. Thomas, D. (2014). *Agile is Dead (Long Live Agility).* https://pragdave.me/blog/2014/03/04/time-to-kill-agile.html
13. Jeffries, R. (2015). *Management in Agile*. https://ronjeffries.com/articles/015-12/management/
14. Cockburn, A. (2016). The heart of agile. *CrossTalk, November/December 2016*, 4-6.
15. Agile 2 THE NEXT ITERATION OF AGILE. (2020). https://agile2.net/
16. Bollati, V., & Garzás, J. (2018). Nuevas tendencias en el futuro de la agilidad. *Novatica, 240*, 1-22.
17. Kerievsky, J. (2016). An Introduction to Modern Agile. *InfoQ.* https://www.infoq.com/articles/modern-agile-intro
18. Leopold, K. (2018). *Rethinking Agile: Why Agile Teams Have Nothing To Do With Business Agility.* Leanability Press.
19. Berg, C., Cagle, K., Cooney, L., Fewell, P., Lander, A., Nagappan, R., & Robinson, M. (2021). *Agile 2: The Next Iteration of Agile.* John Wiley & Sons.
20. Ozkan, N., & Gök, M. S. (2021). Towards the End of Agile: Owing to Common Misconceptions in the Minds of Agile Creators. En *ICSOFT* (pp. 224-232).
21. Highsmith, J. (2002). *Agile software development ecosystems.* Addison-Wesley Professional.
22. Kruchten, P. (2019). The end of agile as we know it. En *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)* (pp. 104-104). IEEE.
23. Zwikael, O., & Smyrk, J. (2010). The Impact of a Project Benefit Methodology on the Project Management Discipline. Project Management Institute.
24. Dingsøyr, T., Fægri, T.E., Dybå, T., Haugset, B., & Lindsjørn, Y. (2016). Team Performance in Software Development: Research Results versus Agile Principles. *IEEE Software, 33*, 106-110.
25. Aghina, W., Handscomb, C., Ludolph, J., West, D., & Yip, A. (2019). How to select and develop individuals for successful agile teams: A practical guide. *Survey. McKinsey & Company.*
26. Mendes, E., Viana, D., Vishnubhotla, S. D., & Lundberg, L. (2018). Realising individual and team capability in agile software development: A qualitative investigation. En *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 183-190). IEEE.
27. Deak, A., Stålhane, T., & Sindre, G. (2016). Challenges and strategies for motivating software testing personnel. *Information and software Technology, 73*, 1-15.
28. Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer, 34(9)*, 120–127.
29. Moe, N. B., Dingsøyr, T., & Dybå, T. (2009). Overcoming barriers to self-management in software teams. *IEEE software, 26(6)*, 20-26.
30. Hoda, R., & Murugesan, L. K. (2016). Multi-level agile project management challenges: A self-organizing team perspective. *Journal of Systems and Software, 117*, 245-257.
31. Medinilla, Á. (2012). *Agile management: Leadership in an agile environment.* Springer Science & Business Media.
32. Cockburn, A. (2006). *Agile software development: the cooperative game.* Pearson Education.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Análisis comparativo de librerías de Python para visualización de información: un estudio de caso

Maria L. Godoy[1], Martín Cardozo[1], Sebastian Bazterrica[1], Andrea Lezcano Airaldi[1]

[1] Universidad Nacional del Nordeste, Corrientes, Argentina
{marialgodoy15, martincardozo, sebastianbazterrica}@gmail.com.ar
alezcano@exa.unne.edu.ar

Resumen. El control de plagas en áreas como agricultura y ganadería es una problemática de interés dado que influye directamente en la productividad. El seguimiento y control adecuados mediante visualizaciones de información puede facilitar el proceso de toma de decisiones por parte de los interesados. En este artículo se realiza una comparación empírica de las principales librerías de visualización de datos para Python en el contexto de control de parásitos en ganado bovino. Para cada librería se analizan aspectos como complejidad de código, tiempo de ejecución y usabilidad. Los resultados indican que la selección final de una librería depende de las características de cada proyecto y reafirman la necesidad de crear equipos interdisciplinarios con personal capacitado en programación.

Palabras Clave: visualización de información, python, estudio de caso

1 Introducción

El control de plagas en el ganado y la agricultura es una problemática de interés dado que no solo afecta la salud de las especies sino también la productividad [1]. El seguimiento y reporte adecuados de los tratamientos sanitarios puede facilitar la toma de decisiones por parte de los responsables involucrados. En este sentido, la visualización de información proporciona un recurso útil para identificar, evaluar y gestionar los peligros para la salud de la manera temprana y eficiente posible [2]. Por ello es común encontrar visualizaciones estándares en los reportes académicos de control de plagas, muchas veces construidas con herramientas generalistas y haciendo uso de plantillas para los cálculos de mayor complejidad.

En el área de la Ciencia de Datos, existen varios lenguajes de programación utilizados para la visualización de datos, tales como R, Python, Julia o Matlab, entre otros. Cada uno ofrece librerías y paquetes para visualización con diferentes características para construir gráficos personalizados ajustados a las necesidades de los interesados. En particular, Python se caracteriza por una sintaxis sencilla con una curva de aprendizaje práctica para programadores novatos [3]

Este artículo presenta un estudio de caso en el contexto del monitoreo de tratamiento contra parásitos en ganado bovino. Se realizó una comparación empírica de distintas librerías de visualización de Python con el objetivo de identificar ventajas y desventajas

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

de las mismas, proporcionar una visión general de las más utilizadas y discutir cuál de ellas lograba presentar los datos de la forma más simple y efectiva posible. En particular, se enfoca en cuatro librerías: Matplotlib [4] y Seaborn [5] para visualizaciones estáticas y Plotly [6] y Bokeh [7] para visualizaciones interactivas.

Además de esta sección introductoria, el trabajo se encuentra organizado de la siguiente manera. En la Sección 2 se describen los antecedentes y los trabajos relacionados. En la Sección 3 se detalla la metodología de investigación propuesta, en la Sección 4 se reportan los principales hallazgos y en la Sección 5 se presenta la conclusión.

2 Antecedentes y Trabajos Relacionados

2.1 Visualización de datos en Python

El lenguaje Python comprende una gran variedad de librerías para visualización de datos. En 2017 Vanderplas [8], describe las bibliotecas de visualización en un gráfico de red con los grupos y las relaciones entre las herramientas, adaptado más adelante por Nicolas P. Rougier [9], como se muestra en la Fig. 1. En 2019, Anaconda publicó Pyviz [10], una plataforma abierta para informar sobre las herramientas actuales de visualización en Python.

Entre las distintas librerías, hay alternativas simples, como Matplotlib, y alternativas más complejas, como Altair, construidas sobre JavaScript.

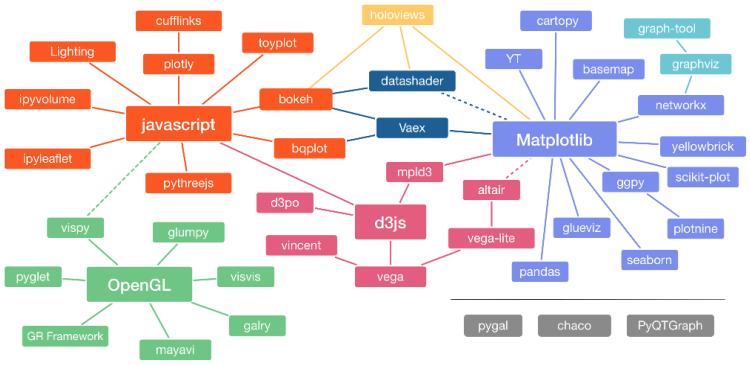

Fig. 1. Librerías para visualización de datos en Python. Fuente: [9]

Como se puede observar, la mayoría de las herramientas se centran alrededor de tres grandes grupos: visualizaciones estáticas basadas en Matplotlib, visualizaciones interactivas basadas en Javascript y HTML o visualizaciones y animaciones de mayor complejidad basadas en OpenGL.

Por otro lado, [11] distingue además dos grupos: visualizaciones científicas y visualización de información, siendo este último el foco de este estudio.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

2.2 Otras comparaciones

Otros autores han realizado análisis similares al que se presenta en este trabajo. En esta sección, se discuten algunos de los más relevantes en relación con los objetivos propuestos en este artículo.

En [12], los autores comparan librerías utilizadas en el proceso de análisis de datos, distinguiendo distintas categorías de acuerdo con el tipo de tarea realizada: librerías nativas para tareas generales, preparación de datos, visualización de datos, machine learning, deep learning y big data. En el artículo se analizan 20 librerías de acuerdo con factores como contribuciones, mantenimiento y tamaño de la comunidad. Si bien los resultados no son determinantes, para visualización de datos los autores recomiendan tanto Matplotlib, como Seaborn o Plotly.

Por otro lado, en [13], Acar et. al se enfocan en librerías de criptografía y analizan cómo el diseño y usabilidad de las librerías influyen en su utilización, con el fin de proponer mejoras para el desarrollo de futuras librerías. En este caso, los autores llevaron a cabo un experimento donde los usuarios debieron realizar una serie de tareas y evaluaron tanto la experiencia de los programadores como el código resultante.

Por último, Herda y Mcnabb [14] estudian el rendimiento y la usabilidad de librerías de visualización para grandes conjuntos de datos en el contexto de ciudades inteligentes, con foco en aplicaciones geoespaciales. Los autores analizan la capacidad de diferentes librerías para la creación de mapas estáticos e interactivos para que los gobiernos locales puedan presentar información de manera abierta y transparente.

Este trabajo es similar a los mencionados anteriormente puesto que busca analizar y caracterizar el rendimiento de librerías de visualización de datos, haciendo foco en visualizaciones de información, como así también evaluar la experiencia del equipo de programadores al utilizarlas.

3 Metodología

El objetivo de este artículo es realizar una comparación empírica de distintas librerías de visualización de datos en Python, tanto estáticas como interactivas para obtener una visión general de sus propiedades. Partiendo de datos del tratamiento contra parásitos en ganado bovino, se conformó un grupo de 8 programadores y se buscó replicar una visualización “modelo”. La misma consistió en un gráfico de líneas que presenta el nivel de concentración en sangre del ganado de tres drogas diferentes en un periodo de tiempo de 30 días.

Para llevar a cabo el estudio se siguieron los pasos propuestos por Yin [15] y Wohlin [16] para el desarrollo de estudios de caso y enumerados a continuación: diseño del estudio, recolección de datos, análisis de los datos recolectados y reportes de resultados.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

3.1 Recolección y Análisis de Datos

Selección de Librerías de Visualización.
En primer lugar, se seleccionaron cuatro librerías de Python para comparar empíricamente en base a su popularidad y su idoneidad para las tareas de interés: Matplotlib [4], Seaborn [5], Bokeh [7] y Plotly [6], descritas a continuación.

— **Matplotlib:** Es una biblioteca completa para crear visualizaciones estáticas. Puede hacer uso de librerías como Numpy o Pandas para lo referido al tratamiento de datos y proporciona un buen rendimiento incluso para grandes conjuntos de datos. Una desventaja de esta librería es que no tiene la capacidad de crear gráficos que sean interactivos para el usuario y su sintaxis es más bien compleja; sin embargo, ofrece un alto grado de personalización.
— **Seaborn:** Está basada en Matplotlib y se integra con las estructuras de datos de Pandas. Seaborn es capaz de "entender" directamente una estructura de datos, representando fácilmente la relación que guardan los mismos para detectar tendencias y patrones en pocas líneas de código.
— **Plotly:** Es una librería de código abierto construida sobre JavaScript. Permite a los usuarios crear visualizaciones interactivas basadas en la web que se pueden mostrar en cuadernos Jupyter, guardar en archivos HTML independientes o servir como parte de aplicaciones web puras creadas con Python, como tableros de control. Se caracteriza principalmente por su sintaxis simple, permitiendo crear visualizaciones completas en pocas líneas de código.
— **Bokeh:** Es una librería de visualizaciones interactivas que permite crear una amplia variedad de gráficos e integrarlos de manera sencilla en una aplicación web. Su principal ventaja es la flexibilidad para personalizar los gráficos.

Análisis Comparativo.
Las librerías seleccionadas fueron analizadas en base a los siguientes criterios:

- *Visualizaciones disponibles.* Se navegó por la documentación de cada librería con el fin de identificar el rango de gráficos permitidos, considerando aquellos más utilizados para visualización de información, desde los más comunes como gráficos de líneas o de barras, hasta técnicas más complejas como gráficos de violín o diagramas de Sankey.
- *Líneas de Código*. Como medida de complejidad, se consideró el número de líneas de código necesarias para crear el gráfico línea, excluyendo comentarios y líneas en blanco.
- *Tiempo de ejecución de CPU:* Se midió el tiempo de ejecución de la CPU requerido para completar la función crearVisualizacion(), utilizando la función process_time() de la librería nativa Time. Para una mayor comparabilidad, se excluyeron la adquisición y el procesamiento de datos.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Encuesta de Experiencia.
Para evaluar la usabilidad de cada librería se llevó a cabo una encuesta entre los miembros del equipo utilizando el Cuestionario SUS (System Usability Scale) [17], para determinar el nivel de dificultad que tuvieron para lograr los objetivos planteados y valorar la experiencia al utilizar cada librería. El mismo se detalla en la Tabla 1 y consta de 10 ítems con cinco opciones de respuesta en la escala de Likert que contemplan una variedad de características de usabilidad.

Tabla 1. Cuestionario SUS

ID	Descripción
1	Usaría esta librería frecuentemente
2	Encuentro esta librería innecesariamente compleja
3	Creo que la librería fue fácil de usar
4	Necesitaría ayuda de una persona con conocimientos técnicos para usar esta librería
5	Las funciones de esta librería están bien integradas
6	Creo que la librería es muy inconsistente
7	La mayoría de las personas aprendería a usar esta librería en forma muy rápida
8	Encuentro que la librería es muy difícil de usar
9	Me siento confiado al usar esta librería
10	Necesité aprender muchas cosas antes de ser capaz de usar esta librería

4 Resultados

El objetivo de este estudio fue brindar un panorama general de las librerías de visualización de datos más utilizadas en Python e identificar sus ventajas y desventajas para la creación de gráficos efectivos. En esta sección se describen los principales hallazgos.

4.1 Selección de Librerías de Visualización.

Una vez definidas las librerías, se asignó al equipo de programadores la tarea de replicar una visualización “modelo” con la información necesaria para el seguimiento de los efectos de una medicación contra parásitos en ganado bovino en un período de 30 días.

El modelo consistía en un gráfico de líneas con elementos básicos tales como: título, leyendas, etiquetas de datos, marcadores y anotaciones. Por otro lado, se requería modificar parámetros como el grosor de la serie de datos o el grado de opacidad de las líneas de cuadrícula, que definía su presencia o ausencia.

Las tareas necesarias para replicar el modelo tuvieron diversos grados de complejidad dependiendo del tipo de librería y visualización (estática o interactiva) y del nivel de experiencia de los integrantes del equipo.

En este sentido, cabe mencionar que todas las librerías analizadas cuentan con una amplia documentación, galerías de ejemplos y guías para el usuario, lo que permitió que aquellos integrantes que no estuvieran familiarizados pudieran completar las tareas requeridas. La Fig. 2 muestra un ejemplo de los gráficos logrados con cada librería.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

6

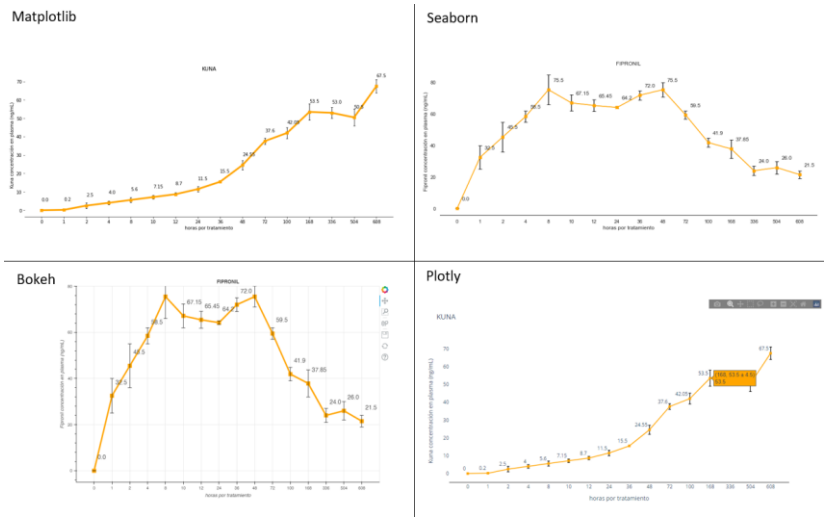

Fig. 2. Salida resultante de cada librería

Salvando las diferencias inherentes a cada tecnología, todas las librerías permitieron arribar al gráfico modelo. El análisis detallado de las características de cada caso es descrito a continuación.

4.2 Análisis Comparativo

Visualizaciones disponibles.
En primer lugar, se analizaron los alcances de cada librería en cuanto a la creación de visualizaciones. Para ello, se tuvieron en cuenta los principales gráficos utilizados en el área de visualización de información [18].

La Tabla 1 muestra los tipos de gráficos soportados por cada librería. Si bien las cuatro librearías ofrecen las principales técnicas de visualización, se puede observar que Plotly es la que mayor cantidad de gráficos soporta, seguida por Bokeh, Seaborn, y en último lugar, Matplotlib.

En algunos casos, si bien la librería no ofrece funciones nativas para la creación de un gráfico, este se puede lograr por medio de diversas estrategias o adaptaciones de otros, como en el caso del gráfico de cascada en Matplotlib, que puede crearse usando una librería adicional o a partir de un gráfico de barras.

Tabla 2. Técnicas de visualización permitidas por cada librería.

Visualización	Estáticas		Interactivas	
	Matplotlib	Seaborn	Bokeh	Plotly
Cajas y Bigotes	✓	✓	✓	✓
Diagrama de puntos	✓	✓	✓	✓
Diagrama de Sankey	✓	-	✓	✓
Gráfico de Área	✓	✓	✓	✓

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Visualización	Estáticas		Interactivas	
	Matplotlib	**Seaborn**	**Bokeh**	**Plotly**
Gráfico de Barras	✓	✓	✓	✓
Gráfico de Burbujas	✓	✓	✓	✓
Gráfico de Cascada	-	-	-	✓
Gráfico de Dispersión	✓	✓	✓	✓
Gráfico de Línea	✓	✓	✓	✓
Gráfico de Torta	✓	✓	✓	✓
Gráfico de Violín	✓	✓	✓	✓
Gráficos Combinados	✓	-	✓	✓
Histograma	✓	✓	✓	✓
Mapa coroplético	-	✓	-	✓
Mapa de calor	✓	✓	✓	✓
Mapa de árbol (*treemap*)	-	✓	✓	✓
Proyección solar (*sunburst*)	-	-	✓	✓
Tablas	-	-	✓	✓

Líneas de Código

La Fig. 3 muestra el promedio entre los ocho miembros del equipo de líneas de código (LOC) requeridas por cada librería para la función crearVisualización().

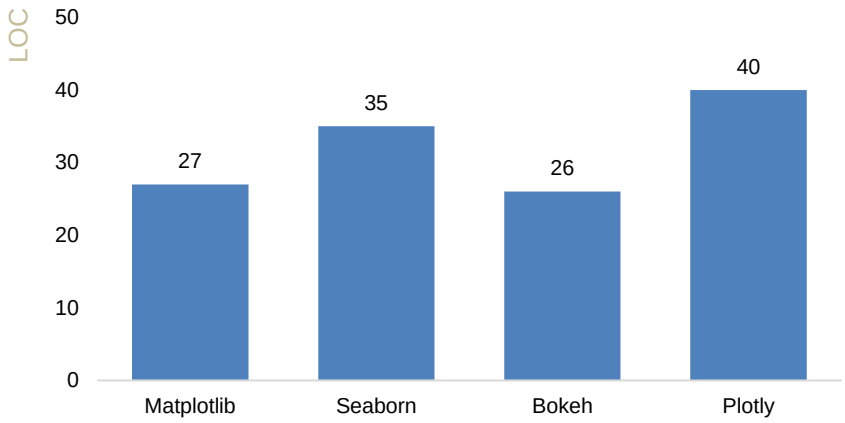

Fig. 3. Promedio de líneas de código por librería.

Si bien Plotly ofrece los resultados más estéticos y amigables para el usuario, demostró ser la librería con mayor complejidad. Esto puede deberse, en parte, a que algunos programadores recurrieron a estrategias menos directas para lograr emular el gráfico modelo. Entre las librerías estáticas, la de menor complejidad fue Matplotlib.

Tiempo de ejecución de CPU

La Fig. 4 muestra el promedio de los tiempos de ejecución de CPU de cada una de las librerías de la función CrearVisualización(), expresado en segundos.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

8

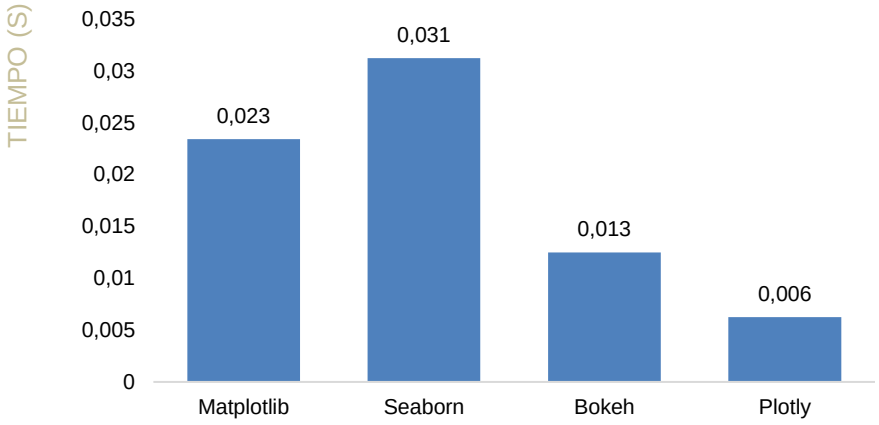

Fig. 4. Promedio de tiempo de ejecución de CPU por librería.

Teniendo en cuenta el poder de procesamiento de cada computadora como limitación, en general los ocho miembros del equipo mostraron resultados similares. Con respecto a las librerías estáticas, Seaborn (basada en Matplotlib) fue la que más tiempo consumió. En cuanto a visualizaciones interactivas, Plotly resultó ser la más efectiva, con tiempos de ejecución en el orden de los milésimos, con Bokeh en segundo lugar.

Encuesta de experiencia

Una vez finalizadas las tareas necesarias para replicar el gráfico modelo, los integrantes del equipo completaron una encuesta referida a su experiencia al utilizar las distintas librerías de visualización.

El equipo estuvo conformado por ocho miembros, cinco hombres y tres mujeres, de entre 19 y 26 años. En cuanto a la experiencia con el lenguaje de programación y las librerías propuestas, un miembro reportó contar con más de 5 años de experiencia; 5 integrantes contaban con 2 años o menos y 2 estaban recién iniciando su aprendizaje.

La Tabla 3 muestra el promedio de los resultados obtenidos por librería. Seaborn y Plotly obtuvieron valores por encima del promedio SUS, indicando un grado de usabilidad aceptable. En general, se observó una preferencia por Plotly que, en palabras de un miembro del equipo “resulta fácil de entender y genera resultados estéticos.”

Table 3. Media de puntaje SUS para cada librería

	Media SUS
Matplotlib	47,69
Seaborn	51,75
Bokeh	46,94
Plotly	53,38

Por otro lado, los integrantes del equipo expresaron que las galerías resultaron útiles al proporcionar ejemplos de salidas junto con el tipo de datos necesario y el código

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

subyacente, lo cual les permitió lograr el objetivo buscado realizando pequeñas modificaciones a dichos ejemplos para adaptarlo a la necesidad específica.

5 Conclusiones

Este trabajo buscó proporcionar una visión general de las librerías de visualización de información utilizadas en Python en el contexto del seguimiento y control de la salud de ganado bovino. Se analizaron y compararon las librerías más utilizadas, tanto para visualizaciones estáticas como interactivas. Por otro lado, se evaluó la experiencia del equipo de programadores al interactuar con cada librería.

Entre las librerías para visualizaciones estáticas, Seaborn resulto intuitiva y fácil de usar, mientras que Matplotlib, si bien tiene una mayor complejidad en la sintaxis, ofrece también un mayor grado de personalización.

En general, los miembros del equipo demostraron una preferencia por Plotly, que además de ser la que mayor cantidad de gráficos soporta, cuenta con una comunidad activa y una documentación completa que incrementa su usabilidad.

Los resultados contribuyen a afirmar que la elección de una u otra librería dependerá de las características y las necesidades de cada proyecto, como así también la capacidad y experiencia de cada programador.

Por otro lado, esto demuestra la necesidad de incorporar técnicos o profesionales programadores para el desarrollo de las visualizaciones en otras disciplinas donde los resultados necesitan ser analizados mediante gráficos.

Referencias

[1] Y. P. Cid, T. P. Ferreira, V. S. Magalhães, T. R. Correia, and F. B. Scott, “Injectable fipronil for cattle: Plasma disposition and efficacy against Rhipicephalus microplus,” *Vet Parasitol*, vol. 220, pp. 4–8, Apr. 2016, doi: 10.1016/J.VETPAR.2016.02.008.

[2] D. Gotz and D. Borland, “Data-Driven Healthcare: Challenges and Opportunities for Interactive Visualization,” *IEEE Comput Graph Appl*, vol. 36, no. 3, pp. 90–96, 2016, doi: 10.1109/MCG.2016.59.

[3] K. Rivers, E. Harpstead, and K. Koedinger, “Learning curve analysis for programming: Which concepts do students struggle with?,” *ICER 2016 - Proceedings of the 2016 ACM Conference on International Computing Education Research*, pp. 143–151, Aug. 2016, doi: 10.1145/2960310.2960333.

[4] “Matplotlib — Visualization with Python.” https://matplotlib.org/ (accessed Oct. 14, 2022).

[5] M. Waskom, “seaborn: statistical data visualization,” *J Open Source Softw*, vol. 6, no. 60, p. 3021, Apr. 2021, doi: 10.21105/JOSS.03021.

[6] “Plotly Python Graphing Library.” https://plotly.com/python/ (accessed Oct. 14, 2022).

[7] “Bokeh.” https://bokeh.org/ (accessed Oct. 20, 2022).

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

[8] J. VanderPlas, “Python’s Visualization Landscape (PyCon 2017) - Speaker Deck,” 2017. https://speakerdeck.com/jakevdp/pythons-visualization-landscape-pycon-2017 (accessed Oct. 14, 2022).

[9] N. P. Rougier, “Scientific Visualization: Python + Matplotlib,” Nov. 2021. https://www.labri.fr/perso/nrougier/scientific-visualization.html (accessed Oct. 14, 2022).

[10] “Python tools for data visualization — PyViz 0.0.1 documentation.” https://pyviz.org/index.html (accessed Oct. 14, 2022).

[11] J. Bednar, “Anaconda | Python Data Visualization 2018: Why So Many Libraries?,” Nov. 15, 2018. https://www.anaconda.com/blog/python-data-visualization-2018-why-so-many-libraries (accessed Oct. 14, 2022).

[12] I. Stancin and A. Jovic, “An overview and comparison of free Python libraries for data mining and big data analysis,” *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings*, pp. 977–982, May 2019, doi: 10.23919/MIPRO.2019.8757088.

[13] Y. Acar *et al.*, “Comparing the Usability of Cryptographic APIs,” *Proc IEEE Symp Secur Priv*, pp. 154–171, Jun. 2017, doi: 10.1109/SP.2017.52.

[14] “(PDF) Python for Smarter Cities: Comparison of Python libraries for static and interactive visualisations of large vector data.” https://www.researchgate.net/publication/358919728_Python_for_Smarter_Cities_Comparison_of_Python_libraries_for_static_and_interactive_visualisations_of_large_vector_data (accessed Oct. 20, 2022).

[15] R. K. Yin, *Case Study Research and Design*, 6th ed. SAGE Publications, Inc, 2017.

[16] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*, vol. 9783642290442. Springer-Verlag Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-29044-2.

[17] J. Brooke, “SUS: A ‘Quick and Dirty’ Usability Scale,” in *Usability Evaluation In Industry*, CRC Press, 1996, pp. 207–212. doi: 10.1201/9781498710411-35.

[18] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker, “Beagle: Automated Extraction and Interpretation of Visualizations from the Web,” *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, doi: 10.1145/3173574.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

ISCAT, aplicación para la integración de herramientas de análisis estático de código orientada a estudios empíricos en la calidad de software

Celeste M. Ojeda Rodríguez, Sebastián Bazterrica, Martín Cardozo, Juan Andrés Carruthers

Grupo de Investigación en Calidad de Software, Facultad de Ciencias Exactas y Naturales y Agrimensura, Universidad Nacional del Nordeste

{ceojedarodriguez, sebastianbazterrica, martincardozo}@gmail.com
jacarruthers@exa.unne.edu.ar

Resumen. La Ingeniería del Software Basada en Evidencia proporciona los medios para que la evidencia actual de la investigación acerca de la calidad en el desarrollo software pueda integrarse con la experiencia práctica. En particular, para el estudio de la calidad del código fuente se suelen utilizar métricas e indicadores vinculados con la calidad del producto de colecciones de proyectos software. Sin embargo, dada la gran cantidad de proyectos que pueden estar contenidos en estas colecciones la generación de métricas no resulta una tarea trivial. De esta manera surge la aplicación Integrated Source Code Analysis Tool (ISCAT), para dar soporte a la generación de métricas de código de un lote de proyectos software. La herramienta permite escanear un lote de proyectos de manera automática con las herramientas Sonar Qube y Source Meter.

Palabras Clave: Sonar Qube, Source Meter, Análisis estático de código, Análisis lotes de proyectos

1 Introducción

La Ingeniería del Software Basada en Evidencia o empírica proporciona los medios para que la evidencia actual de la investigación pueda integrarse con la experiencia práctica y los valores humanos en el proceso de toma de decisiones para el desarrollo y mantenimiento de software [1]. Desde la perspectiva de los investigadores sirve como una metodología para realizar una agregación imparcial de resultados empíricos, y para los profesionales, es un mecanismo para respaldar y mejorar las decisiones de adopción de tecnología [2].

En particular, la calidad en el desarrollo software, puede estudiarse desde el punto de vista de: i) la calidad del proceso de desarrollo software, y ii) la calidad del código fuente [3]. En este último caso, es necesario obtener métodos empíricos para demos-

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

trar la calidad del producto software [4] utilizando evidencia directamente relacionada con el código fuente resultante [5], siendo que modelos de calidad de procesos tales como CMMI e ISO/IEC 12207 abordan pocas prácticas para mejorar características específicas de la calidad del producto [6]. Una práctica habitual para evaluar la calidad del producto es la utilización de métricas e indicadores del código fuente vinculados con la calidad del producto [7].

El uso generalizado de repositorios para el código fuente (por ej., SourceForge, GitHub o Maven) les han otorgado a investigadores y profesionales acceso a millones de proyectos software y en consecuencia código fuente. De esta manera, los proyectos alojados en estas plataformas facilitan la construcción de colecciones. A su vez estas sirven de insumos para estudios empíricos orientados a demostrar la efectividad de las métricas como predictores de las características de calidad del software [7 - 9].

Sin embargo, dada la gran cantidad de proyectos que pueden estar contenidos en estas colecciones la generación de métricas no resulta una tarea trivial, existen ejemplos como SF100 [10], Qualitas corpus [11] y Github Java Corpus [12] que contienen 100, 112 y 14.807 proyectos java respectivamente. Aunque existan herramientas que facilitan el análisis automático del código fuente, estas en la mayoría de los casos escanean un proyecto a la vez. Por ello se necesita una herramienta que dé soporte al proceso de análisis de una colección masiva de proyectos software. ISCAT es una aplicación Python, cuya función principal es la integración de las herramientas para el análisis de código Sonar Qube y Source Meter, para analizar lotes de proyectos de forma automática.

El resto del artículo se encuentra organizado de la siguiente manera. En la sección 2, están los trabajos relacionados. En la sección 3 y 4 se presenta la aplicación desarrollada y su evaluación en un lote de proyectos respectivamente. Finalmente, en la sección 5 se encuentran las conclusiones del trabajo desarrollado.

2 Trabajos relacionados

En los últimos años un gran número de herramientas para el análisis estático del código fuente han sido reportadas en estudios empíricos para la recolección de metadatos de proyectos software [13 – 15]. Estos metadatos en forma de métricas o indicadores cuantifican características del código fuente como: tamaño, complejidad, estructura, acoplamiento, cohesión, herencia, entre otras [16].

Algunos ejemplos de herramientas recurrentes en la literatura para la generación de metadatos son: CKJM [17] que calcula 8 métricas orientadas a objetos con el código compilado en Java, entre las cuales se encuentran el conjunto de Chidamber y Kemerer [18]. PMD [19] evalúa la calidad del código reportando fallas en el código como variables sin usar, bloques “catch” vacíos o sentencias “while” vacías, y también métricas de diseño. Findbugs (ahora SpotBugs) [20], similar a PMD, busca defectos en el código Java. Sonar Qube [21] es una plataforma para análisis continuo de la calidad, fiabilidad y seguridad del código, detectando defectos, vulnerabilidades y errores. Source Meter [22] en su capa gratuita permite: calcular métricas, detectar duplicaciones, definir indicadores y crear métricas propias.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Sin embargo, estas herramientas en su mayoría, carecen de funcionalidades que permitan su ejecución de manera iterativa en un lote de proyectos. Como se ha mencionado anteriormente, esto es un limitante al momento de generar metadatos necesarios para realizar estudios empíricos del software. De esta manera surge ISCAT que busca brindar los metadatos obtenidos de colecciones de proyectos a grupos de investigación.

3 ISCAT

ISCAT es una aplicación desarrollada en el lenguaje de programación Python que busca automatizar el análisis estático del código integrando herramientas de uso libre para analizar un lote de proyectos software. A continuación, se detallan las herramientas que conforman el flujo de trabajo:

3.1 Sonar Qube [21]

Sonar Qube es una plataforma que brinda una serie de herramientas para realizar escaneo y análisis de proyectos en base al sistema de construcción utilizado en el mismo (Sonar Scanner for Gradle, Sonar Scanner for Maven, Sonar Scanner for Ant) o la plataforma/herramienta de automatización y gestión usada (Sonar Scanner for .Net, Sonar Scanner for Jenkins, Sonar Scanner for Azure Devops). Para realizar un análisis genérico que no se apoye en ninguna de estas tecnologías específicas, existe Sonar Scanner, el cual se adaptará al tipo de proyecto según el lenguaje de programación presente en los ficheros de código fuente. Asimismo, al ser una herramienta On Premise, es necesario desplegar un servidor local para su uso.

Esta herramienta calcula un total de 116 métricas, las cuales se agrupan en 9 categorías, diferenciando entre:

- **Métricas de complejidad:** miden la complejidad de los elementos del código fuente (típicamente algoritmos). Por ejemplo, complejidad ciclomática, complejidad cognitiva.
- **Métricas de duplicaciones:** indican el número de bloques de líneas duplicados, así como también, el número de archivos y líneas involucradas en duplicaciones. Por ejemplo, bloques duplicados, archivos duplicados, líneas duplicadas, etc.
- **Métricas de problemas**: indican el número de problemas planteados por primera vez en código nuevo, el recuento total de problemas marcados como falsos positivos, entre otros. Por ejemplo, problemas falsos positivos, problemas abiertos, nuevos problemas, etc.
- **Métricas de mantenibilidad:** indican el número total de code smells, la deuda técnica, es decir, el esfuerzo que tomará corregir todos los code smells, así como también, la relación entre el costo de desarrollar software y el costo de arreglarlo. Por ejemplo, deuda técnica, code smells, calificación de mantenibilidad, etc.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

- **Métricas de puertas de calidad:** indican el estado de los umbrales de calidad definidos para el proyecto. Por ejemplo, estado de la puerta de calidad, detalles de la puerta de calidad, etc.
- **Métricas de fiabilidad:** indican el número de errores y el esfuerzo para corregirlos. Por ejemplo, errores, clasificación de confiabilidad, esfuerzo de remediación de confiabilidad, etc.
- **Métricas de seguridad:** indican el número de vulnerabilidades de seguridad, el esfuerzo para corregirlas y el número de puntos de acceso de seguridad. Por ejemplo, vulnerabilidades, clasificación de seguridad, puntos de acceso de seguridad, etc.
- **Métricas de tamaño:** miden las propiedades básicas del sistema analizado en términos de cardinalidades. Por ejemplo, número de líneas de código comentadas, número de clases o métodos.
- **Métricas de pruebas:** indican la cobertura de las pruebas unitarias, y las condiciones para dicha cobertura, el número de pruebas unitarias con su duración y resultado. Por ejemplo, número de condiciones por línea, líneas a cubrir, número de pruebas unitarias omitidas, etc.

3.2 Source Meter [22]

SourceMeter es una herramienta para el análisis de código fuente con soporte a distintos lenguajes de programación, entre ellos, C/C++, Java, C#, Python y RPG. Esta herramienta permite encontrar los puntos débiles de un sistema en desarrollo analizando su código fuente.

Esta herramienta calcula un total de 119 métricas a nivel de componente, archivo, paquete, clase y método. Estas se agrupan en 6 categorías, diferenciando entre:

- **Métricas de cohesión:** miden hasta qué punto los elementos del código fuente son coherentes en el sistema. Por ejemplo, falta de cohesión en métodos.
- **Métricas de complejidad:** miden la complejidad de los elementos del código fuente. Por ejemplo, esfuerzo de Halstead, nivel de anidamiento, peso de métodos por clase.
- **Métricas de acoplamiento:** mide la cantidad de interdependencias de los elementos del código fuente. Por ejemplo, acoplamiento entre clases de objetos, número de invocaciones entrantes, número de invocaciones salientes.
- **Métricas de documentación:** mide la cantidad de comentarios y documentación de los elementos del código fuente en el sistema. Por ejemplo, líneas comentadas de código, documentación total de la API, API pública no documentada.
- **Métricas de herencia:** miden los diferentes aspectos de la jerarquía de herencia del sistema. Por ejemplo, profundidad del árbol de herencia, número de ancestros, número de hijos.
- **Métricas de tamaño:** miden las propiedades básicas del sistema analizado en términos de cardinalidades. Por ejemplo, número de líneas de código, número de clases o métodos, número de atributos.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

3.3 Modelo de datos

Al diseñar el modelo de datos para ISCAT se consideraron los datos generados por cada herramienta y de esta manera obtener un modelo uniforme que comprender las salidas obtenidas en cada caso. Para Sonar Qube se realizó un análisis sobre su API Rest y la información que brinda relacionada a los proyectos, componentes y métricas. En el caso de Source Meter se realizó un análisis sobre los resultados que se almacenan en archivos .csv que genera la herramienta como resultado de cada análisis.

Una diferencia entre ambas herramientas se puede observar en el análisis a nivel de clases que realiza Source Meter que no se evidencia en Sonar Qube, de esta manera se volvió necesaria la integración de dos tablas adicionales (classes y class_measures). Para la tabla projects se adicionaron dos campos de tipo timestamp, lastanalysissq y lastanalisyssm (uno por cada herramienta) para facilitar la actualización de los proyectos que fueron previamente analizados. A su vez, se agregó un campo de tipo varchar, tool a la tabla metrics para poder identificar a qué herramienta corresponde cada métrica.

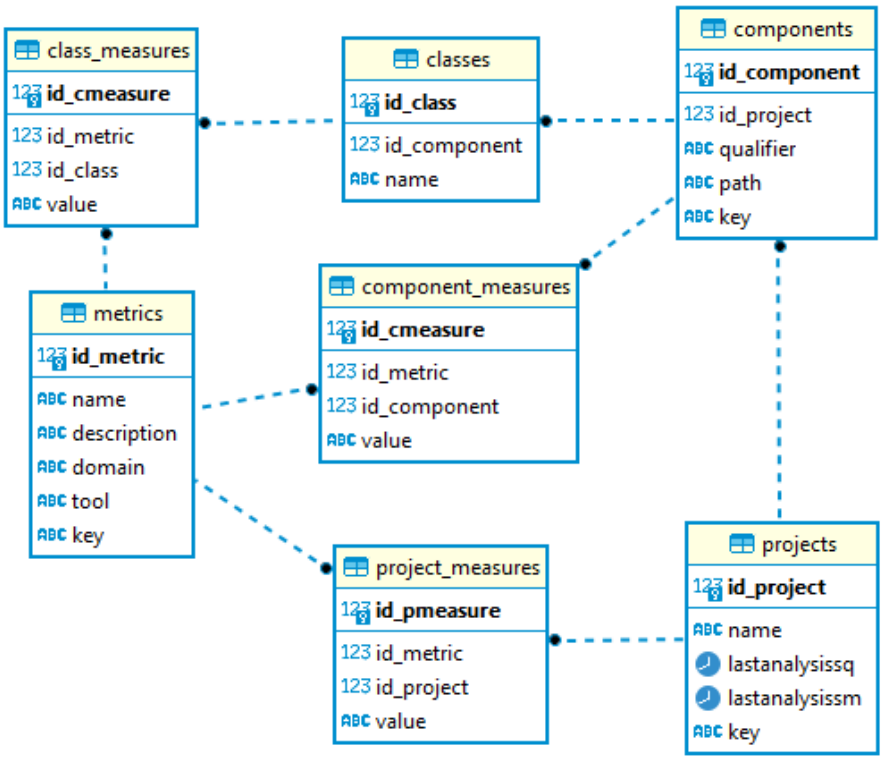

Fig. 1. Modelo de datos

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

3.4 Análisis de proyectos

La funcionalidad principal de la aplicación es la ejecución del análisis de uno o más proyectos utilizando la herramienta a elección con el objetivo de automatizar el proceso. Se simplifica el proceso de recolección de métricas de un conjunto de proyectos, que de otra manera se haría individualmente por cada proyecto y herramienta.

Para cada herramienta integrada se desarrolló un módulo, el cual se ejecuta según la elección del usuario. Además, el usuario debe proveer un directorio donde se encuentran los proyectos a analizar. Una vez proporcionadas por el usuario la elección de herramienta y el directorio de los proyectos, ISCAT puede iniciar el análisis.

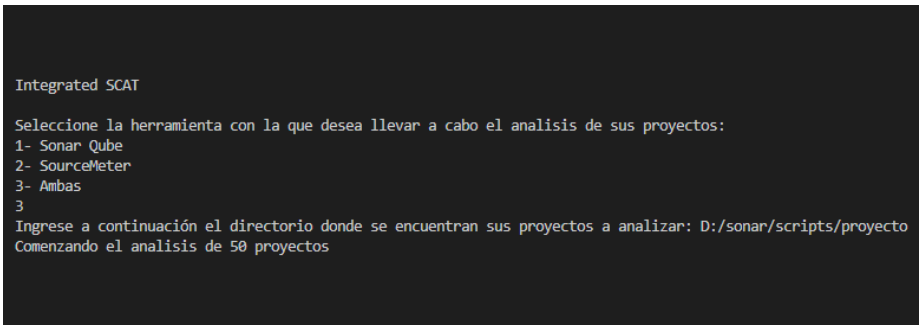

Fig. 2. Elección de herramientas e ingreso de directorio raíz de los proyectos.

Dependiendo de la herramienta se establecieron una serie de parámetros de entrada:

- **Sonar Qube:** las ubicaciones del ejecutable de Sonar Scanner, de los ficheros de código fuente, los ficheros compilados y el enlace donde está desplegado el servidor de Sonar Qube. En caso de no especificar los directorios de los archivos fuente y binarios, de manera predeterminada se utiliza la raíz de la carpeta del proyecto.
- **SourceMeter:** las ubicaciones del ejecutable de SourceMeter, de los ficheros de código fuente y la carpeta destino donde se guardan los resultados generados a partir del análisis realizado.

Al iniciar el análisis, los proyectos en el directorio brindado por el usuario almacenan los datos con las mediciones obtenidas para cada uno de ellos. Los resultados de Sonar Qube se pueden encontrar en la plataforma o servidor local, o también se puede consumir la información a través de su API Rest; y los de Source Meter se pueden encontrar de manera local en la carpeta especificada en la configuración inicial dentro del proyecto. Asimismo, para ISCAT se puede consultar a una base de datos local donde se almacenan los resultados obtenidos para cada uno de los proyectos analizados durante su ejecución.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

4 Validación

Para validar la herramienta se realizó el análisis de un lote de 50 proyectos software pertenecientes a Qualitas Corpus [14]. Esta colección de proyectos codificados en Java fue analizada por las herramientas que forman parte de Integrated SCAT para así poder medir el tiempo de ejecución y comprobar su efectividad.

El análisis se llevó a cabo durante un periodo de aproximadamente 7 hrs y 33 min. Si desglosamos este análisis por cada una de las herramientas utilizadas se puede comprobar que el análisis de los 50 proyectos con Sonar Qube llevo un total de 5 hrs y 38 min, mientras que el análisis de los mismos con Source Meter llevó un total de 1 hrs y 55 min, donde se observa un menor tiempo de análisis para la segunda herramienta.

El proceso de análisis se llevó a cabo sin supervisión del usuario hasta su finalización. Se determina así que la herramienta es capaz de agilizar los tiempos necesarios para generar métricas para las herramientas Sonar Qube y Source Meter de un lote de proyectos de forma automatizada.

5 Conclusión y futuros trabajos

Este articulo presenta ISCAT, como una herramienta dirigida a grupos de investigación en la calidad de software para ofrecer soporte en la automatización de procedimientos de análisis estático de código de un lote de proyectos software. Esta facilita el cómputo de métricas de código fuente a través de Sonar Qube y Source Meter, y la integración con otras herramientas. La funcionalidad principal de la herramienta fue validada con una colección de 50 proyectos Java, demostrando que es capaz de realizar el análisis de un lote de proyectos de manera automática con cada una de las herramientas que integran a ISCAT.

Como futuro trabajo, se considera emplear la herramienta en la generación de metadatos de una colección curada de proyectos software para la ejecución de estudios empíricos en Ingeniería del Software. Además, no se descarta la incorporación de nuevas herramientas para el análisis estático del código fuente, como ser Understand [23], para incluir información evolutiva y relativas a dependencias del sistema para extender el catálogo de métricas para cada proyecto analizado.

Referencias

1. B. A. Kitchenham, T. Dybå, and M. Jørgensen, “Evidence-based Software Engineering,” in Proceedings of the 26th International Conference on Software Engineering, 2004, pp. 273–281.
2. Dyba, T., Kitchenham, B. A., & Jorgensen, M. (2005). Evidence-based software engineering for practitioners. IEEE software, 22(1), 58-65.
3. Lehman, M. M. Laws of software evolution revisited. in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinfor-matics) vol. 1149 108–124 (Springer Verlag, 1996).

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

4. Kitchenham, B. & Pfleeger, S. L. Software quality: the elusive target. IEEE Softw. 13, 12–21 (1996).
5. Garvin, D. A. What Does “Product Quality” Really Mean? MIT Sloan Management Review 25–43 (1984).
6. García-Mireles, G. A., Moraga, M. Á., García, F., & Piattini, M. (2015). Approaches to promote product quality within software process improvement initiatives: a mapping study. Journal of Systems and Software, 103, 150-166.
7. Pecorelli, F., Palomba, F., & De Lucia, A. (2021). The relation of test-related factors to software quality: a case study on apache systems. Empirical Software Engineering, 26(2), 1-42.
8. Vescan, A., Camelia, S., & Budur, A. D. (2021). Towards a Reliability Prediction Model based on Internal Structure and Post-Release Defects Using Neural Networks. In Evaluation and Assessment in Software Engineering (pp. 379-386).
9. Yang, X., Chen, J., Yedida, R., Yu, Z., & Menzies, T. (2021). Learning to recognize actionable static code warnings (is intrinsically easy). Empirical Software Engineering, 26(3), 1-24.
10. Tempero, E. et al. The Qualitas Corpus: A curated collection of Java code for empirical stud-ies. in Proceedings - Asia-Pacific Software Engineering Conference, APSEC 336–345 (2010). doi:10.1109/APSEC.2010.46.
11. G. Fraser and A. Arcuri, "Sound empirical evidence in software testing," 2012 34th International Conference on Software Engineering (ICSE), 2012, pp. 178-188, doi: 10.1109/ICSE.2012.6227195.
12. M. Allamanis and C. Sutton, “Mining Source Code Repositories at Massive Scale using Language Modeling,” in 2013 10th Working Conference on Mining Software Repositories (MSR), 2013, pp. 207–216.
13. Ceccato, M., Capiluppi, A., Falcarin, P. & Boldyreff, C. A large study on the effect of code obfuscation on the quality of java code. Empir. Softw. Eng. 20, 1486–1524 (2015).
14. Okutan, A. & Yıldız, O. T. Software defect prediction using Bayesian networks. Empir. Softw. Eng. 19, 154–181 (2014).
15. Behnamghader, P. et al. A scalable and efficient approach for compiling and analyzing commit history. in International Symposium on Empirical Software Engineering and Measurement (IEEE Computer Society, 2018). doi:10.1145/3239235.3239237.
16. Carruthers, J. A., Diaz Pace, J. A. & Irrazabal, E. A. A Systematic Mapping Study of Empirical Studies performed with Collections of Software Projects. (2021) https://arxiv.org/abs/2109.07624.
17. CKJM - Chidamber and Kemerer Java Metrics, https://www.spinellis.gr/sw/ckjm, último acceso 07/10/2022.
18. Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. IEEE Transactions on software engineering, 20(6), 476-493.
19. PMD Source Code Analyzer, https://pmd.github.io, último acceso 07/10/2022.
20. Spotbugs, https://spotbugs.github.io, último acceso 07/10/2022.
21. Sonar Qube Documentation, https://www.sonarqube.org/documentation, último acceso 07/10/2022.
22. Source Meter, https://sourcemeter.com, último acceso 07/10/2022.
23. Understand, https://www.scitools.com/, último acceso 09/10/2022.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Reporte de validación de un sistema multiagente para la mejora de calidad de proyectos ágiles de software

Tortosa, Nicolás[1]; Teng, Jazmín[1]; Bravin, Juan[1]; Pinto, Noelia[1]; Acuña, César[1]

CInApTIC (Centro de Investigación Aplicada a TICS); UTN Facultad Regional Resistencia
{nicotortosa@gmail.com}

Abstract. En los últimos años, la demanda de productos de software con mayor funcionalidad, usabilidad, seguridad y performance ha impulsado la necesidad de adopción de herramientas para dar soporte a los procesos de desarrollo de software que, por un lado, permitan la gestión de conocimiento para la innovación y mejora de productos y procesos, y, por el otro, favorezcan la toma de decisiones ejecutivas para la renovación y adaptación de las organizaciones. Asimismo, la evolución de las prácticas de la Ingeniería de Software para adaptarse dinámicamente a diferentes escenarios, ha generado avances, también, en el seguimiento de proyectos de software. Entre estos avances se destaca la implementación de enfoques inteligentes para el ciclo de desarrollo de software, potenciando las ventajas de los campos de Inteligencia Artificial y de Ingeniería de Software. En este artículo se presentan resultados de validación del primer prototipo de un sistema multiagente que permite recomendar acciones en la gestión de proyectos ágiles de software, de forma tal de mejorar los niveles de calidad del proceso en pos de alcanzar un mejor producto de software final.

Keywords: Ingeniería de Software, Calidad de Software, Sistemas multi-agentes, Enfoque inteligente

1 Introducción

El seguimiento y control de proyectos ágiles de software tiene como objetivo fundamental el monitoreo de todo el proceso de desarrollo del producto que se está construyendo. Cuanta más asistencia automática se ofrezca en esta cuestión, más efectivos serán los resultados obtenidos posibilitando mejorar los niveles de calidad asociados al proceso, evitando desviaciones en el proyecto o, al menos, posibilitando su detección temprana.

Por ello, cada vez más las organizaciones se inclinan por incorporar a su stack de herramientas de gestión, el uso de aplicaciones integradas a sus procesos de negocios, de forma tal de garantizar la continuidad operacional de manera confiable y segura. Por parte de la industria del software, las empresas buscan optar por herramientas que incrementen la automatización de procesos tradicionalmente manuales, por ejemplo: seguimiento de product backlog, control de parámetros críticos, reportes estadísticos, etc.

Teniendo en cuenta esta realidad, sumado al dinamismo de escenarios actuales que requiere la adopción de enfoques ágiles para el desarrollo de productos de software, en

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

publicaciones anteriores se ha presentado el framework AQF [1-2], una propuesta que integra un modelo de calidad (QuAM) junto a una herramienta de software (QuAGI) que permite la automatización de dicho modelo y que ayuda al seguimiento y gestión de la calidad de proyectos desarrollados bajo prácticas ágiles.

A partir de la implementación de AQF en ambientes reales de producción de software, se lograron llevar a cabo diversas experiencias de validación. Como resultado, y gracias al feedback recibido por parte de los equipos participantes, se ha observado la necesidad de enriquecer el framework de forma tal de ofrecer una nueva herramienta que permita liberar de trabajo de monitorización manual de los proyectos a quienes cumplen roles tales como administradores, líderes de proyecto, etc.

Esto ha dado origen al desarrollo de la herramienta i-QuAGI [3-4-5], un sistema multi-agentes (SMA), diseñado bajo el enfoque basado en la ingeniería de software orientada a agentes [6], que tiene como objetivo recomendar acciones al equipo de forma tal de mejorar los niveles de calidad del proceso de desarrollo de software. Se busca, entonces, incorporar a AQF una herramienta que dé soporte al equipo de desarrollo, a partir de recomendaciones automáticas que surjan del seguimiento del proyecto ágil y sus actividades, las cuales muchas veces son afectadas por acciones en segundo plano que pasan desapercibidas e impactan negativamente en los niveles de calidad del proceso de desarrollo asociado.

El objetivo de este artículo se centra en presentar resultados de validación respecto a la integración del sistema i-QuAGI, al funcionamiento de AQF, usando una suite de datos de prueba para verificar la interacción entre componentes que configuran actualmente al framework.

El resto del artículo se estructura de la siguiente forma: la sección 2 describe brevemente al SMA i-QuAGI; la sección 3 presenta el diseño de la experiencia de validación y metodología seguida; a continuación, la sección 4 expone los primeros resultados obtenidos y el análisis que se ha llevado adelante con ellos; por último, la sección 5 resume las conclusiones a las que se llegó durante la realización de este trabajo y se presentan trabajos futuros.

2 Características de diseño e implementación de i-QuAGI

El SMA i-QuAGI ofrece 2 enfoques en su implementación, contribuyendo en el seguimiento y evaluación de calidad de procesos ágiles de desarrollo de software. Por un lado, el sistema, apoyado en técnicas de inteligencia artificial, permite la monitorización automática de proyectos ágiles[1] que son gestionados a través del framework AQF. Y, por otro lado, da soporte a los procesos de toma de decisiones asistiendo a responsables de proyectos mediante reportes que informan sobre recomendaciones para mejorar la evaluación de calidad del proyecto en cuestión.

[1] Proyectos de desarrollo de software guiados por procesos y enfoques ágiles tales como SCRUM, XP, etc

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

En el caso de i-QuAGI, los agentes que componen el SMA son: *Agente "Product Backlog"*, que detecta eventos generados[2] a partir de la gestión del product backlog en cada proyecto registrado al usar QuAGI desde el framework AQF, y el *Agente "Evaluación de Calidad"*, que detecta eventos que pueden afectar de forma negativa[3] la calidad del proyecto y reacciona emitiendo alertas y/o recomendaciones, a partir de los datos generados en la gestión de los componentes de la Evaluación de Calidad del proceso ágil en cuestión. Ambos agentes funcionarán de forma autónoma y transparente a las personas usuarias de QuAGI, sin que esto obstaculice la gestión de los proyectos activos.

Derivado del relevamiento de necesidades que se realizó a líderes de proyecto que utilizan AQF para la gestión de sus proyectos ágiles, a continuación, se listan las funcionalidades a las que responde el sistema i-QuAGI, en esta primer versión a validar:

- *Administración de Usuarios*: el sistema i-QuAGI permite el acceso a dos tipos de usuario: Rol de "Project Manager" (responsable máximo del Proyecto) y Rol de "System Manager" (responsable técnico de la configuración del SMA).
- *Registro de alertas que afecten los niveles de calidad del proceso*: Las notificaciones se registran como alertas activas en el SMA.
- *Notificación automática de recomendaciones al rol "Project Manager"*: A medida que se avance en la gestión de cada proyecto con QuAGI, el SMA detecta eventos que afecten la calidad asociada al proceso y emite alertas automáticas que se traducen en recomendaciones para ajustes de parámetros en el mismo proyecto.
- *Emisión de Reportes*: El sistema i-QuAGI permite la emisión de informes de interés para quien administre cada proyecto ágil gestionado. Actualmente provee reportes estadísticos de alertas y recomendaciones emitidas, e informes de avance histórico en la evaluación de calidad del proceso asociado en cada caso.

Para el correcto funcionamiento de un SMA, es imprescindible que los agentes que integran el sistema se comuniquen, y que su comunicación pueda implementarse de forma sencilla e inteligible. En el desarrollo e implementación de i-QuAGI se ha utilizado para el proceso de comunicación y paso de mensajes entre agentes, lo definido en el estándar de Foundation of Intelligent Physical Agents (FIPA) [7].

Los procesos asociados a cada uno de los agentes del sistema i-QuAGI, dependerán no solo de la comunicación entre ellos, sino de su integración con los componentes, ya existentes, de AQF: QuAM y QuAGI. Así, la información que se genere producto de la comunicación entre agentes, tendrá su origen en el uso de QuAGI por parte de cada proyecto considerado. Por ello, para la parte lógica del SMA, se diseñaron componentes que exponen recursos REST para el intercambio de información, que proporcionará al sistema la capacidad de analizar, interpretar y manipular contenido para poder ser devuelto de una manera sencilla hacia quienes hagan uso de i-QuAGI.

[2] Para conocer más información respecto a eventos que se detectarán, se puede acceder al trabajo publicado y referenciado en [3]

[3] Para conocer cuáles son los atributos que miden el impacto negativo en la calidad de procesos ágiles de desarrollo de software, consultar el trabajo publicado y referenciado en [1]

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Tal como se observa en la Figura 1, el SMA i-QuAGI presentará sus componentes en una pantalla inicial en forma de dashboard, donde se incluirá un set de indicadores relevantes para las personas que administren los proyectos en curso.

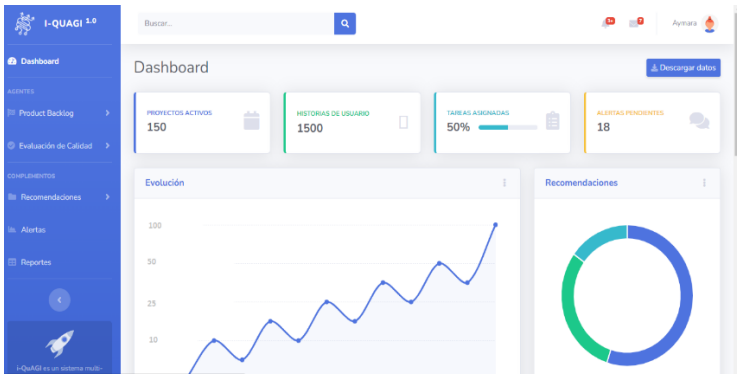

Fig. 1. Dashboard de i-QuAGI

Las notificaciones automáticas respecto a Recomendaciones y Alertas, se pueden implementar a través de las siguientes opciones: correos electrónicos enviados a la persona que lidera cada proyecto (en cada momento que se registre una alerta o recomendación, o en un único momento, de acuerdo a la periodicidad deseada), o mensajes directos en QuAGI en forma de avisos. La configuración dependerá de la decisión de quien administre el proyecto, de forma tal que, aun cuando no se esté haciendo uso en primer plano de i-QuAGI, el seguimiento y control de la calidad del proceso siga activa a través del SMA, el cual reaccionará ante eventos que pudieran afectar los niveles de calidad emitiendo recomendaciones y alertas, que se informarán de forma automática.

La arquitectura y la configuración de este primer prototipo del sistema i-QuAGI, se implementa siguiendo además un entorno simulado, donde se adoptaron las siguientes políticas de sistema que podrían ser modificadas por otras implementaciones:

- *Política de detección de eventos*: la detección de eventos puede ser considerada aprobada o desaprobada, de acuerdo a si corresponde con lo especificado en el modelo de Requerimientos del Dominio del sistema i-QuAGI [8].
- *Política de emisión de alerta*: el Agente Product Backlog puede ofrecer el servicio de alertas, en caso de que el evento detectado aprobado se derive del seguimiento del proyecto usando QuAGI. En la Figura 2, se observa cómo iQuAGI informa sobre alertas al usuario.
- *Política de emisión de recomendaciones*: el Agente Evaluación de Calidad puede ofrecer el servicio de recomendaciones, en caso que una alerta sea atendida y se requiera datos de mejora de los niveles de calidad del proyecto ágil.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

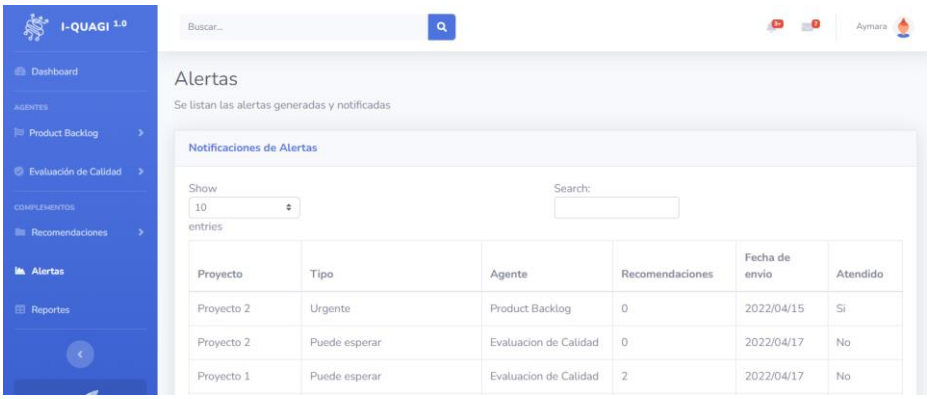

Fig. 2. Pantalla de alertas de i-QuAGI

Para ejecutar las pruebas de simulación del entorno, se trabajó con un conjunto de datos de pruebas disponibles en [9] correspondientes a requerimientos de software expresados como historias de usuario, y que formarán parte de un proyecto de prueba, cuya gestión se realizará a través del uso del framework AQF.

3 Diseño de experiencia de validación

En esta sección se describe la experiencia de validación diseñada con el objetivo de demostrar la correctitud y completitud del modelo diseñado para i-QuAGI como herramienta de monitorización automática del proyecto ágil cuya calidad de proceso se quiere evaluar, así como las ventajas que aporta para el seguimiento e implementación de dicho proceso.

El proceso de validación contempla el uso de estudios de casos, un enfoque adecuado para llevar adelante actividades de validación en ingeniería de software, ya que los objetos de estudio son fenómenos contemporáneos, que resultan difíciles de estudiar de forma aislada [10], como resulta ser el caso de procesos de desarrollo de software llevado a cabo por equipos que implementan, por ejemplo, prácticas ágiles en sus ciclos.

Cabe destacar que la experiencia de validación abarcó el análisis de 2 variables independientes y subjetivas:

- *Utilidad del sistema i-QuAGI*: Validar que las alertas y recomendaciones ofrecidas son correctas, respondiendo al diseño del modelo QuAM y lo registrado por QuAGI.
- *Aporte a la evaluación de calidad*: Validar que la monitorización automática ofrecida por i-QuAGI, mejora la calidad del proceso asociado al proyecto ágil que se evalúa.

El estudio de caso se presenta de acuerdo a la siguiente metodología:

1. *Diseño del Estudio*: Esta actividad comprende la definición de componentes asociados al fin del estudio, para ello se establecen 2 tareas relevantes:

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

   a. Definición de Objetivos
   b. Definición de Hipótesis
2. *Preparación para la recolección de datos*: Esta actividad comprende la planificación y disposición de elementos necesarios para la ejecución de la experiencia, incluyendo:
   a. Especificación de la población y de la muestra
   b. Planificación de la experiencia
3. *Recolección de evidencias*: Esta actividad permite generar inputs para su posterior análisis.
4. *Análisis de los datos*: Esta actividad tiene por objetivo describir los resultados obtenidos desde un punto de vista cuantitativo y cualitativo.

A continuación, se expone el estudio de caso ejecutado para una primera validación del prototipo de i-QuAGI, a partir de la simulación con datos de prueba de un proyecto ágil utilizando QuAGI y su evaluación de calidad en base a los componentes establecidos por QuAM.

3.1 Estudio de Caso: Centro de Reciclado online

De acuerdo con lo antes expuesto, se propone, para la experiencia que aquí se presenta, un estudio de caso en el que se simula la realización del proceso de *Desarrollo de una aplicación web para Centro de Reciclado de la ciudad*, utilizando QuAGI como herramienta de gestión del proyecto.

1. *Diseño de estudio*
   a. *Objetivo del estudio:* Validar el diseño e implementación de la primera versión del prototipo obtenido para i-QuAGI
   b. *Hipótesis:* El uso de la herramienta i-QuAGI mejorará los niveles de calidad correspondientes al proceso del proyecto ágil que se evalúa
2. *Preparación de la experiencia*
   a. *Población y muestra:* El caso de prueba contempla el seguimiento de un product backlog *de 51 historias de usuario*, que se agruparán en aproximadamente *5 sprints*, teniendo en cuenta un equipo conformado por: Cliente, 2 Developers y 1 Project Leader. Para la muestra se considerará la ejecución del primer sprint, conformados por 15 historias de usuario.
   b. *Planificación de la experiencia:* En primera instancia se procede a la carga de las historias de usuario como parte del product backlog del proyecto inicializado en QuAGI. Luego, en segundo lugar, se realiza el agrupamiento de historias de usuario en sprints. En la Figura 3, se observa un ejemplo de cómo QuAGI muestra esta información.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Fig. 3. QuAGI y la gestión web de proyectos ágiles

3. *Recolección de evidencias*
   Para la ejecución del estudio de caso, se han definido 3 puntos de control, de forma tal que la validación en cada instante permita validar el funcionamiento del sistema multiagente. Los instantes de análisis son: a) Al iniciar el proyecto y luego de la primera actualización del product backlog, b) Durante el avance del sprint, a mitad de proceso, c) Finalizando el sprint.
   El proceso de simulación avanza de acuerdo a la evolución del proyecto en cuestión, a partir de la actualización que se produce haciendo uso de QuAGI.
   Y, resulta importante destacar que, entre cada punto de control se tomará registro de los niveles de calidad correspondientes a cada componente del modelo QuAM, y de las variaciones, si las hubiera, para un posterior análisis de resultados. Además, se registran peculiaridades que se consideran importantes en el impacto del uso del sistema i-QuAGI.
4. *Análisis de los datos*
   Los resultados obtenidos y el análisis realizado en la ejecución de este estudio de caso, se exponen en la siguiente sección del artículo.

4 Resultados de la experiencia de validación

El estudio de los resultados de validación se enfoca en la contribución del sistema i-QuAGI en la evaluación de calidad de procesos ágiles, respecto a cada uno de los componentes considerados en el modelo QuAM[4]. Por ello, luego de la implementación y de la simulación del uso de i-QuAGI, se llevó adelante el análisis de los datos que se lograron recolectar durante la ejecución del estudio de caso.

Primeramente, se tomó registro de los niveles de calidad al inicializar el proyecto, finalizando el punto de control a) descripto en la fase 3 de la descripción del estudio de caso. En ese momento, como se observa en la Figura 4, los niveles de calidad para cada

[4] Se pueden consultar cada uno de ellos en la presentación referenciada en [1].

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

componente en el proceso fueron en su mayoría de nivel MALO[5], según informó el reporte de *Evaluación de calidad* proporcionado en la herramienta QuAGI.

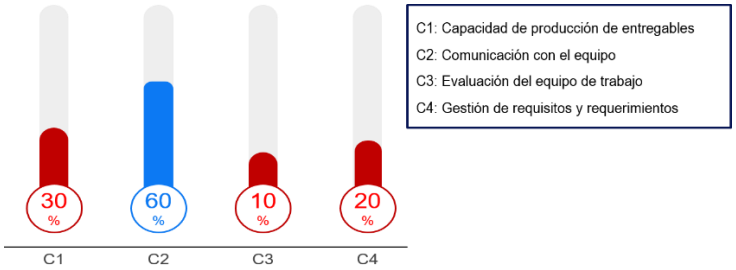

Fig. 4. Niveles de calidad del proceso ágil luego de la primera fase de control

Al mismo tiempo, el sistema i-QuAGI comenzó a monitorear el proceso ágil de desarrollo de software emitiendo las primeras alertas, al detectar eventos que, en segundo plano y fuera del alcance visible a través del seguimiento automático, pudieran estar afectando negativamente los niveles de calidad asociados. A medida que los eventos se fueron detectando cuando se mina el proyecto con historias de usuario, el *Agente Product Backlog* del sistema i-QuAGI ha emitido alertas, tal como se muestran a continuación en la Tabla 1.

Tabla 1. Alertas emitidas por evento detectado – Agente Product Backlog

Evento	**Alertas emitidas**
Creación de historias de usuario sin estimar	13
Creación de historias de usuario sin detalle de tareas	15
Ausencia de criterios de aceptación en las historias de usuario definidas	15
Existencia de historias de usuario sin priorizar	10

Asimismo, el *Agente Evaluación de Calidad* también detectó eventos que fueron traducidos por el SMA en alertas informadas al rol Project Manager. En este caso, los eventos generaron la emisión de 1 (una) alerta en cada caso, y fueron los que se detallan en la Tabla 2.

Tabla 2. Eventos detectados – Agente Evaluación de Calidad

Evento
Más del 40% de historias de usuario sin puntos de historia
Más del 40% de historias de usuario sin validar los criterios de aceptación
Más del 60% de historias de usuario no se adecúa a template en su redacción.
Más del 50% de las tareas sin responsables.
Más del 60% de historias de usuario sin especificar su prioridad.

[5] La figura 4 muestra los niveles de calidad en función a porcentajes de cumplimiento teniendo en cuenta el máximo valor posible en la escala establecida por QuAM. Los componentes 1, 3 y 4 obtuvieron un nivel de calidad MALO. El componente 2, un nivel de calidad BUENO

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

El sistema i-QuAGI permite, al rol Project Manager, la configuración de la metodología de notificación de alertas más adecuada al contexto. En la simulación, que aquí se presenta como estudio de caso, las alertas emitidas fueron notificadas a través de un único email, habiendo pasado 3 horas de iniciado el proyecto en la plataforma de seguimiento QuAGI. De esta manera, se comprobó el correcto funcionamiento de notificación automática demostrando que el sistema i-QuAGI funciona en segundo plano, sin necesidad de intervención humana.

Con las alertas emitidas, quien administra el Proyecto en cuestión puede decidir, a partir del dashboard de i-QuAGI, *atender* las alertas para que el sistema genere las recomendaciones correspondientes. Por ejemplo, se seleccionó uno de los grupos de alertas "*Creación de Historias de Usuario sin estimar"* del Agente Product Backlog, para que el sistema i-QuAGI muestre las recomendaciones asociadas, tal como puede observarse en la Figura 5.

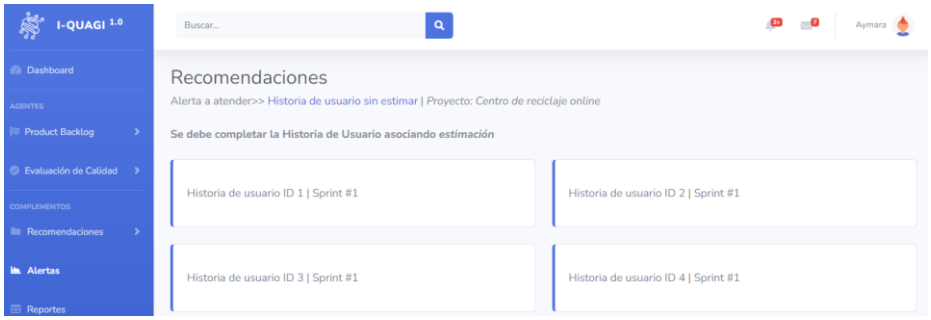

Fig. 5. Ejemplo de pantalla de *Recomendaciones*

A partir de allí, el sistema i-QuAGI permite la redirección automática a la herramienta de seguimiento del proyecto para que se ajusten los parámetros recomendados y se verifique, entonces, la variación en los niveles de calidad. Esto se produce gracias a la integración entre la plataforma QuAGI y el SMA i-QuAGI, que también fue validada en esta experiencia y ejecución de estudio de caso.

Luego de actualizar el proyecto a partir de las recomendaciones del SMA al atender el 100% de las alertas del grupo "*Creación de historias de usuario sin estimar*", casi el 75% de las alertas del grupo "*Ausencia de criterios de aceptación en las historias de usuario definidas*" (ambos grupos de alertas emitidas por el Agente Product Backlog) y la alerta "*Más del 60% de historias de usuario no se adecúa a template en su redacción*" (emitida por el Agente Evaluación de Calidad) , se volvió a consultar los niveles de calidad del proceso ágil, usando la funcionalidad de *Reportes de Calidad* de QuAGI, obteniéndose notables mejores en la evaluación. Tal como se ve en la Figura 6, el Componente 1 (C1) correspondiente a la "Capacidad de producción de entregables", mejora su nivel de calidad aumentando su porcentaje de cumplimiento, desde un 30% inicial a un 80%, alcanzando el nivel *Bueno*. Y, el Componente 4 (C4) correspondiente a la "Gestión de Requisitos y Requerimientos", mejora su nivel de calidad desde un 20% a un 90%, alcanzando un nivel *Muy Bueno*.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

10

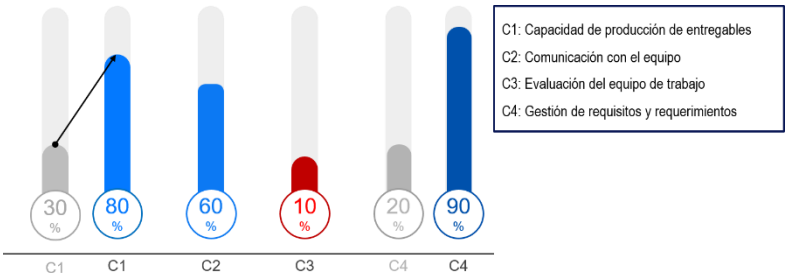

Fig. 6. Niveles de calidad del proceso ágil luego de la segunda fase de control

Al tratarse de una simulación, trabajando el estudio de caso con historias de usuario correspondientes a un proyecto ficticio y con la participación del equipo técnico responsable del desarrollo del sistema, no fue posible validar el funcionamiento respecto a los Componentes 2 y 3, ya que requieren de la interacción y flujo de información dentro de un equipo real abocado a un proyecto ágil de desarrollo de software.

4.1 Reporte de validación de la experiencia

Según proponen Runeson y Host en [11], el proceso de validación culmina con la presentación de informes que destacan ciertos aspectos de la experiencia. Debido a la característica de este estudio de caso, se reemplaza esta actividad con una más integral que, basada en informes cuantitativos, presenta conclusiones que caracterizan a la experiencia. En el caso de la validación de i-QuAGI particularmente, es necesario que se lleve a cabo tanto a nivel micro (a nivel de cada uno de los agentes) como macro (a nivel de la estructura global, entendida como el resultado de las interacciones entre las partes componentes).

Por tanto, y teniendo en cuenta la hipótesis planteada, se observa que el uso complementario de i-QuAGI, cuando se realiza el seguimiento del proyecto ágil a través de QuAGI, ha beneficiado el proceso de evaluación de la calidad del proceso de desarrollo asociado, mejorándose notablemente los niveles obtenidos en los componentes que han podido evaluarse y cuya evolución se mostró más arriba, en esta sección.

Asimismo, durante la experiencia de validación, se ha decidido registrar el tiempo que tomaría mejorar los niveles de calidad comparando el uso o no de i-QuAGI. Como se observa en la Figura 7, sin utilizar el apoyo automático del sistema i-QuAGI mejorar los niveles de calidad asociados al Componente 1 y al Componente 4, requirió un total de 7hs. En cambio, si se utiliza de forma complementaria la asistencia de i-QuAGI, para alcanzar mejores niveles de calidad, se requirieron solo 2hs de trabajo total. Entonces, utilizar el SMA i-QuAGI como asistente de seguimiento del proyecto ágil ahorra al menos 5 horas de trabajo en el proyecto, pues resulta de mucha ayuda a quien toma el rol de administración del proyecto, pues las alertas y recomendaciones habían permitido detectar desviaciones en el registro de requerimientos del cliente y completar información valiosa para el seguimiento del proyecto.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

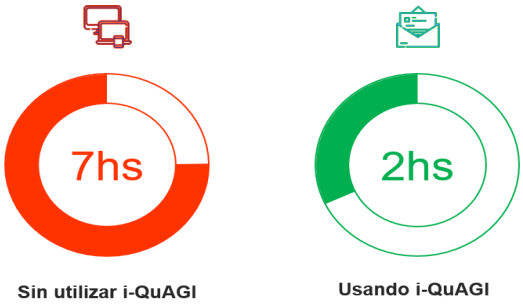

Fig. 7. Comparación de tiempos requeridos en la mejora de calidad

La experiencia de ejecución del estudio de caso que aquí se presenta, ha permitido, además, validar el funcionamiento mancomunado entre los 2 agentes que componen a i-QuAGI, la integración funcional con la herramienta QuAGI y su alineación a las definiciones establecidas en QuAM. Esto ha permitido demostrar que los principales beneficios de un enfoque basado en agentes, como el que aquí se expone, provienen de su flexibilidad, adaptabilidad y descentralización. La característica reactiva que se ha impreso sobre los agentes del sistema i-QuAGI, permite que se realice un monitoreo constante de sus entornos y la reacción automática a los eventos que se detectan y que afectan a los niveles de calidad del proceso.

Se ha observado, por último, la necesidad de incorporar al dashboard del sistema multiagente, un resumen de highlights respecto a proyectos similares anteriores que permitan, a quienes hacen uso de i-QuAGI, comparar resultados históricos, en los que, además, el sistema se basa para enriquecer su base de conocimiento.

5 Conclusiones y Trabajos Futuros

En la industria de software, la toma de decisiones es seguramente la tarea más importante y compleja para quienes administran los proyectos de desarrollo de software. Este trabajo ha presentado el reporte de validación de una propuesta basada en técnicas y metodologías de inteligencia artificial para asistir a quienes gestionan proyectos ágiles, con el objetivo de dar un apoyo concreto a la etapa de evaluación de calidad asociada al proceso de desarrollo del software.

Los agentes inteligentes pueden actuar en nombre de usuarios o entidades involucradas en un sistema, pero los resultados dependen de una rigurosa aplicación de la metodología adoptada. Por ello, es importante señalar que el procedimiento de seguimiento y monitorización automática descrito anteriormente no reemplaza las actividades de control y gestión por parte de quien administre el proyecto ágil. Por el contrario, sirve como complemento al acelerar el proceso de eliminación de errores en el seguimiento del proyecto ágil antes de tiempo favoreciendo el incremento en los niveles de calidad del proceso asociado, lo que, seguramente, impactará en una mejor calidad del producto que se obtenga. Esto ha sido demostrado a partir de los resultados preliminares obtenidos en la experiencia de validación que se ha descripto y que muestran notables mejoras

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

en los niveles de calidad, a partir de la atención a las recomendaciones ofrecidas por i-QuAGI, y reducción del tiempo insumido en el proceso de seguimiento y evaluación de calidad asociada al proyecto ágil en cuestión.

Como trabajos futuros se prevé extender la experiencia de validación aquí presentada, diseñando nuevos estudios de caso que contemplen proyectos que sean guiados por prácticas ágiles y en los que participen equipos de trabajo reales, de forma tal de observar el rendimiento de la interacción entre QuAGI y el sistema inteligente i-QuAGI, en todos los componentes definidos por QuAM. Y, completar el modelo del sistema i-QuAGI, contemplando la inclusión de datos históricos que complementen la información proporcionada para la toma de decisiones respecto al proceso ágil de desarrollo de software.

Referencias

1. Pinto, N. Framework para la evaluación de calidad de proyectos ágiles de software. (2020). Tesis Doctoral. Universidad Nacional de La Plata.
2. Pinto, N., Acuña, C. et al. Validación de la reingeniería aplicada sobre la primera versión de Agile Quality Framework. XIX Simposio Argentino de Ingeniería de Software (ASSE)-JAIIO 47 (CABA, 2018). 2018.
3. Teng, J., Maidana, L., Tortosa, N., Pinto, N., Acuña, C. “i-QuAGI: Primera aproximación al diseño conceptual de un sistema multi-agente para la gestión y evaluación de calidad de proyectos ágiles de software”. CONAIISI 2021
4. CADI 2022
5. Tortosa, N.; Pinto, N.; Acuña. C.; Tomaselli, G. (2022). Sistemas inteligentes como herramienta para el seguimiento de proyectos Agiles: una revisión sistemática de la literatura. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação, 1(15).
6. Abdalla, R., & Mishra, A. (2021). Agent-Oriented Software Engineering Methodologies: Analysis and Future Directions. Complexity, 2021, 1-21.
7. Información disponible en http://www.fipa.org/
8. Teng, J., Maidana, L.,Tortosa, N., Pinto, N., Acuña, C. (2021) i-QuAGI: Un enfoque inteligente para la gestión de calidad en proyectos de software ágiles. IV Jornadas de Calidad de Software y Agilidad.
9. Información disponible en https://data.mendeley.com/datasets/7zbk8zsd8y/1
10. Kitchenham, B. A., Dyba, T., & Jorgensen, M. (2004). Evidence-based software engineering. In Proceedings. 26th International Conference on Software Engineering (pp. 273-281). IEEE.
11. Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), 131.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Modelos Ágiles en la Administración Pública: Análisis de Aplicabilidad

Adolfo Flaschka[1], Carlos Cortés Parra[1], Gabriela Rivero[2],

Joaquín Beck[3], Pablo Provasi[3]

[1] Universidad Tecnológica Nacional - Facultad Regional Resistencia, UTN-FRRe, French 414, H3500 Resistencia, Argentina
[2] Universidad Nacional de Misiones, UNaM-FCEQyN, Félix de Azara 1552, N3300 Posadas, Argentina
[3] Universidad Nacional del Nordeste, UNNE-FaCENA, Av. Libertad 5470, W3400 Corrientes, Argentina
{leoflaschka, cecortes, reqaby, joaquinrbeck, pprovasi}@gmail.com

Resumen. En este trabajo sobre peopleware se aborda una mirada de la administración de grupos de trabajo o fuerza laboral en organismos de gobierno o sector público. En la introducción describimos aspectos conceptuales de peopleware y sus ideales para alcanzar la satisfacción laboral plena. En la segunda sección presentamos un breve análisis desde la idiosincrasia, casos específicos en organismos públicos. En la tercera sección proponemos cómo implementar prácticas de peopleware en organismos de gobierno. En la cuarta sección proponemos un ejemplo práctico sobre cómo administrar grupos de trabajo de manera eficiente y satisfactoria en ambientes de alta productividad basados en modelos ágiles (Agile Model) para peopleware, con la especificación de escenarios y determinación de métricas. En la quinta sección proponemos el desarrollo, implementación y aplicación del modelo peopleware. En la sexta sección presentamos un marco de discusión y conclusiones de este trabajo. Finalmente mencionamos algunos lineamientos de trabajos futuros apoyados en el desarrollo de modelos empíricos.

Palabras clave: Peopleware, modelos ágiles, organismo de gobierno, sector público, administración pública, grupos de trabajo, satisfacción laboral.

1 Introducción

Tradicionalmente, la industria de tecnologías de la información o TI (Tecnología Informática), ha considerado como únicos aspectos el hardware y el software. En 1977 Peter G. Newmann usó el término Peopleware que fuera acuñado en 1980 por Meilir Page Jones, en 1987 por Tom DeMarco y Timothy Lister en el libro *Peopleware: Productive Projects and Team* [1]. Peopleware se refiere al factor humano en los proyectos de TI y como dice que quizás el principal problema de los equipos es que no son tecnológicos sino sociológicos.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

En *Achieving Business Agility* [2], se propone un framework para lograr la agilidad empresarial, que permite agilizar no sólo un grupo de trabajo, sino toda una organización. Para lo cual plantea que se debe dominar dos sistemas operativos comerciales. El primer sistema es principalmente jerárquico y es común a la mayoría de las empresas. Proporciona eficiencia, estabilidad y otros aspectos escalables necesarios para cumplir la misión actual (por ejemplo, operaciones con personas, servicios financieros, comerciales y cumplimientos). El segundo sistema es una red centrada en el cliente, que es vital para entregar rápidamente soluciones innovadoras a un mercado que se mueve más rápido.

El rol del líder es un aspecto fundamental para la adopción de prácticas ágiles. Jordy Alemany [3] en su página de LinkedIn, plantea a este respecto que las organizaciones muchas veces no solo son incapaces de atraer talentos sino que queman el que ya poseen por factores como jornadas eternas, urgencias constantes, ambientes tóxicos y falta de reconocimiento. Todas éstas, muy vigentes en el ámbito público. Además, que la principal razón por la que la gente abandona un trabajo, es un mal jefe y la falta de afecto y aprecio de los integrantes del equipo.

En este trabajo abordamos aspectos sociológicos de grupos de trabajo relacionados con la producción de sistemas y su ciclo de vida en ambientes TICs (Tecnologías Informáticas y Comunicaciones).

Quienes hemos trabajado en áreas directamente relacionadas con el desarrollo de sistemas, de infraestructura y la administración de proyectos, sabemos cuáles temas a veces nos preocupan, nos distraen, nos afectan de alguna manera generando malestar. En algunas ocasiones los asuntos más básicos generan preocupaciones que nos mantienen la mente ocupada con disconformidades. Los trabajadores estamos acostumbrados a arreglarnos con poco y de mala calidad, a trabajar de manera tan precaria que gran parte del tiempo estamos pensando como podrían ser mejor las cosas y el ambiente donde nos desenvolvemos, sin lograr mejores condiciones.

Por tales motivos, proponemos mejorar las condiciones laborales, el ambiente de trabajo y sus herramientas. Entre otros temas se trata de mejorar la comodidad del espacio de trabajo. Un lugar donde el empleado se sienta ansioso por llegar y orgulloso de pertenecer. Un lugar donde se sienta más a gusto y entretenido que en su casa, donde los empleados puedan conseguir ese espacio para desarrollar su creatividad. Para ello hay que alcanzar niveles de satisfacción laboral que hagan la diferencia, que las dimensiones del espacio asignado, los elementos de la oficina y su ambiente superen lo necesario y suficiente en alguna medida a estimar mediante algún cálculo que permita administrar los niveles de satisfacción de los grupos de trabajo.

2 Análisis de Idiosincrasia

Al analizar la idiosincrasia referida a las administraciones públicas municipales, provinciales o nacionales de Argentina, se puede observar que aplicar modelos de peopleware para gestionar grupos de trabajo en desarrollo de software puede ser una tarea de magnitudes importantes. Las organizaciones públicas tienen limitaciones en sus procedimientos administrativos funcionales que se encuentran asociados al organigrama o estructura organizacional, son regidos por las leyes de procedimientos administrativos nacionales, los diferentes códigos de procedimientos administrativos

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

provinciales y las numerosas resoluciones de los consejos municipales. Por ejemplo, la Ley (Argentina) 25.164 *Marco de Regulación del Empleo Público Nacional*, Publicada en el Boletín Oficial del 08-oct-1999 Número: 29247 Página: 1.

La administración pública tiene como fin primario cumplir con la función social de brindar al ciudadano bienestar. Para este fin, se vale también de herramientas de TI, para procesar con eficiencia los volúmenes importantes de información para la toma de decisiones. En este marco y para alcanzar este objetivo, tiene que ajustarse, igual que la industria, a cambios permanentes propios del aparato público, propiciados por la rotación del poder político y por la misma evolución tecnológica. Dicho ésto, la adopción de prácticas ágiles podrían brindar una potencial respuesta a esa dinámica, pues en la realidad, como lo sostiene la bibliografía desde hace varios años, la complejidad de los proyectos de TI del aparato público es mucho mayor que los de la industria, lo que se evidencia muchas veces por productos o servicios, de baja calidad.

Geoffrey A. Moore [4], plantea en su libro que el Ciclo Vital de Adopción de Tecnología está divido en 5 etapas. La quinta, que corresponde a los “rezagados”, es en la que los autores de este trabajo coincidimos en ubicar a la administración pública. No obstante, en los últimos tiempos se evidencia una tendencia a reubicar algunos sectores dentro de la cuarta etapa conocida como “mayoría tardía”. Un aspecto interesante en esta reclasificación es la presencia de generaciones jóvenes que entre otras cosas han tenido que enfrentarse a generaciones mayores quienes han defendido modelos tradicionales, ofreciendo gran resistencia al cambio.

En las organizaciones públicas, los equipos de TI, están sometidos al mismo régimen de horarios fijos de otras reparticiones. Con esta medida se presume que la creatividad necesaria para la productividad en éste área disciplinar, está circunscripta a un horario fijo, como la línea de producción de una fábrica.

El ambiente laboral de los equipos de trabajo, por lo general están dispuestos en ambientes compartidos por grupos de actividades heterogéneas, lo que propicia una alta incidencia de interrupciones. Teniendo en cuenta que la recuperación productiva después de una interrupción es de 15 minutos, Sutherland, J.; Schwaber, K. [5], es de esperar que las métricas propuestas de factor ambiental (E-Factor, environmental factor), reflejan índices muy bajos, que impactan en ineficiencia y frustración.

2.1 Casos de Organismos Públicos

Las organizaciones públicas presentan estructuras de trabajo de tipo piramidales, burocráticas y autocráticas, reglamentadas por convenios colectivos de trabajadores, centradas en el manual de funciones y no en personas o grupos de tareas. A esto, se suman barreras procedimentales, normativas y legales, que dificultan el abordaje de las reformas.

Respecto de los recursos humanos, observamos escenarios normales o comunes, donde su fuerza laboral presenta desánimo, apatía, descontento y malestar. Algunas de estas causas pueden ser monotonía de funciones, falta de incentivos, falta de capacitaciones periódicas, justamente porque se considera que la función no lo requiere, estancamiento en la carrera laboral, discriminación laboral y hasta injusticias con designaciones de personas incompetentes en cargos de toma de desición. Esto inclusive ocurre contradiciendo las mismas normas y leyes totalmente obsoletas que siguen rigiendo actualmente, p.ej., en la Provincia del Chaco - Argentina, existe la

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Ley 292-A [6] *Estatuto para el Personal de la Administración Pública Provincial* que reglamenta la manera de ingresar a la administración pública, explícitamente describe que el ingreso debe hacerse por concurso y el mismo estado provincial incumple dicha ley, realizando ya hace años diversas incorporaciones masivas aplicando leyes especiales para cada caso, prácticas que aún siguen vigentes.

En la actualidad, si se cumpliera estrictamente la ley respecto a selección de personas por concurso, también se consideran mecanismos obsoletos, ya que existen varios modelos muy eficientes de selección de personas que difieren de los modelos basados en concursos de antecedentes y oposición. Uno de ellos y tal vez el más conocido y utilizado es el denominado "assessment center" [7].

Jurgen Appelo dice en su libro *Management 3.0* [8], maneras de trabajar con las personas para que sean más productivas. No obstante, aún hoy en día, el sector privado de empresas en el mundo siguen buscando incentivar a sus recursos humanos mediante premios por resultados. El autor contradice estas prácticas, ya que tienden a estancar al personal empeorando sus resultados. Si se analiza este concepto en Argentina, cualquier recompensa monetaria o basada en resultados es difícil de llevar a cabo.

2.2 La Imposibilidad Legal de Aplicar Modelos Basados en Recompensas

Es fácil suponer que ni las áreas de recursos humanos ni los jefes de desarrollo de sistemas tengan conocimiento o apliquen técnicas ágiles de desarrollo de software y mucho menos modelos que consideren al peopleware. Si se piensa en las limitaciones de las organizaciones respecto a las posibilidades de realizar desarrollo de personas no sería muy difícil aplicar las seis reglas propuestas por Appelo [8], para las recompensas que brindan la mejor oportunidad de aumentar el desempeño de las personas y su disfrute del trabajo, al mismo tiempo que fomentan la motivación intrínseca en lugar de destruirla. Estas reglas pueden describirse de la siguiente forma:

- No comprometerse a brindar recompensas en forma adelantada.
- Si ese es el caso, mantener pequeñas recompensas.
- Es preferible brindar pequeñas recompensas continuamente, y no todas de una vez.
- Es mejor recompensar o felicitar en público, no en privado.
- Se debe recompensar el comportamiento o el camino recorrido y no el resultado.
- Es muy común hablar de "mis empleados", recompense a sus compañeros, no a sus subordinados.

Mecanismos difíciles de aplicar cuando la ley impone el concurso de cargos como único mecanismo de premiación o se basa en trabajo exclusivamente por horas presenciales sin posibilidad de utilizar otro modelo para buscar, evaluar y premiar el desempeño de personas.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

3 Implementar Prácticas de PeopleWare en Organismos de Gobierno

Las personas son componentes fundamentales de los sistemas, y las cosas son las herramientas, útiles, equipos, materiales e insumos sanitarios que permiten facilitarles el trabajo. La interacción entre las personas y sus cosas generan las relaciones necesarias (de funcionamiento y comportamiento) [9] que le dan a los sistemas la dinámica necesaria para mantenerse activos. De la misma forma que se requieren herramientas de buena calidad y de altas prestaciones, se vuelve indispensable contar con recursos humanos de características similares y variados, con buen nivel de capacidad intelectual y experiencias, puesto que la diversidad en el grupo es una característica indispensable para generar dinámica y mantener la estabilidad del sistema [8][9]. La TGS (Teoría General de los Sistemas) [9] describe las características fundamentales que definen a los sistemas como tales, una de ellas es justamente la variedad o diversidad.

3.1 Resumen de Casos en Universidades Públicas

Un caso típico es la administración o gobernanza de una universidad pública, donde cada área o departamento (grupo de trabajo) sigue un procedimiento basado en el tratamiento de expedientes presentados. Los individuos que intervienen en ese proceso, en algunos casos utilizan recursos de apoyo como computadoras y sistemas que le permiten agilizar o reducir los tiempos requeridos para el “pronto despacho” del trámite.

En estos tipos de grupos no se observa que fuese necesario desarrollar nada por ellos mismos, en los casos en que se requiera modernizar alguna etapa del trámite, y si se piensa en una solución informática, se solicitará la intervención del área de sistemas. Sin embargo, como describiremos en la sección cuatro, se pueden tratar algunos temas relacionados al bienestar y aceptación de condiciones laborales para estos grupos de tareas.

3.2 Caso Específico de Un Grupo de Referencia

En referencia al organismo público de tipo universitario descrito en 3.1 podría ser el personal del área de sistemas o TIC (Tecnología Informática y Comunicaciones). Este grupo normalmente se encuentra sometido a cierto nivel de presión, puesto que cada vez más los modelos funcionales de las diferentes áreas convergen a los sistemas que son desarrollados, mantenidos y soportados por el área TIC, además del mantenimiento de la infraestructura de datos y el soporte técnico de los usuarios.

La falta de presupuesto para la modernización de sistemas, la falta de recursos humanos, la ausencia de capacitación, de tiempo, comodidades, recursos técnicos y herramientas, forman parte de un escenario operacional desagradable que puede considerarse normal.

Este ejemplo podría recaer casi en cualquier área y organismo de la administración pública ya sea relacionada a la informática o no. Las observaciones empíricas fueron cotejadas en extensas reuniones periódicas de trabajo con los profesionales a cargo de

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

las diferentes dependencias TICs de la UNNE (Universidad Nacional del Nordeste) y UTN (Universidad Tecnológica Nacional Resistencia).

4 Propuesta Práctica de Implementación en PeopleWare

En este trabajo proponemos un modelo como ejemplo de implementación que permita determinar el grado de satisfacción del equipo de desarrollo en ambientes gubernamentales y administrar de manera eficiente y satisfactoria a grupos de trabajo con perfil de productividad extrema [10]. No se trata de una medición real puesto que el modelo aún no ha sido desarrollado, sólo permanece en formato de propuesta, no obstante, los valores de las métricas fueron sugeridos por el grupo en base a valoraciones empíricas reales.

Para definir que es PeopleWare, se debería contextualizar un ambiente laboral, si el ambiente ya es de tipo Agile [11][12][13] se tendría que enmarcar el modo con el cual opera ese grupo de trabajo y con ello se simplificaría la tarea, aunque no sería obligatorio, puesto que el modelo debería funcionar para cualquier ambiente laboral incluso de tipo militar, con algunas variantes un poco más extremas impuestas por el tipo de mando super vertical.

4.1 Escenarios en PeopleWare

La burocracia y los cambios eventuales producidos por el personal de gestión (directivo) con cierta frecuencia interfieren en la implantación del modelo ágil [14].

En párrafos anteriores hemos descrito que las organizaciones públicas presentan estructuras obsoletas de trabajo, de tipo piramidales, burocráticas y autocráticas, centradas en el manual de funciones, los convenios colectivos de trabajadores, el trabajo a reglamento y no centradas en personas.

Resulta importante mencionar el concepto de Tribus, según Dave Logan, John King, & Halee Fisher-Wright en el libro *Tribal Leadership: Leveraging Natural Groups to Build a Thriving Organization* [15], al definir una tribu, se deben cumplir ciertos requisitos. Una tribu puede contener de 20 a 150 personas, las cuales pueden conocerse entre sí. Las tribus tienen una mejor convivencia, funcionan mejor que los equipos de trabajo. Los autores presentan a los líderes una propuesta de cómo evaluar la cultura tribal de la organización con una puntuación que va del 1 al 5 donde la más alta es la mejor.

Para este análisis situamos empíricamente a la mayoría de las organizaciones públicas entre las etapas 2 y 3, donde la segunda etapa se caracteriza porque las personas se encuentran sin motivaciones de realizar bien el trabajo, o lo realizan a demanda sin nuevas perspectivas de crecer personal ni grupalmente. Por lo general tienen primero el “no se puede”. En la tercera etapa, la persona cree ser el centro de la organización y el más importante de todos. El conocimiento es poder, por lo que la gente quiere acumularlo y no compartirlo con la idea de creerse indispensable, desde contactos con clientes hasta chismes sobre la empresa.

En primer lugar, para contextualizar sería necesario visualizar al menos dos escenarios, favorable o desfavorable, sobre un mismo grupo que permita contrastar diferencias significativas (ver Fig. 1).

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Al ser el ámbito de la administración pública tan rígido e inflexible a la hora de proponer mejores prácticas de peopleware se puede enfocar en detectar variables según sean controlables o no, y de ser necesario crear nuevas variables. Existen propuestas referidas a mejorar los ámbitos de trabajo, todos enfocados en grupos de trabajo pertenecientes a empresas.

Considerando el concepto de matriz de datos de Johan Galtung (1966) [16], revisado por Juan Samaja [17], la unidad de análisis para este trabajo de investigación es cada uno de los integrantes del equipo de desarrollo, la variable de análisis es la valoración del ambiente de trabajo, los indicadores de la misma están presentados en la Tabla 1, y el procedimiento utilizado para asignar valores es mediante una escala Likert a través de una encuesta.

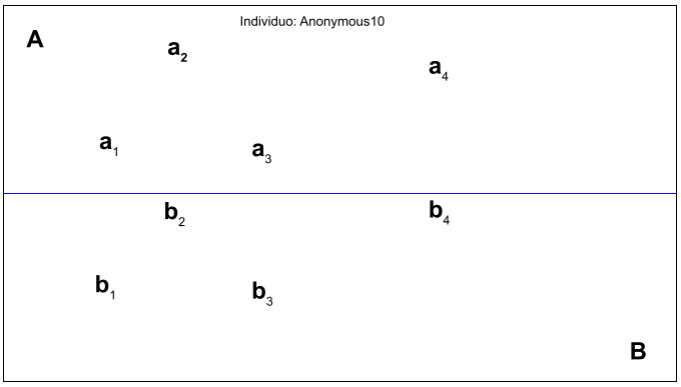

Fig. 1. Conjuntos de métricas agradables y desagradables por individuo.

a. Por un lado se asume que se debe tener un buen trato con los integrantes de los diferentes grupos de trabajo que conforman la fuerza laboral de la organización, en un ambiente de respeto, amistad y espíritu colaborativo. A estas métricas se las denomina virtudes agradables (a) para cada individuo.
b. Por otra parte se asume que hay que cumplir acuerdos contractuales con los clientes, entregar el producto final a tiempo y de calidad de acuerdo a las especificaciones establecidas, aparecen situaciones de presión, situaciones incómodas, falta de herramientas o carencias de algún tipo (las cosas). A estas métricas se las denomina obligaciones incómodas o desagradables (b) para cada individuo.

4.2 Ejemplo de Ponderación de Métricas

La contextualización de un ambiente laboral promedio debe ser lo más descriptivo posible, ajustado a una relación directamente proporcional, al aumentar el nivel de detalle se mejora o aumenta la definición de ambos contextos *a* y *b*, lo que llevará a un análisis de relaciones funcionales y de comportamiento [9] de los diferentes grupos que conforman la fuerza laboral de la organización con mayor precisión.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Para ello cada métrica debería ser considerada en su aspecto virtuoso *(a)* y en su aspecto desagradable *(b)*, puesto que un mismo individuo puede tener distintos grados de valoración subjetiva de la misma métrica.

La representación gráfica está ilustrada en escala 1:20, donde los vectores expresan magnitudes o pesos porcentuales, (ver Fig. 2).

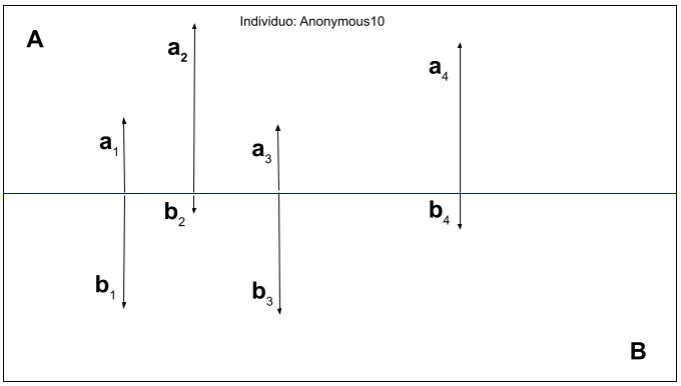

Fig. 2. Valoraciones por individuos.

Si bien la variable es la misma, puede tener miradas, apreciaciones o valoraciones diferentes, es decir, puede estar compuesta por la suma de magnitudes de aprobación a% y de desaprobación b%, donde b% es el resto de

$$100\% - a\% = b\% \quad (1)$$

o al revés, en caso de que se decida ponderar las métricas desde una valoración de desaprobación, donde

$$a\% = 100\% - b\% \quad (2)$$

Tabla 1. Ponderación de métricas por individuo.

Secciones y Métricas		Aprobaciones ***a*%**	Desaprobaciones ***b*%**
Calidad del Esparcimiento	$\mathbf{a_0}$	**61,25%**	**38,75%**
Completitud de la sala de recreación	a_1	40%	60%
Calidad del espacio para café	a_2	90%	10%
Comodidad del amoblamiento	a_3	35%	65%
Conformidad con la TV de sala	a_4	80%	20%
+ Agregar Item			
Calidad de los Sanitarios	$\mathbf{a_{10}}$	**63,3%**	**36,66%**
Considera adecuadas las toallas de papel	a_{11}	45%	55%
Le gusta la calidad del secador de aire	a_{12}	70%	30%
Le gusta el tipo de Jabón para las Manos	a_{13}	75%	25%
+ Agregar Item			
Agrado de su Espacio de Trabajo	$\mathbf{a_{20}}$	**76,75%**	**23,25%**
Considera adecuado su computador de escritorio	a_{21}	65%	35%
Le agrada su monitor o pantalla de escritorio	a_{22}	70%	30%

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Considera adecuado su espacio de almacenamiento	a_{23}	90%	10%
Considera adecuada las dimensiones del espacio físico asignado para su labor	a_{24}	80%	20%
Le agrada la calidad de su escritorio	a_{25}	82%	18%
+ Agregar Item			

4.3 Ejemplo de Análisis y Valoración de Métricas

En segundo lugar, del análisis de ese contraste entre a y b, surge una línea entre ambas condiciones funcionales tales que esta línea que divide ambas realidades claramente no es una recta, más bien tiene forma de curva sinuosa que busca el equilibrio entre las propiedades de a y b (ver Fig. 3).

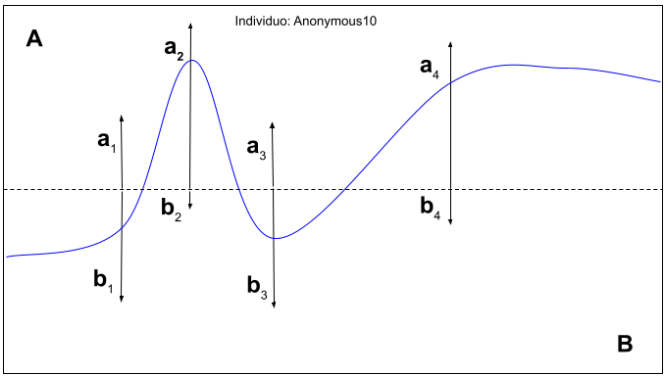

Fig. 3. Curva ponderada ajustada por valoraciones de individuo

Con la aplicación de este ejemplo de modelo propuesto para peopleware, se trata de ecualizar la curva de máxima eficiencia productiva manteniendo una funcionalidad agradable y sin que el grupo de trabajo se rompa.

5 Desarrollo e Implantación del Modelo PeopleWare

Se puede realizar una encuesta creada y analizada por el líder de la tribu, en este caso el jefe del área o producto de la elaboración grupal en conjunto con todos los miembros de la tribu. Esas variables pueden plasmarse en una encuesta. Una vez identificadas las variables (propuestas) que podemos manejar, las ponderamos a efectos de ver cuáles tienen más impacto en el grupo y posteriormente realizamos una encuesta individual. Los resultados de esa encuesta proporcionarán una idea clara de cuáles son los factores (secciones de la Tabla 1) que pueden influir como mejoras en el ambiente de trabajo y detectar cuales producen efectos negativos.

5.1 Aplicación de las Encuestas

Pueden plantearse diversas formas de aplicar las encuestas para evaluación. Basados en nuestra propia experiencia grupal, sería conveniente expresar los “reclamos” de

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

manera directa, accediendo voluntariamente a un formulario vía sistema web service, sin esperar la fecha u oportunidad de completar un cuestionario programado con regularidad. Mientras que las consultas para evaluación sobre el "nivel de satisfacción" se pueden aplicar de forma programada el último día laboral de la semana, con el formato de micro encuestas dirigidas sobre temas específicos, por ejemplo, sobre alguna sección de la Tabla 1.

El propósito de aplicar micro encuestas, es no generar demoras o malestar al sentirse el individuo obligado a responder un extenso cuestionario. Esto podríamos ajustarlo solamente con una prueba empírica de la aplicación del sistema.

6 Conclusiones

Dentro de lo que identificamos como organismos de gobierno no se han encontrado casos de referencia donde se hayan aplicado modelos de peopleware basados en esquemas ágiles. Tampoco se han hallado grupos que apliquen estos modelos en ambientes más específicos como en las universidades públicas, donde se esperaría tener equipos de trabajo que apliquen algunos de esos modelos actualizados. En estos últimos, hasta donde se ha podido observar, se siguen utilizando modelos administrativos tradicionales basados en escalafones con cierto grado de verticalidad bien pronunciada, y con sobre abundancia de jefes y subjefes.

A corto plazo no se observa viable la posibilidad de promover cambios en el modelo impuesto por el establishment. Sin embargo, la aplicación de modelos ágiles son comunes en empresas privadas del rubro informático, donde aparecen con mayor frecuencia y de forma más natural.

La incertidumbre permanente, los cambios constantes en las políticas gubernamentales, la falta de previsibilidad, de reglas claras y la rigidez de los procesos, deberían hacer un ámbito propicio para la aplicación de modelos ágiles orientados al desarrollo de software.

Lo cierto es que a corto plazo no se visualiza un panorama de aplicación de modelos ágiles, ni que los ambientes de trabajo apliquen técnicas de peopleware en la administración pública. No obstante, el modelo propuesto en la sección cuatro puede ayudar a contrarrestar los problemas anteriormente descritos.

7 Trabajos Futuros

De lo expuesto surge una clara línea de investigación y desarrollo futuro, proponer un estudio en profundidad apoyado con la aplicación del modelo empírico y el análisis estadístico sobre el funcionamiento individual y grupal en al menos una organización modelo, con la que se pueda validar correctamente los datos y sus efectos.

Dada la naturaleza de analizar valoraciones subjetivas sería de utilidad el uso de paradigmas como la lógica difusa y la teoría de funciones de clasificación en aprendizaje de máquinas.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

References

1. DeMarco, T.; Lister, T.: Peopleware: productive projects and teams. Addison-Wesley (2013)
2. Scaled Agile, Inc.: SAFe. Achieving Business Agility with SAFe® 5.0. Scaled Agile. https://techtalk.at/wp-content/uploads/2021/07/White_Paper_5.0_TechTalk.pdf. White Paper (2019). Accedido el 25 de Septiembre de 2022
3. Alemany, J.: LinkedIn. https://www.linkedin.com/in/jordialemanymonzo/. Accedido el 25 de Septiembre de 2022
4. Moore, G. A.: Crossing the chasm: Marketing and selling technology products to mainstream customers. HarperBusiness (1991)
5. Sutherland, J.; Schwaber, K.: Business object design and implementation: OOPSLA '95. Workshop Proceedings The University of Michigan. pp. 118, 1995. ISBN 3-540-76096-2
6. Ley 292-A - Estatuto para el personal de la administración pública provincial. *Poder Legislativo del Chaco*. http://www2.legislaturachaco.gov.ar:8000/Documentos/Ley/VistaPublicaLey/300135 (2014). Accedido el 25 de Septiembre de 2022
7. Capitán, Á. O.: Desarrollo metodológico de un "assessment center" basado en un sistema de gestión por competencias. *Lan harremanak: Revista de relaciones laborales*, No. (24), pp. 197-218 (2011)
8. Appelo, J.: Management 3.0: Leading Agile developers, developing Agile leaders. Addison-Wesley, Upper Saddle River, NJ (1969)
9. Von Bertalanffy, L.: Teoría General de los Sistemas: Fundamentos, Desarrollo, Aplicaciones. Primera Edición 1968. 1ª edn. en español (1976). Fondo de Cultura Económica, 7ª reimpresión, México (1989)
10. Gregory, P.; Lassenius, C.; Wang X.; Kruchten, P.: Agile processes in software engineering and extreme programming. *22nd International conference on Agile software development*, XP 2021 Virtual Event Proceedings (2021)
11. Schwaber, K.: Advanced Development Methods, Inc.. Control Chaos. http://www.controlchaos.com/ (2010); Scrum.Org: The Home of Scrum. https://www.scrum.org/scrumorg-trademarks-and-copyrights (2010). Accedido el 25 de Septiembre de 2022
12. Schwaber, K.: Controlled Chaos: Living on the Edge. Copyright Advanced Development Methods, Inc. (1996)
13. Schwaber, K.: Scrum Development Process. Copyright Advanced Development Methods, Inc. (1997)
14. Bollati, V. A.; Garzás, J.: Nuevas tendencias en el futuro de la agilidad. *Asociación de Técnicos de Informática*, Novatica, No. 240, pp. 1-22 (2018). *Repositorio institucional CONICET digital*. http://hdl.handle.net/11336/81240 (2018). Accedido 25 de Septiembre de 2022
15. Logan, D.; King, J.; Fisher-Wright, H.: Tribal Leadership: Leveraging Natural Groups to Build a Thriving Organization (2009)
16. Galtung, J.: Teoría y métodos de la investigación social. Eudeba, Buenos Aires (1966)
17. Samaja, J.: Epistemología y metodología. Elementos para una teoría de la investigación científica. 3ra ed. pp. 186. Eudeba (2004)
18. Gutiérrez Goicoechea, J. J.: Peopleware: Qué es y su importancia en el desarrollo de proyectos. CW OpenWebinars, https://openwebinars.net/blog/peopleware-que-es-y-su-importancia-en-el-desarrollo-de-proyectos/ (2021). Accedido el 25 de Septiembre de 2022
19. Takeuchi, H.; Nonaka, I.: The New New Product Development Game. Harvard Business Review (1986). https://hbr.org/1986/01/the-new-new-product-development-game. Accedido el 25 de Septiembre de 2022

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

20. Scrum Manager®, Iubaris Info 4 Media SL. C/ Bari 39 2ª Planta, (50197) - Zaragoza, España. Tel & Whatsapp: (+34) 644 441 617. admin@scrummanager.net, https://www.scrummanager.net/bok/index.php?title=New_New_Product_Development_Game. Accedido el 25 de Septiembre de 2022
21. Imai, K.; Nonaka, I.; Takeuchi, H.: Gestión del proceso de desarrollo de nuevos productos: cómo aprenden y desaprenden las empresas japonesas. División de Investigación, Harvard Business School (1984)
22. Soriano Pinilla, J. J.: Estudio de caso práctico de aplicación de metodologías AGILE a la enseñanza en formación profesional de grado superior. Trabajo fin de máster (Modalidad B), máster universitario en profesorado de educación secundaria obligatoria, bachillerato, formación profesional y enseñanzas de idiomas, artísticas y deportivas. pp. 6. Universidad de Zaragoza (2013)

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Uso de Técnicas Empíricas para la Evaluación del Impacto de las Emociones en la Calidad de Software

Gabriela Tomaselli[1], Cesar Acuña[1], Noelia Pinto[1]

[1]CInApTIC – Facultad Regional Resistencia – Universidad Tecnológica Nacional
{gabriela.tomaselli, csr.acn, ns.pinto}@gmail.com

Resumen. El uso de técnicas de Ingeniería de Software Empírica posibilita reunir evidencia a fin de corresponder con la realidad, ideas o teorías sobre la construcción de software, a través de mediciones y experimentos realizados de modo sistemático, disciplinado, cuantificable y controlado. En el marco del proyecto "Evaluación del impacto de las emociones en la calidad de software desde el punto de vista del usuario" se han llevado adelante diversas experiencias mediante la aplicación de técnicas empíricas destinadas a detectar las emociones de los usuarios en la utilización de productos de software y evaluar su impacto en la calidad percibida. El objetivo de este trabajo es presentar cada una de las técnicas, describiendo las experiencias y sus resultados.

Palabras clave: Ingeniería del Software Empírica, Calidad de Software, Emociones, Computación Afectiva.

1 Introducción

La Ingeniería de Software Empírica es la parte de la Ingeniería de Software que se enfoca en reunir evidencia, a través de mediciones y experimentos que involucran sistemas de software (productos de software, procesos y recursos) [1]. En los últimos años ésta área de la Ingeniería de Software ha cobrado gran importancia y su actividad ha sido creciente, buscando evaluar métodos, herramientas y técnicas propuestos de modo sistemático, disciplinado, cuantificable y controlado; con este objetivo se aplican técnicas empíricas basadas en la experimentación a fin de corresponder ideas o teorías con la realidad, es decir mostrar con hechos las especulaciones, suposiciones y creencias sobre la construcción de software [2,3].

La gestión de la calidad en las organizaciones dedicadas al desarrollo de proyectos de software ofrece una ventaja competitiva puesto que de esta forma aseguran que sus productos y servicios cuenten con estándares mínimos de calidad en el mercado; sin embargo, existen aspectos relacionados con el aseguramiento de la calidad que aún deben ser tratados, entre ellos la relación entre factores como experiencia de usuario y el impacto de las emociones en la calidad percibida por el usuario al interactuar con el software.

La Experiencia del Usuario (UX) es definida por Arhippainen y Tähti [4] sencillamente como la experiencia que obtiene el usuario cuando interactúa con un producto en condiciones particulares. Muchas veces esta experiencia se ve afectada por

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

emociones del usuario, por lo que el análisis de ambos factores combinados es fundamental para analizar el impacto de cierto producto o servicio en el mercado. En la actualidad, y cada vez con mayor frecuencia, se ha comenzado a considerar fuertemente la influencia de las emociones sobre los procesos de pensamiento racional, y cómo éstas afectan en gran medida el proceso de toma de decisiones en los seres humanos. Con relación a esta consideración cobra relevancia el concepto de Computación Afectiva (CA), definido como la disciplina científico-tecnológica que trata sobre el reconocimiento y generación de emociones por parte de las computadoras [5]. Los aspectos afectivos o emocionales en el procesamiento que realizan las computadoras podrían lograr una mejor adecuación a las necesidades de los usuarios [6] y con ello, mejorar la calidad percibida por las personas usuarias del software.

El punto de partida es comprender qué son las emociones, adoptándose a este fin la definición planteada por Ekman: "la emoción es una reacción a eventos considerados relevantes a las necesidades, metas o preocupaciones de un individuo, que existe durante un tiempo determinado (segundos, como máximo minutos)" [7].

Actualmente existen numerosos estudios e investigaciones dirigidos a las emociones que se generan en los usuarios, a la hora de utilizar aplicaciones de software. Las empresas reconocen el impacto de las emociones en las experiencias de los usuarios, sin embargo este campo aún no encuentra modelos asertivos a la hora de diagnosticar, evaluar y proponer mejoras [8], y aún más, entender de qué manera impactan en la industria del software.

A partir de lo previamente expresado surge la motivación del proyecto "Evaluación del impacto de las emociones en la calidad de software desde el punto de vista del usuario", dentro del cual se encuadra el trabajo que aquí se presenta, y que a su vez forma parte de una línea de investigación más amplia y madura dentro del Centro de Investigación Aplicada en Tecnologías de la Información y Comunicación (CInApTIC), relacionada a la Ingeniería y Calidad de Software orientado a PyMEs de la región Noreste Argentino (NEA).

En el marco de este proyecto se han llevado adelante diversas actividades destinadas a detectar las emociones de los usuarios en la utilización de productos de software y evaluar su impacto sobre la percepción de la calidad; dichas actividades comprenden distintas técnicas propias de la Ingeniería del Software Empírica, a saber: revisiones sistemáticas de la literatura, encuestas, y Focus Group. El objetivo de este trabajo es describir cada una de ellas y presentar los resultados obtenidos mediante su aplicación.

El resto del trabajo se estructura de la siguiente manera: en la Sección 2 se describen las experiencias llevadas a cabo por aplicación de diversas técnicas empíricas, y en la última Sección se mencionan las conclusiones y trabajos futuros relacionados.

2 Desarrollo

La Ingeniería de Software Empírica utiliza métodos de investigación reconocidos y probados a fin de generar conocimiento científico que reduzca la brecha entre la teoría y la práctica, aumentando el cuerpo de conocimiento y a la vez mejorando las técnicas de modo que los resultados de la investigación sean útiles y aplicables en la industria.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

En esta sección se presentarán cada una de las técnicas empíricas utilizadas durante el desarrollo del proyecto, describiendo las experiencias y sus resultados.

2.1 Técnica: Revisión Sistemática de la Literatura

Una Revisión Sistemática de la Literatura (RSL) es un método para identificar, evaluar e interpretar todas las investigaciones pertinentes a una determinada pregunta de investigación, área temática o fenómeno de interés [9]. Con el objetivo de dar respuesta a una o más preguntas específicas, se trata de resumir la investigación académica al respecto, siguiendo una estrategia de búsqueda imparcial y reproducible, que no solamente proporcione material de análisis, sino que también permita identificar vacíos en determinadas áreas de interés.

A partir de las publicaciones de Kitchenham sobre revisiones sistemáticas de literatura [10,11] y su artículo de Ingeniería de Software Basada en Evidencias (EBSE, por sus siglas en inglés) [12], su utilización cobró relevancia como herramienta para la obtención de evidencias científicas, y desde entonces gran número de RSL fueron realizadas y publicadas.

Las RSL permiten recolectar y sintetizar evidencia de distintas fuentes, distinguiéndose de las revisiones tradicionales narrativas o clásicas por contar con un enfoque metodológico destinado a minimizar la posibilidad de llegar a conclusiones erradas, que puedan resultar de sesgo en los estudios primarios o en el proceso de revisión [13].

Las RSL llevadas a cabo siguen los lineamientos establecidos por Kitchenham en [11], dividiéndose en tres fases principales: Planificación de la revisión, Ejecución de la revisión, y Presentación de resultados. La planificación resulta en una guía para conducir el proceso de revisión sistemática, a través de las siguientes etapas: Formulación de preguntas, Selección de fuentes de datos, Procedimiento de selección, y Valoración de la calidad.

Luego, la ejecución de la revisión consiste en poner en práctica el protocolo establecido para extraer los datos y obtener el conjunto de estudios relevantes que permiten dar respuesta a las preguntas de investigación. La fase final involucra redactar y presentar los resultados de la revisión y eventualmente difundir los resultados obtenidos.

RSL 1. En [14] se llevó a cabo una RSL con el objetivo principal de estudiar evidencias empíricas del análisis de las emociones y la interacción emocional de personas con sistemas, a fin de contribuir al estudio de la influencia de las emociones del usuario final en su percepción sobre la calidad del software. Este trabajo se enmarcó en las actividades preliminares relacionadas con el proyecto de investigación y desarrollo “Evaluación del impacto de las emociones en la calidad de software desde el punto de vista del usuario”.

Como resultado de esta investigación se puso de manifiesto la relevancia del análisis de la interacción emocional de los usuarios en la evaluación de calidad del software, demostrando, a partir del estudio empírico de diversos casos, que ya no basta solo con estudiar características del proceso de desarrollo y del producto final para medir la calidad del software.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Además, si bien el estudio demostró la evidencia de iniciativas que abordan las emociones de los usuarios y la calidad del software, se concluyó que el conocimiento existente no aporta propuestas que permitan la definición de modelos o estrategias que evalúen el impacto de las emociones en la calidad de software percibida por el usuario.

RSL 2. En [15] se presentaron discusiones y resultados que se lograron luego de un proceso de RSL respecto a estudios que proponen contribuciones para la evaluación de emociones en experiencias de usuarios al interactuar con software.

Si bien, los resultados de investigación a los que se arribaron demostraron la existencia de alternativas que trabajan la evaluación de emociones de usuarios de software, fue notoria la necesidad de contar con una propuesta que, independientemente del entorno, permita captar las emociones a partir de experiencias de uso, y relacionar esto con la calidad final percibida.

La RSL dejó al descubierto limitaciones en cuanto al estudio de las emociones de los usuarios y la calidad del software percibida, debido a la ausencia de propuestas que incluyan, de forma integral, la definición de un modelo junto a una estrategia de evaluación del impacto de las emociones en la calidad de software percibida por el usuario.

RSL 3. El objetivo de [16] era identificar investigaciones que hayan implicado la implementación de sistemas para el reconocimiento de emociones, mediante la detección del compromiso emocional del usuario mientras interactúa con aplicaciones de software, para lo cual se realizó una RSL destinada a analizar el estado del arte actual respecto a tecnologías y aplicaciones disponibles para evaluar las emociones que se producen en las interacciones de personas con software.

Los resultados de esta investigación pusieron de manifiesto la relevancia del análisis de la interacción emocional de los usuarios en la evaluación de calidad del software, demostrando, a partir del estudio empírico de diversos casos, la importancia de contar con aplicaciones que permitan automatizar la medición de emociones para luego interpretar el resultado de dichas interacciones y evaluar la calidad del software.

Asimismo, si bien el estudio mostró evidencia de iniciativas que utilizan la tecnología para la evaluación de emociones en usuarios de software, se concluyó que el conocimiento existente no aporta propuestas que permitan la definición de modelos o estrategias que evalúen el impacto de las emociones en la calidad de software percibida por el usuario.

2.2 Técnica: Encuesta

Las encuestas son, probablemente, el método de investigación más utilizado por todo el mundo; son investigaciones que proporcionan una visión general, mediante la recogida de información estandarizada de una población específica o una muestra representativa de la misma (sujetos del estudio), por medio de un cuestionario o entrevista [17].

Una encuesta es un método empírico utilizado para recopilar información de o sobre personas para describir, comparar o explicar su conocimiento, actitudes o comportamiento; en la mayoría de los casos, los datos relativos a la encuesta provienen

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

de formularios o cuestionarios, pero estos por sí solos no constituyen la encuesta. La encuesta como método de investigación es un proceso complejo compuesto por las siguientes actividades: Establecer los objetivos de la encuesta; Diseñar la encuesta; Desarrollar el cuestionario; Evaluar y validar el cuestionario; Obtener los datos de la encuesta; Analizar los datos obtenidos; y Reportar los resultados [18].

Un aspecto importante a ser tenido en cuenta al diseñar una encuesta es el mecanismo de administración, esto es: Cuestionarios auto-administrados (vía Internet), Encuestas telefónicas, o Entrevistas personales, siendo la primera de estas la opción más usada en Ingeniería de Software.

Encuesta 1. Los aspectos emocionales en el uso del software deben ser considerados ya no solo como mecanismo de análisis de la aceptación de software por parte de las personas usuarias, sino también en el propio ciclo de desarrollo mediante la utilización de herramientas y métodos para medir emociones o experiencias de emoción y utilizar los resultados para mejorar productos y servicios, con el fin último de aumentar su calidad final. En [19] se presentaron los resultados de la encuesta realizada a empresas de la Industria del Software, particularmente aquellas de la región NEA, a través de un cuestionario online como instrumento de recolección de evidencia empírica, con el objetivo de obtener un panorama general en cuanto a la consideración del impacto de las emociones sobre la percepción de la calidad de software por parte de las empresas.

La realización de este estudio permitió no sólo obtener un diagnóstico de la situación actual, sino también conocer si las empresas tienen en cuentas las emociones del usuario, si utilizan métodos o tecnologías para evaluarlas, y en el caso de no hacerlo, si consideran necesario implementarlo. Los resultados obtenidos mostraron que, si bien apenas un muy pequeño porcentaje de las personas encuestadas ha evaluado las emociones que se generan al usar sus productos o servicios de software, entre quienes no lo hicieron la mayor parte alegó desconocimiento, en tanto en ningún caso calificaron la propuesta como irrelevante con relación a la calidad percibida de sus productos o servicios. Asimismo, se comprobó que la mayoría de las personas que participaron de la encuesta considera importante tener en cuenta el estudio de las emociones que sus productos generan en quienes lo usan, y ante la consulta acerca de si considerarían implementar tal evaluación, la totalidad de las respuestas fueron afirmativas.

Encuesta 2. Con el objetivo de conocer la relación que existe entre las emociones que percibe una persona al utilizar un software y la calidad percibida del mismo, en [20] se presentaron los resultados de la encuesta realizada a personas usuarias de tres aplicaciones web, a través de un formulario online como instrumento de recolección de evidencia empírica.

A efectos de relevar las emociones más frecuentemente percibidas por los usuarios al utilizar software, se utilizó el test PrEmo, desarrollado por Pieter Desmet, que considera 14 emociones producidas en la interacción con un producto, divididas en dos grupos: las positivas, que incluyen alegría, admiración, orgullo, confianza, satisfacción, fascinación y deseo, y las negativas, tristeza, miedo, vergüenza, desprecio, furia, aburrimiento y disgusto; cada emoción está representada mediante animaciones con

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

expresiones dinámicas faciales y corporales, lo que convierte a PrEmo en una herramienta muy visual [21].

Los resultados de esta investigación pusieron de manifiesto la diversidad de emociones que se producen en las interacciones de personas con software y, en consecuencia, la importancia que se debe dar a dichas emociones al estudiar la relación que poseen con el impacto sobre la percepción de la calidad del software. Asimismo, el estudio permitió definir la hipótesis que mientras mayores sean las emociones positivas percibidas por una persona al utilizar un software, mayor será la tendencia a percibir una calidad más aceptable del producto, permitiendo abordar como trabajo futuro investigar la medida en que estas influyen positiva o negativamente sobre la calidad percibida de las aplicaciones.

2.3 Técnica: Focus Group

Los Focus Groups surgieron como método de investigación social en la década de 1950; son debates cuidadosamente planificados, diseñados para obtener las percepciones de los miembros del grupo sobre cierta área de interés.

La técnica Focus Group está orientada principalmente a la recolección de información cualitativa; consiste en dividir a los entrevistados en grupos de entre 3 y 12 participantes, siendo guiada y facilitada la discusión por uno o más moderadores, siguiendo una estructura predefinida y realizando preguntas para así conseguir datos significativos [22].

Se considera que este método tiene como beneficios producir información franca, a veces reveladora, y que es bastante barato y rápido de realizar; sin embargo, como debilidades surgen los sesgos causados por la dinámica de grupo y el tamaño de las muestras, en general pequeñas, por lo que puede dificultarse la generalización de los resultados [23].

Focus Group 1. El objetivo de [24] fue avanzar en la identificación de la relación existente entre las emociones que experimenta una persona al utilizar software y su calidad percibida, buscando validar y complementar resultados previamente obtenidos.

Con este objetivo, se realizaron una serie de entrevistas utilizando la técnica *Focus Group*, la cual permite llevar adelante un proceso de investigación cuali-cuantitativa centrado en la observación de ciertas emociones en experiencias de uso de software, bajo un contexto controlado del que participaron usuarios, moderadores y observadores.

Los resultados de la experiencia realizada corroboraron los obtenidos anteriormente, observándose varias semejanzas en las emociones detectadas previamente mediante la encuesta de aplicación del test PrEmo; la técnica de Focus Group aportó información extra de carácter cualitativo que permitió validar los datos cuantitativos obtenidos, ampliando los mismos mediante observaciones directas en experiencias presenciales, y aportó nuevos puntos de vista al análisis, relevantes a la identificación de emociones de los participantes.

Además, los resultados reforzaron la hipótesis de la correlación existente entre las emociones experimentadas en la utilización del software y la calificación otorgada con relación a su calidad y usabilidad.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

3 Conclusiones y trabajos futuros

La utilización de técnicas de Ingeniería del Software Empírica permite reunir evidencia destinada a contrastar teorías y propuestas con hechos mediante la experimentación sistemática y controlada. Este trabajo mostró los resultados de la aplicación de distintas técnicas de recolección de evidencia empírica con el fin de identificar las emociones de las personas usuarias en la utilización de productos de software y evaluar su impacto en la calidad percibida.

Las revisiones sistemáticas de la literatura llevadas a cabo pusieron en evidencia, por un lado que, si bien existen investigaciones incipientes al respecto, presentan limitaciones en cuanto a la relación entre las emociones del usuario y la calidad de software percibida, y por otro lado, las tecnologías de las que se dispone para evaluar emociones en usuarios de software, no se encuentran propuestas que incluyan, de forma integral, la definición de un modelo junto a una estrategia de evaluación del impacto de las emociones de la calidad de software percibida por el usuario.

La encuesta aplicada a empresas de la industria del software del NEA mostró que, si bien solo un muy pequeño porcentaje de las personas encuestadas evaluó las emociones que se generan al usar sus productos o servicios, existe un gran interés de la industria por contar con herramientas que favorezcan la obtención de productos y servicios de software considerando a las personas usuarias y sus emociones.

A través de otra instancia de encuesta, involucrando personas usuarias de aplicaciones web y empleando como herramienta de clasificación de emociones el test PrEmo, se comprobó que como resultado de dicha interacción se generan distintas emociones, en su mayoría positivas.

Con el objetivo de ampliar y validar esa experiencia, se realizó un relevamiento utilizando la técnica Focus Group, que permitió obtener datos cuali-cuantitativos centrados en la observación de las emociones, y reafirmó la hipótesis de la existencia de una correlación entre las emociones y la calidad de software percibidas.

Analizar las emociones y comportamiento de las personas al usar aplicaciones de software, brindan un marco apropiado para, no sólo detectar, medir y analizar dichas emociones o respuestas de comportamiento, sino también para poder proponer modificaciones al software, que permitan mejorar su calidad final. Incluso, en este mismo escenario, resulta de interés considerar la medida en que la evaluación de las emociones que experimentan los usuarios al utilizar software puede influir en su proceso de desarrollo, retroalimentando actividades o incluso generando nuevas.

Los resultados descriptos constituyen el punto de partida permitiendo identificar las emociones preponderantes de las personas al utilizar el software. Como trabajos futuros, se pretende, en base a la información obtenida a partir de las experiencias llevadas a cabo, avanzar en la definición de un modelo de evaluación de calidad centrado en la relación entre emociones generadas desde la experiencia de uso y la percepción de calidad del software por parte del usuario.

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

Agradecimientos

Este trabajo se enmarca en las actividades relacionadas con el proyecto de investigación y desarrollo “Evaluación del impacto de las emociones en la calidad de software desde el punto de vista del usuario” (PID: SIUTIRE0005517TC), del Centro de Investigación Aplicada en Tecnologías de la Información y Comunicación (CInApTIC), de la Facultad Regional Resistencia, financiado por la Secretaría de Ciencia y Tecnología de la Universidad Tecnológica Nacional. Agradecemos a la Ing. Dafne Torres y a los estudiantes de la carrera de Ingeniería en Sistemas de Información Rodrigo Cuevas y Mateo Mecozzi, todos ellos becarios de investigación, y muy especialmente a la Arq. María José Kiszka, por sus invaluables aportes a la realización de las experiencias aquí descritas.

Referencias

1. Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., & Wesslen, A. (2000). *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers.
2. Zelkowitz, M., & Wallace, D. (19917). Experimental validation in software engineering. *Information and Software Technology*, *39(11)*, 735-743.
3. Basili, V., Shull, F., & Lanubile, F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, *25(4)*, 456-473.
4. Arhippainen, L., & Tähti, M. (2003). Empirical evaluation of user experience in two adaptive mobile application prototypes. En *MUM 2003. Proceedings of the 2nd International Conference on Mobile and Ubiquitous Multimedia* (pp. 27-34). Linköping University Electronic Press.
5. Picard, R.W. (1999). Affective Computing for HCI. *HCI*, (1), 829-833.
6. Picard, R.W. (2003). Affective computing: challenges. *International Journal of Human-Computer Studies*, 59(1-2), 55-64.
7. Ekman, P. (1994). Moods, emotions, and traits. *The nature of emotion: Fundamental questions* (pp. 56-58). Oxford University Press.
8. Van Hout, M. (2008). Comprendiendo, midiendo, diseñando (para la) emoción. *Revista Faz*, *2*, 88-97.
9. Pizard, S., Acerenza, F., Casella, V., Moreno, S., García, R., Lezama, J., & Vallespir, D. (2019). *Conceptos de ingeniería de software basada en evidencias: Versión 2.* https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/25557/1/PIZ19.pdf
10. Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004), 1-26.
11. Kitchenham, B., & Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering, *Technical Report EBSE 2007-001*, Keele University and Durham University Joint Report.
12. Kitchenham, B., Dybå, T., & Jørgensen, M. (2004). Evidence-based software engineering. En *Proceedings of the 26th International Conference on Software Engineering (ICSE)* (pp. 273-281). IEEE.
13. Dybå, T., & Dingsøyr, T. (2008). Strength of evidence in systematic reviews in software engineering. En *Proceedings of International Symposium on Empirical Software Engineering and Metrics (ESEM)* (pp. 178–187).

AJEA- Actas de Jornadas y Eventos Académicos de UTN
DOI: https://doi.org/10.33414/ajea.1180.2023

14. Pinto, N., Acuña, C.J., Tomaselli, G., & Tortosa, N. (2019). Impacto de las emociones del usuario en la percepción de la calidad del software: Una revisión sistemática. En *Actas del 7mo Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI 2019)* (pp. 533-541).
15. Pinto, N., Torres, D., Acuña, C., & Tomaselli, G. (2020). Hacia la evaluación de emociones en experiencias de uso de software: Una revisión sistemática. En *Actas del 8vo Congreso Nacional de Ingeniería Informática – Sistemas de Información (CONAIISI 2020)* (pp. 74-81).
16. Tortosa, N., Ibañez, L., Alegre, N., Pinto, N., & Acuña, C.J. Evaluación del impacto de las emociones en la Calidad del Software: Una revisión sistemática de tecnologías para evaluar emociones de personas al usar software. (2020). En *Proceedings del 2020 IEEE Congreso Bienal de Argentina (ARGENCON)*.
17. Genero Bocco, M., Cruz Lemus, J.A., & Piattini Velthuis, M.G. (2014). *Métodos de Investigación en Ingeniería del Software*. Editorial Ra-Ma.
18. Kitchenham, B.A., & Pfleeger, S.L. (2008). Personal opinion surveys. En *Guide to advanced empirical software engineering* (pp. 63-92). Springer.
19. Tomaselli, G., Acuña, C.J., Pinto, N., & Torres, D. (2021). Vinculación Universidad-Industria: Relevamiento sobre Impacto de las Emociones en Calidad de Software. En *Actas Congreso Argentino y Latinoamericano de Ingeniería 2021 (CADI CLADI CAEDI 2021).*
20. Tomaselli, G., Alegre, N., Cuevas, R., Acuña, C.J., & Pinto, N. (2021). Emociones en el uso de software: Una experiencia de relevamiento utilizando PrEmo. En *Libro de Actas de las Cuartas Jornadas de Calidad de Software y Agilidad (JCSA 2021)* (pp. 74-83).
21. Pérez, G. (2013). *Diseño Emocional: Metodologías y herramientas para cuantificar emociones.* https://wiki.ead.pucv.cl/Diseño_Emocional:_Metodologías_y_herramientas_para_cuantificar_emociones
22. Soares Silva, I.; Keating, J.B.; Veloso, A.L. (2014). Focus group: Considerações teóricas e metodológicas. *Revista Lusófona de Educação*, (26), 175-189.
23. Kontio, J., Lehtola, L., & Bragge, J. (2004). Using the focus group method in software engineering: obtaining practitioner and user experiences. En *Proceedings of 2004 International Symposium on Empirical Software Engineering (ISESE '04)* (pp. 271-280).
24. Tomaselli, G., Acuña, C., Pinto, N., & Kiszka, M.J. (2022). Relevando emociones en el uso de software: Una experiencia empírica. En *Actas del 6° Congreso Argentino de Ingeniería / 12° Congreso Argentino de Enseñanza de Ingeniería (CADI CAEDI 2022).*

JCSA 2022

V EDICIÓN | 24 Y 25 DE NOVIEMBRE

Jornadas de Calidad de Software y Agilidad

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL RESISTENCIA

UNIVERSIDAD NACIONAL DE MISIONES

FaCENA - UNNE